WebSphere Development Studio Client for iSeries
Advanced Edition

EGL Server Guide for iSeries

Version 6 Release 0

SC31-6841-01

WebSphere Development Studio Client for iSeries
Advanced Edition

EGL Server Guide for iSeries

Version 6 Release 0

SC31-6841-01

Note
Before using this document, read the general information under Chapter 7, "Notices," on page 31.

Second Edition (May 2005)

This edition applies to version 6, release 0 of WebSphere Development Studio Client for iSeries Advanced Edition (product number 5724-D46) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. Installing and configuring EGL Server for iSeries

This chapter contains general information on the installation and customization of EGL Server for iSeries on the host and EGL on the workstation.

Installing EGL Server for iSeries

EGL Server for iSeries is available in the following WDSC plugin, which is in your installation directory under iseries\eclipse\plugins:

```
com.ibm.etools.egl.generators.cobol.iseriesruntime
```

In the plugin, the following binaries are included in the executables directory:

1. QEGL.zip, which contains the java parts for the gateway server. You must ftp this file (in binary) to the iSeries™ system, and unzip the contents into an Integrated File System directory named /QEGL
2. QEGL.SAVF, which contains the primary library for those system objects that constitute the EGL server for iSeries. You must ftp this file (in binary) to the iSeries system, and use the RSTLIB command to restore the contents into a library named QEGL

Objects created or replaced during installation

The following list provides a general description of the objects created or replaced during the installation process:

Table 1. Objects Created or Replaced during the Installation Process

Object and library name	Type	Description
QEGL QSYS	*LIB	The primary library for those system objects that constitute EGL Server for iSeries.
QVGN* QGPL	*FILE	Contains various database and source files, some of which are used during development and run time of EGL applications. See "iSeries library and file setup" for a description of these files.
/QEGL	DIR	The Integrated File System directory where Java™ parts for the gateway server reside.

iSeries library and file setup

The outputs from generation are placed into files in the library identified in the build descriptor option **destLibrary**. The default value of that option is QGPL.

You must create a set of files in that library before the preparation step can run. The next table lists those files.

Table 2. Generation Output Files

File name	Type	Description
QVGNCBLS	PF-SRC	EGL generation COBOL source
QVGNCLS	PF-SRC	EGL generation CL source
QVGNDDSS	PF-SRC	EGL generation DDS source file

Table 2. Generation Output Files (continued)

File name	Type	Description
QVGNEVF	PF-SRC	EVF parts control file
QVGNMAPG	PF-DTA	EGL generation form group source
QVGNTAB	PF-DTA	EGL generation table data
QVGNWORK	PF-SRC	EGL generation work file

The following commands can be used to create these files:

```
CRTSRCPF FILE(QGPL/QVGNCBLS) RCDLEN(92 ) TEXT('EGL GENERATION - COBOL SRC')
CRTSRCPF FILE(QGPL/QVGNCLS ) RCDLEN(92 ) TEXT('EGL GENERATION - CL SRC')
CRTSRCPF FILE(QGPL/QVGNDDSS) RCDLEN(92 ) TEXT('EGL GENERATION - DDS SRC')
CRTSRCPF FILE(QGPL/QVGNEVF ) RCDLEN(92 ) TEXT('EGL GENERATION - VARIABLES')
CRTSRCPF FILE(QGPL/QVGNWORK) RCDLEN(150) TEXT('EGL GENERATION - WORK FILE')
CRTPF FILE(QGPL/QVGNMAPG) SRCFILE(QEGL/QVGNPDDS) SRCMBR(TBLMAP) MBR(*NONE)
  TEXT('EGL GENERATION- MAP GROUP FILE')
  MAXMBRS(*NOMAX) AUT(*CHANGE) OPTION(*NOSRC *NOLIST)
CRTPF FILE(QGPL/QVGNTAB) SRCFILE(QEGL/QVGNPDDS) SRCMBR(TBLMAP) MBR(*NONE)
  TEXT('VISUALGEN TABLE DATA') MAXMBRS(*NOMAX) AUT(*CHANGE) +
  OPTION(*NOSRC *NOLIST)
```

To avoid member name collisions when multiple application developers are using the same host iSeries system, it is highly recommended that you copy these QVGN* files from the QGPL library to the application-developer user library that is identified in the build descriptor option **destLibrary**.

To create libraries for multiple users, do as follows:

1. Type the following command on the command line--

   `CRTLIB xxxxxx`

   where *xxxxxx* is the library name
2. Create a duplicate of the VGN files in the new library by typing this command--

   `WRKOBJ OBJ(QGPL/QVGN*) OBJTYPE(*FILE)`
3. Place *option=3* (copy) next to each file, then type this command--

   `TOLIB(XXXX)`

Alternatively, you may use the REXX program EGLSETUP in QEGL/QREXSRC to create these files. The command is as follows:

STRREXPRC SRCMBR(EGLSETUP) PARM(*xxxxxx*

xxxxxx
The library name

If you are using client server support for EGL to call non-EGL-generated programs, locate QVGNRNCL in the QVGNSAMP file of your QEGL library. Run CRTCLPGM on it and place it in any library that contains a non-EGL program that will be called from an EGL-generated client. Failure to prepare this for your non-EGL programs will result in unresolved references to QVGNRNCL.

iSeries preparation script file FDAPREP

To process the build plan successfully, the preparation script FDAPREP is invoked on iSeries by the build server. The script normally resides in the QEGL/QREXSRC file, but can be copied to another location and customized.

The script is invoked, by the build server, using the STRREXPRC command: STRREXPRC SRCMBR'(FDAPREP)' SRCFILE'(*LIBL/QREXSRC)' ... (other parms) Thus it is necessary to have the file containing the preparation script in a file included in *LIBL.

The preparation script is written in standard rexx, and you may want to modify it for customization purposes. The SYMPARM variables defined by the user in the build descriptor at generation time are available to this program as standard rexx variables and they can be used to influence the logic according to your needs. In addition, some standard variables are always defined, as in the following example:

```
EZENLS="ENU"
EZEPID=""
EZEDATA="31"
EZEENV="ISERIESC"
EZEMBR="iTest"
EZEGTIME="08:57:38"
EZEGMBR="iTest"
EZEGDATE="02/23/04"
EZESQL="N"
EZETRAN="iTest"
EZEFUNCTION="PMN"
EZEDESTLIBRARY="TEST"
```

The generation step creates a variables file appname.evf, which contains the variables and is passed to the preparation script.

Customizing EGL

If you have installed EGL Server for iSeries in a language other than the following languages, EGL Server for iSeries creates an abbreviated conversion table in the file QVGNSCTB of the QEGL library.

Suffix	Language
ENU	U.S. English
ENP	Uppercase English
CHS	Simplified Chinese
DEU	German
DES	Swiss German
ESP	Spanish
JPN	Japanese (Katakana)
KOR	Korean
PTB	Brazilian Portuguese

General considerations for EGL Server for iSeries

This section describes the general considerations for administering EGL Server for iSeries.

Using data description specifications generated by EGL

During generation, EGL can generate data description specifications (DDS) information from EGL record definitions that are used for file I/O operations.

The DDS information generated by EGL is useful only to the iSeries system administrator or application developer. The system administrator can use the DDS source members, or modified versions of them, to create the files that do not already exist on the iSeries system. Using the DDS source information to create the files qualifies these files for iSeries data management functions, such as specifying key fields, unique keys, and logical files.

You are not required to use the DDS source information to create files because EGL does not require that the files an application accesses be externally described. EGL relies on the record definition, which is built into the *PGM object, for the structure of a record. However, using the DDS information guarantees agreement between the application's view of the record structure and the record data stored on the iSeries system.

Modifying the generated DDS information

Indexed and indexed alternate specification record organizations might require that you modify the corresponding DDS source member. Adding DDS keywords to the file- and record-level identifiers in the DDS source member is the minimum modification necessary. Table 3 shows the DDS keywords and the conditions under which they are required.

You can add other DDS keywords to optimize record retrieval and simplify application logic. For example, logical files can be used to select a subset of physical file records. You can also to build your own DDS source member, based on your knowledge of the EGL record definitions in the application. In this case, individual field names and field lengths in the DDS source need not match those of the EGL record definition. However, the record length and key field length of the EGL record definition and the DDS source must be equal.

Table 3. Conditions for using DDS keywords

DDS keyword	Condition
PFILE(*pfname*)	When using the DDS information to create a logical file. *pfname* identifies the physical file on which the logical file is based. PFILE is a record-level keyword.
UNIQUE	When the application tests for the unique or duplicate record I/O error conditions. UNIQUE is a file-level keyword.

Restrictions on logical files

EGL supports simple logical files that use only one record format. The DDS source information specifies only one file on the PFILE keyword.

Changing DDS member types

EGL creates DDS source members without specifying a member type. To assist you in modifying the DDS source information, change the member type to one of the following:

- PF for a DDS source member describing a physical file
- LF for a DDS source member describing a logical file

Changing the member type to PF or LF enables the Source Entry Utility (SEU) prompting to help you to modify the DDS source member.

Application run-time considerations

The following sections describe the considerations to keep in mind during application run time.

Starting and ending commitment control cycles

To use iSeries Commitment Control Services for single-system iSeries applications, you must explicitly start and end a commitment control cycle using the start commitment control (STRCMTCTL) command to start the commitment control and the end commitment control (ENDCMTCTL) command to end the commitment control. EGL Server for iSeries does not implicitly start or end commitment control cycles for single-system iSeries applications. However, DB2® implicitly starts commitment control automatically for applications that use SQL I/O statements. After commitment control is started for the job, both native database I/O and SQL I/O can use the common commitment control that iSeries provides.

For EGL client/server applications and Web applications, commitment control is started by the run-time CL for the application.

If necessary, the commitment control for an SQL application can be changed by modifying the FDAPREP REXX program and is further controlled by a user-defined SYMPARM during generation.

If no commitment control cycle is active and the application attempts to open a file requiring commitment control, the application ends with an error condition. Messages in the job log explain the exact nature of the error. The application ends abnormally under these conditions because it might attempt to explicitly commit changes to a file, but that is possible only with an active commitment control cycle.

Chapter 2. Reviewing and preparing the generated output

This chapter provides an overview of the generation output files and describes how EGL prepares the output files before running your applications.

The format and content of the output files are in COBOL, control language (CL) source, and structured binary streams. Each output file is described in Table 4.

Outputs of generation

After you generate an application, a number of objects must be transferred to the iSeries host system as members in various iSeries physical files. On the iSeries host system, these members must be prepared before the application can be run.

Table 4 provides information about the types of files produced by generation, including the following:

- Type of object produced
- Physical file name where the object is written as a member
- How the member name of the object is derived
- Whether production is controlled by a generation option
- Whether the object can be modified after generation is performed

Refer to the *EGL Reference Guide* for more information on controlling and modifying generation and preparation of iSeries objects.

Table 4. Objects transferred to an iSeries host by the EGL preparation utility

File type	Physical file name	PF member name and generated file name	EGL build descriptor option	Modifiable
Objects generated for applications				
Application ILE COBOL program	QVGNCBLS	Application name *applname*.cbl	None	No
Run-time CL	QVGNCLS	Application name *applname*.clr	None	Yes
Objects generated for tables				
Table Binary Image	QVGNTAB	Table name *tablname*.tab	genDataTables	No
Objects generated for form groups				
Print services program (See note 3)	QVGNCBLS	Form group name *formgname*.cbl	genFormGroup, genHelpFormGroup	No
Form group module (See note 4)	QVGNMAPG	Form group name *formgnameFM*.fmt	genFormGroup, genHelpFormGroup	No
Objects generated for all member types (applications, tables, form groups)				
Generation variables file	QVGNEVF	*mbrname*.evf	None	No
Objects generated for message tables				
Message file	QVGNMSGS	Member specified when generation was requested *tablename*.msg	genDataTables	Yes
Build plan	not applicable	*applName*.BuildPlan.xml	buildPlan	No

Table 4. Objects transferred to an iSeries host by the EGL preparation utility (continued)

File type	Physical file name	PF member name and generated file name	EGL build descriptor option	Modifiable
Objects generated for file creation				
Data definition specification (DDS)	QVGNDDSS	File name as specified in EGL record definitions *filename*.dds	genDDSFile	Yes

Notes:

1. The generator produces ILE COBOL for the iSeries environments.
2. Generated application, table, and form group objects are environment dependent. All objects are generated for one environment and cannot be used in another environment.
3. This object is produced only if the form group contains print maps.
4. This object is produced only if the form group contains terminal maps.

Objects generated for programs

The follow sections describes the objects generated for applications.

Application ILE COBOL program

The generated application is an ILE COBOL program that contains the following:

- Application control logic
- Logic for application processes, statement groups, and I/O operations
- Data for both the application and application control

Run-time CL

The run-time CL sets commitment control and adds libraries to the iSeries library list when an application runs. The CL is generated from the templates efk24ebc.tbl and efk24eec.tpl, which can be customized.

The name of the run-time CL is as follows:

applname.clr

applname
Name of the application.

Data definition specification (DDS)

The generator produces iSeries data definition specifications (DDS) to create instances of iSeries physical and logical files that the application uses. The DDS produced is the result of the *indexed*, *relative*, *serial*, and *alternate specification* record types used within the application member being generated. The build descriptor option **createDDS** enables the production of the DDS output type. The command file uploads the DDS files to the host system, but the command file does not manage processing beyond that point.

Objects generated for data tables

The following section describes the objects generated for data tables.

DataTable binary image file

The dataTable binary image file contains the run-time EGL dataTable member contents as defined by EGL. The dataTable contents are already converted to the code page of the target run-time environment. The dataTable contents are formatted to an application defined structure (possibly containing various data types) and the contents are treated as binary data. You might not be able to view the contents outside the scope of EGL and utilities.

The build descriptor option genDataTables enables the production of the table binary image files.

Message definitions

For message tables, the generator produces a file containing the raw message definitions. This file is processed by the preparation script file, FDAPREP, to create an iSeries native message file object (*MSGF type).

The build descriptor option genDataTables enables the production of the message file, which is the iSeries implementation of the message table. Execution of the build plan uploads the message file and invokes the preparation script to generate the message object on iSeries.

Objects generated for form groups

The following section describes the objects generated for form groups.

Form group format module

The form group format module is a generated structure that describes the form layout for text forms in the form group. The generator builds the structure as a binary image file converted to the code page of the target system. This object is produced when you specify the genFormGroup or genHelpFormGroup build descriptor options and when the application has defined text forms in the form group.

Understanding preparation

This section describes the preparation process for a generated application to run in the iSeries environment. The preparation process is significantly different from that used in VisualAge® Generator.

The preparation is accomplished by using the iSeries Remote Build Server, which is a component of the EGL run time. For details, see “Starting the iSeries Remote Build Server” on page 10.

When an application is generated, an xml file, called the Build Plan, is created in the generation directory. The build plan is launched to prepare the application on iSeries, using a Java program called the build plan launcher.

When the application is generated with the build descriptor option **prep** set to *yes*, the build plan is launched automatically at the end of the generation. Otherwise the build plan can be launched later manually by following the process described later in this document.

The Build Plan Launcher uses the build plan and communicates with the Build Server to accomplish the preparation. The build plan contains all the information necessary to transfer the applicable generated files to iSeries and to build (compile and bind) the application.

A key component of the preparation is the Preparation Script. This is a Rexx Script, FDAPREP, which is installed as part of the runtime code and is described in “iSeries preparation script file FDAPREP” on page 2.

Starting the iSeries Remote Build Server

The Remote Build Server is a program, named CCUBLDS, that runs as a job on the iSeries. It listens on a tcp/ip port. Once started, it runs continuously until the job is canceled. Following is an example of the command to start the build server job:

```
SBMJOB CMD(CALL PGM(*LIBL/CCUBLDS) PARM('-p' '2600')) JOB(CCUBLDS) JOBQ(QSYS/QSYSNOMAX)
```

Here the server port is 2600, but any available port number can be used. The build server must be invoked by an administrator userid that is authorized to access user profiles.

Verifying the iSeries Remote Build Server

After the build server has started, you should verify that it is running properly. At the Windows workstation where the prep step will be run, do as follows:

1. In the plugin named *com.ibm.etools.egl.distributedbuild*, locate the directory containing the program ccubldc.exe. Add this directory to the PATH environment variable.
2. Execute the following command form the command line:

   ccubldc -h *host@port* -au *userId* -ap *password* -b id -r 37 -k 1252

   host
   IP address of the iSeries host machine

   port
   Port number of the build server

   userId
   Userid that the prep client will use

   password
   Login password for the userid on the iSeries host

Following is an example of the expected response:

```
05/03/09 14:58:56 (c) Copyright, IBM Corp. 2001 Copyright (c) 2002 Rational Software Corporation
05/03/09 14:58:57 *** Success ***
05/03/09 14:58:57
Command: id
****************** Build Script Output Follows *****************
uid=926(USERID) gid=102(GROUPID) groups=102(GROUPID)
****************** End Of Build Script Output ******************
05/03/09 14:58:58 *-------------------------------------------------------
```

Launching the build plan manually

You may wish to create a build plan and to invoke that plan at a later time. This case might occur, for example, if a network failure prevents you from preparing code on a remote machine at generation time.

To launch a build plan in this case, complete the following steps:

1. Make sure that eglbatchgen.jar is in your Java classpath, as happens automatically on the machine where you install EGL. The jar file is in the WebSphere® Studio installation directory (like c:\myStudio), in the following subdirectory:

   wstools\eclipse\plugins\
   com.ibm.etools.egl.batchgeneration_*version*

version
The installed version of the plugin; for example, 5.1.2

2. Similarly, make sure that your PATH variable includes the following subdirectory of the WebSphere Studio installation directory:

   ```
   wstools\eclipse\plugins\
   com.ibm.etools.egl.distributedbuild_version
   ```

   version
   The installed version of the plugin; for example, 5.1.2

3. From a command line, enter the following command:

   ```
   java com.ibm.etools.egl.distributedbuild.BuildPlanLauncher bp
   ```

   bp The fully qualified path of the build plan file. For details on the name of the generated file, see the EGL help topic *Generated output (reference)*.

Preparing a DB2 application

When you specify an SQL table name in EGL, you can enter the table name on the **Table Specification** window using either the SQL naming convention of collection.tablename or the iSeries SYSTEM naming convention of collection/tablename. Whichever format you use as the standard to qualify table names, tailor the OPTION parameter on the FDAPREP script to be *SQL (collection.tablename) or *SYS (collection/tablename). The default naming convention is *SQL.

When you tailor the FDAPREP script, ensure that the *APOSTSQL and *QUOTE values are part of the OPTION parameter.

Chapter 3. Running a generated application in iSeries

This chapter describes the information required to run EGL applications on an iSeries system.

Making EGL Server for iSeries, COBOL, and generated modules available

The setup tasks that are required to run EGL applications on iSeries are simpler than with other run-time environments because no run-time setup control programs are produced, other than the application program itself.

EGL Server for iSeries and the generated COBOL applications use the run-time job library list (*LIBL) to resolve all named object references

The library list must be set up by the application programmer, system administrator, or EGL application developer before starting the application. To aid in the library setup, Table 5 on page 14 lists the names and types of objects that EGL might use while running in the iSeries environment. EGL searches for these objects dynamically when the application is running by scanning the libraries named in the library list. It is recommended that the installation library, QEGL, be added to the library list of the end user when running EGL applications.

EGL uses the first object it finds that matches the target name in the libraries named in the library list. This first-found object is used in all cases of object resolution except for objects of *FILE type. In this case, EGL uses the first member it finds that matches the target name, after the first member have been qualified with the correct file name. Multiple files with the same name might exist in the libraries named in the library list. EGL checks each library file until it finds the first instance of the member name.

Table 5. Names and types of objects used by EGL at run time

Object and library name	Type	Description
QVGN* QEGL	*PGM *SRVPGM	EGL Server for iSeries program and service program objects.
OVGNMSGF QEGL	*MSGF	EGL Server for iSeries product message file.
QVGNMAPG userlib	*FILE	Members of this file contain the generated applications 5250 form groups. Members are named for the form group it contains.
QVGNTAB userlib	*FILE	Members of this file contain the generated applications tables. Members are named for the table it contains.
QVGNPRNT QEGL	*FILE	This is the standard printer device file for application use of the Printer file. Usually, all jobs on the system share one of these objects.
QVGMAP QEGL	*FILE	This standard display device file is used for interactive applications when they display maps. Usually, all jobs on the system share one of these objects.
mmmmnls userlib	*MSGF	A specific application's message table, where *mmmm* is the message table prefix as defined to the application, and *nls* is the value of the build descriptor option targetNLS when the application was generated.
calltarg userlib	*PGM	Any target of the call or transfer statements coded within an application.
filetarg userlib	*FILE	Any file named on record definitions used by EGL process options within an application.

Note: The designated library *userlib* in Table 5 indicates that the object is in a library named by the application developer at the time the application was developed.

The two exceptions to using the library list to resolve object references by running applications are as follows:

- When EGL application tables and map groups reside in the iSeries IFS file system for improved run-time performance.
- When either:
  - The object (table or view) was explicitly qualified when an SQL record was defined during development
  - The object was implicitly qualified when the application using the record was compiled.

In either case, SQL object resolution is independent of the library list.

Establishing a library list for a job

You can establish a library list for a job in several ways, but each method involves using an iSeries system command. You can also mix the methods. The initial iSeries library list is contained in the job description referenced by the user profile. For more information about the following iSeries system commands, enter the command at a command line entry on iSeries and request command prompting.

ADDLIBLE
Adds a library list entry

CHGCURLIB
Changes the current library

CHGSYSLIBL
Changes the system library list

Running EGL applications under iSeries

To run a EGL application in the iSeries environment, call the *PGM application object just as you would any other *PGM object on iSeries. You can run EGL applications from menus, commands, command lines, or interlanguage program calls. This applies to main and called EGL applications. Examples of how to run a EGL application from an iSeries command line follow:

- To call an application without the use of arguments, use the following format:

  ```
  CALL applname
  ```

- To call a CALLED application that expects a parameter declared as CHAR(16), use the following format:

  ```
  CALL applname ('char arg literal')
  ```

- To call a CALLED application that expects two parameters declared as CHAR (15) and DECIMAL (15,5), use the following format:

  ```
  CALL applname ('char arg literal' 1234)
  ```

EGL run unit concept

EGL applications operate in a run unit much like that of ILE COBOL. EGL's run unit can be considered a subset of the ILE COBOL run unit because the COBOL run unit might exist before and persist longer that EGL's run unit.

EGL's run unit is bounded by the first EGL application on the iSeries program call stack for a specific job. Run units are scoped in a single job. As long as a EGL application is on the program call stack, an EGL run unit is active. Only one EGL run unit can be active in a job at any time. This is the most obvious difference between EGL run units and ILE COBOL run units.

Main or called programs applications can initiate a EGL run unit. EGL man programs can exist in a run unit only if it is the initiating application. Main programs cannot be called from an application that initiates a EGL run unit, even if it is called from a non-EGL program while a EGL run unit was active.

Using activation groups with run units

EGL run units normally correlate on a one-to-one basis with ILE COBOL run units and ILE activation groups. When a EGL application initiates or begins a run unit, a named activation group is also initiated. Using a named activation group ensures that all EGL applications that run in the job share the same resources in terms of ILE resource management.

If your application system consists of non-EGL programs as well as EGL applications, you can add your non-EGL programs to the named activation group or use a different activation group. Sharing commitment control logical units of work and sharing database file Open Data Paths are important aspects to consider when making your decision. To share either ILE resources, using the same

activation group achieves the above result. Conversely, using different activation groups keeps the ILE resources isolated in terms of EGL application use and non-program use.

The ILE activation group name is established when EGL applications are in the preparation phase of application development. EGL preparation templates, which create OS/400® program objects, name the activation group in the iSeries command template for the CRTPGM command. The keyword is ACTGRP and the default is ACTGRP(QVGN).

Chapter 4. Moving prepared applications to other iSeries systems

You can move a prepared application from one iSeries system to another. For example, you might have the compiler on one host development machine but want to run the application on several production machines.

The iSeries and EGL Server for iSeries products on the production machine must be at the same maintenance level as, or at a higher level than, the development machine.

Moving prepared applications to another iSeries system

After an iSeries EGL application is prepared, you can distribute that application to other architecturally similar iSeries systems using the following procedures:

1. Use the CRTSAVF FILE(library/filename) command to create a save file.
2. Use the SAVOBJ or SAVLIB command to save all the application parts you want to move to another system in the save file.
3. Transfer the save file to other iSeries systems by using a communications network or by using physical media such as a tape.
4. Use the RSTOBJ (Restore Object) or RSTLIB (Restore Library) command to store the objects from the save file.

Moving workstation code that is part of a EGL application to an iSeries system

Moving workstation code that is part of a EGL client/server application to an OS/400 system can be done using diskettes or shared folders.

To save the workstation code using the iSeries, do the following:

1. Copy the code to a shared folder along with EGL GUI run-time support
2. Create a save file using the iSeries command CRTSAVF
3. Issue the iSeries command SAVDLO
4. Transfer the save file to another iSeries system by using a communication network or by using a physical media such as a tape
5. Erase the existing shared folder, if one exists
6. Restore the shared folder using the iSeries command (RSTDLO) on the production iSeries

Maintaining backup copies of production libraries

Maintaining backup copies of your production libraries can be accomplished by creating a save file, then issue the SAVLIB command. You can copy the save file to tape.

Chapter 5. Diagnosing problems during run time

This chapter contains information to aid in diagnosing problems that you might encounter while running your EGL applications.

EGL applications are implemented on iSeries just like other third-generation language (3GL) applications or programs. This is important to remember, should you or the IBM® Support Center need to collect extra ordinary problem diagnostic information in the course of investigating a run-time error.

iSeries standard diagnostic commands are available to you when diagnosing problems with a EGL application. These include such commands as the following:

- ADDTRC (Add Trace Statement)
- STRJOBTRC (Start Job Trace)
- ENDJOBTRC (End Job Trace)
- PRTJOBTRC (Print Job Trace Data)
- STRDBG (Start Debug)
- ENDDBG (End Debug)

In most error diagnostic cases, you need to have the following information available when you contact the IBM Support Center:

- The run-time job log, which recorded all messages including second-level text.

  To ensure that second-level text is included, change the job before starting the failed scenario using the command CHGJOB LOG(4 00 *SECLVL). When the job ends, the job log is spooled to the assigned output queue. Usually, the most important information in the job log is the escape messages that initiates the abnormal condition, which caused the EGL application to end. The program sending the message, the program receiving the message, and the instructions being sent to the program are the key pieces of diagnostic information. The other messages are also important. Be sure to inspect and report the entire job log of information.
- The ILE COBOL compiler listing, which includes the following:
  - EGL annotated statements and code-production audit comments (use build descriptor option commentLevel)
  - ILE COBOL source statements (use compiler option OPTION(*SOURCE))
- Any additional spooled files that might have been created as a result of the job ending abnormally, such as dumps or display job snapshots. You can locate all spooled file output from a job by using the WRKJOB command to work with the job. Then you can select the option from the job menu to display spooled output.

iSeries First Failure Data Capture component

EGL applications are linked into the iSeries First Failure Data Capture component when function checks (abends) occur via the iSeries System Programming Interface (SPI) called Report Software Error (QpdReportSoftwareError). EGL uses this function to provide run unit data dumps that can be sent to the IBM Support Center.

EGL also uses the report software error SPI function when illogical conditions are detected during run time that might lead to a function check (abend). In these

cases, a unique signature associated with the error condition is provided to the system service. The system service can use the unique signature to scan a service database for the same signature and possible PTFs that can be applied. This expedites the process of problem resolution in some cases.

When job log messages indicate that the iSeries system problem log has been updated, use the iSeries command DSPPRB (Display Problem) to view a list of the most recent problems captured on the system; then select the option to display the problem associated with QVGNHS. QVGNHS is the service program that constitutes the majority of EGL Server for iSeries. A procedure of this service program issues the SPI function to record the problem. Upon displaying the problem, the menu selections and function keys enable you to display further information, such as spooled files, problem history files, and APAR libraries. The spooled files contain dump data and a copy of the job attributes at the time of the dump.

For more information on using the system problem log in an automated manner, refer to the SystemView® System Manager product documentation.

Chapter 6. messages

Message text can contain one or more inserts. When the message is displayed, an insert is used to fill in names, constants, return codes, and other information. For example, the format of the message insert might look like this: **&1**

The server messages can be viewed online on iSeries by using the Work with Message Description command WRKMSGD MSGF(QEGL/OVGNMSGF).

EGL applications use the standard iSeries message handling functions to communicate with calling programs. Diagnostic information is automatically logged by the job log during run time.

Escape messages

These messages are sent by EGL applications to the program queue of the calling program as iSeries ESCAPE type messages. The calling program must monitor these messages to avoid an iSeries function check.

GEN9001 EGL Server MAIN shell cannot invoke the target program %1.

Explanation: Diagnostic messages preceding this message in the job log explain the nature of the error. In most cases, the application or system programmer will need to adjust your application system to correct the problem.

User Response: Either print the job log or record the messages along with the following:

- - The from program name.
- - The to program name.
- - The instruction numbers.

You can view or print the job log with the DSPJOBLOG command. If no diagnostic messages precede this message, ensure that your job logged all messages by checking the 'Message Logging' or LOG value of your job definition or job description, depending on whether the job is interactive or batch. For interactive jobs, command DSPJOB OPTION(*DFNA) will display the 'Message Logging' value.

Contact your application or system programmer with the information you gathered.

GEN9002 EGL Server encountered an application error which caused the run unit to end.

Explanation: Diagnostic messages preceding this message in the job log explain the nature of the error. In most cases, the application or system programmer will need to adjust your application system to correct the problem.

User Response: Either print the job log or record the messages along with the following:

- - The from program name.
- - The to program name.
- - The instruction numbers.

You can view or print the job log with the DSPJOBLOG command. If no diagnostic messages precede this message, ensure that your job logged all messages by checking the 'Message Logging' or LOG value of your job definition or job description, depending on whether the job is interactive or batch. For interactive jobs, command DSPJOB OPTION(*DFNA) will display the 'Message Logging' value.

Contact your application or system programmer with the information you gathered.

Diagnostic and informational messages

The following messages are sent as DIAGNOSTIC or INFORMATIONAL type messages to the program queue of the calling program. These messages are automatically posted in the job log. Programs that call EGL applications cannot monitor the activities of these messages. Use the WRKJOB(Work with JOB) command to view the job log. The Message Logging job attribute might filter some or all of these messages in some way. To ensure you get all messaged posted in the

job log, use a message logging value of LOG(4 00 *SECLVL) in your iSeries jobs. See the iSeries commands WRKJOB and CHGJOB (Change Job) for more information.

GEN0002 A new level of EGL Server for iSeries is required for program %1

Explanation: The generated COBOL program %1, attempting to run, is not compatible with the installed version of EGL Server for iSeries.

User Response: Contact the system administrator. Version %2 of EGL Server for iSeries should be installed.

GEN0005 Date entered is not valid for defined date format %1

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 070494.

User Response: Enter the date in the format %1.

GEN0009 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 special function word sysVar.handleOverflow is 1, which specifies that the application should end if this overflow condition occurs.

User Response: Have the application developer do one of the following:

- Increase the number of nondecimal digits in the target data item
- Define the application logic to handle the overflow condition using the special function words sysVar.handleOverflow and sysVar.overflowIndicator.

GEN0014 REPLACE attempted without preceding UPDATE option on %1.

Explanation: A replace option was attempted against a record that has not been successfully read by a get or open statement. The read for update might have been lost as the result of a commit or rollback.

This error also occurs if a replace is associated with a specific get or open statement, but that get or open statement was not the one used to select the record.

User Response: Have the application developer run this application with the test facility, tracing for process and statement group flow, to determine the application logic error.

GEN0021 An error occurred in application %1 on EGL statement number %2.

Explanation: The actual error identifying the problem is explained in messages following this message in the job log.

User Response:

GEN0022 Map group format member %1 could not be loaded

Explanation: The form group format member is a generated binary file that contains attributes that describe the format and constant fields for character/text based maps in a form group. Form group format members are stored as members of file %2 and are expected to be found by searching the job's library list.

Diagnostic messages preceding this message in the job log explain the nature of the error.

User Response: Contact your application or system programmer and report the sequence of messages including and preceding this message.

GEN0023 The table %1 could not be loaded.

Explanation: A table is a generated binary file that contains application data. Tables are stored as members of file %2 and are expected to be found by searching the job's library list.

Diagnostic messages preceding this message in the job log explain the nature of the error.

User Response: Contact your application or system programmer, and report the sequence of messages including and preceding this message.

GEN0024 EGL conversion table %1 could not be found

Explanation: Either the name specified on the sysLib.convert call was not a member of the QEGL/QEGLSCTB file or the member that was found is not a conversion table.

User Response: Have the application developer verify that the correct conversion table name was specified in the sysLib.convert call. If the table name was not correct, then change the EGL application and regenerate it. If the table name is correct, verify that the correct conversion table was installed. The conversion table is a member in the file QEGLSCTB in QEGL library.

GEN0026 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 the sysVar.handleOverflow switch is set to either 0 or 1, the application ends.

User Response: Have the application developer correct the application logic either to avoid the error or to handle the error using the special function words sysVar.handleOverflow and sysVar.overflowIndicator.

GEN0027 The data on 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.

User Response: Have the application developer change the application to make sure that the source operand contains valid numeric data.

GEN0031 A call to program %1 failed.

Explanation: Called programs are expected to be found by searching the job's library list.

Diagnostic messages preceding this message in the job log will explain the nature of the error.

User Response: Contact your application or system programmer, and report the sequence of messages including and preceding this message.

GEN0033 Call to function %1 returned exception code %2

Explanation: An exception code was returned on a call to the specified function, indicating that one of the arguments passed to the function was invalid. Refer to the EGL reference guide for further details.

The run unit ends.

User Response: The developer should fix the program so that it does not pass invalid arguments to the function.

GEN0034 Application %1 was defined as a MAIN application and cannot be called.

Explanation: The specified application was defined as either a main textUI program or as a main basic program. It cannot be called by another program.

User Response: If using the call statement to invoke application %1 is valid, have the application developer define %1 as a called application. If application %1 must remain a main application, then have the application developer use transfer statements to invoke it from another main application.

GEN0035 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 application continues.

GEN0036 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 application continues.

GEN0037 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 application continues.

GEN0038 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 application continues.

GEN0039 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 application continues.

GEN0040 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 validation 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 application continues.

GEN0041 A message file prefix was not specified for an application: EZEMNO = %1, NLS code = %2.

Explanation: A user message was requested, but a user message table prefix was not defined for the application.

User Response: Have the application developer do one of the following:

- Add the message table prefix to the application specifications and then generate the application again.
- Change the application to not request the message.

GEN0045 Error retrieving user application message, EZEMNO = %1, NLS code %2. See previous messages.

Explanation: A user message was requested. The previous message in the job log explains the reason for the error.

User Response: Most problems occur because the message file or the specific message cannot be found or access to the message file is not authorized. If the application can not find the message file and you know the library name that contains the message file, you can add the library to your library list (ADDLIBLE command). For other problems, contact your system or application programmer.

GEN0046 Call to printer mapping services program %1 failed.

Explanation: Printer mapping services programs are expected to be found by searching the job's library list.

Diagnostic messages preceding this message in the job log explain the nature of the error.

User Response: Contact your application or system programmer and report the sequence of messages including and preceding this message.

GEN0050 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.

GEN0051 Map %1 was not found in map group %2.

Explanation: The specified form name is not in the form group.

User Response: Have the application developer generate the form group and the application again.

GEN0057 DELETE attempted without preceding UPDATE on record %1.

Explanation: A delete statement ran for a record that was not successfully read by an open or get statement. The read for update might have been lost as the result of an commit or rollback.

User Response: Have the application developer run this application in the EGL debugger to determine the application logic error.

GEN0073 SQL error, command = %1, SQL code = %2

Explanation: The SQL database manager returned an error code for an SQL statement. Application 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:

- - No error routine is specified for the process
- - The SQLCODE indicated a hard (terminating) error and the special function word sysVar.handleHardIOErrors was set to 0, indicating that the application should end on SQL error conditions.

This message is followed by the actual DB2 message describing the SQL error code.

User Response: Have the application developer or system programmer determine the cause of the problem from the SQL code and the SQL error information. Either correct the application or the database definition. Refer to the appropriate Database Manager messages and codes manual for information on the SQL code and SQL error information.

GEN0074 SQL error message: %1

Explanation: This message accompanies message GEN0073 when an SQL error occurs. It displays the relational database manager description of the error and is repeated as many times as necessary to display the complete description.

User Response: Use the information from this

message and GEN0073 to correct the error.

GEN0076 The data on character to hexadecimal move or compare is not valid.

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 an application program error.

User Response: Have the application developer change the application to make sure that the character data item contains valid data when the character to hexadecimal move compare operation occurs. To do so, the application developer can use the hexadecimal map edit characteristic to make sure that input from a variable field contains valid characters.

GEN0080 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.

GEN0086 %1 - No active SETINQ, SETUPD, or UPDATE

Explanation: A get, get next, replace, or delete statement cannot run because a required get or open statement was not run previously in the same application. All rows selected for scanning or updating are released when an application returns to a calling application.

User Response: Have the application developer modify the application.

GEN0093 An error occurred in application %1, process or group %2.

Explanation: An error occurred in the specified application. Other information about the error is given in the messages that follow this message.

User Response: Refer to the error messages following this message to determine the cause of the error.

GEN0096 A mixed data operand is not valid

Explanation: An operand in a move statement involving an item of type MBCHAR contains an invalid mixture of double-byte and single-byte data.

User Response: Have the application developer verify that all operands in the move statement contain valid data.

GEN0109 FIRST MAP must be map %1, not map %2, for application %3.

Explanation: The initial form expected by this application is not the form identified in the message. This error occurs when the application starts.

User Response: Record what function you were using before the error occurred, and have the application developer correct the arguments used to start program %3.

GEN0111 Length of FIRST MAP %1 is not valid.

Explanation: The length of the form %1 received by an application is not the length defined for the form in application %2.

User Response: Have the application developer generate both the application receiving the form and the application that shows the form.

GEN0119 Applications %1 and %2 are not compatible.

Explanation: An application started by a transfer or call statement is not compatible with the initial application in the transaction or job because the application was generated for a different environment.

User Response: Have the application developer regenerate one or both applications so that the target environments for the applications are the same.

GEN0127 A requested function is not supported for map %1, map group %2.

Explanation: An application requested an action that is not supported for the specified form and form group. The form group was modified after it was generated and before the application was generated. Some aspects of the form group when the application was generated (for example, use of help maps or use of the msgField property) were not in the form group when the form group was generated.

User Response: Have the application developer generate the form group and the application again.

GEN0137 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 GEN0073, that provide additional information about the error.

The run unit ends. Messages are logged.

User Response: Determine the cause of the problem from the SQL code and the SQL error information in related message GEN0074, and correct the database definition.

GEN0184 Application %1 and mapping services program %2 are not compatible.

Explanation: The specified application and mapping services program are generated for different systems.

User Response: Have the application developer generate the mapping services program for the same environment as the application.

GEN0185 Length of %1 for record %2 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 is longer than the maximum length defined for the record.
- - The record data ends in the middle of a numeric field.
- - The record data ends in the middle of a DBCHAR character.
- - The record data ends in the middle of a SO/SI string.

User Response: Have the application developer modify the application to set the record length so that it ends on a valid field boundary.

GEN0186 A mixed string in a conversion operation is not valid

Explanation: Conversion of a mixed field from EBCDIC to ASCII or from ASCII to EBCDIC cannot be performed because the double-byte data value is not valid.

User Response: Have the application developer modify the application to make sure that the records to be converted contain valid data.

GEN0187 Conversion table %1 does not support DBCS character conversion.

Explanation: Conversion of a mixed or DBCHAR field from ASCII to EBCDIC or EBCDIC to ASCII cannot be performed because the specified conversion table does not include conversion tables for double-byte characters.

User Response: Have the application developer modify the application to specify a conversion table that contains the double-byte conversion tables valid for the DBCHAR or MBCHAR data being converted.

GEN0188 Conversion Error. Function: %1, Return Code: %2, Table: %3

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 DBCHAR conversion where the source data is in ASCII format, the source data was created under a different DBCHAR code page than the code page that is currently in effect on the system.

User Response: Correct the cause of the error.

GEN0191 Application %1, generation date %2, time %3.

Explanation: An error in application %1 has occurred. Diagnostic messages in the job log explain the nature of the error. Changes to individually generated components of the application may have caused the error.

User Response: Have the application developer verify the generation date and time of the application with that of other generated components.

GEN0192 Mapping services program %1, generation date %2, time %3.

Explanation: An error in mapping services program %1 has occurred. Diagnostic messages in the job log explain the nature of the error. Changes to individually generated components of the application may have caused the error.

User Response: Have the application developer verify the generation date and time of the mapping services program with that of other generated components in the application.

GEN0195 Map format member %1, generation date %2, time %3.

Explanation: An error in map format member %1 has occurred. Diagnostic messages in the job log explain the nature of the error. Changes to individually generated components of the application may have caused the error.

User Response: Have the application developer verify the generation date and time of the map format member with that of other generated components in the application.

GEN0210 EGL Server number %1 is not valid.

Explanation: An attempt was made to start an EGL routine that does not exist or that is not valid.

User Response: Have the application developer generate and compile the application again to ensure the generated COBOL code has not been modified. Afterward, run the refreshed application. If the problem persists, have the system administrator do all of the following:

- 1 - Record the service number from this message.
- 2 - Print the job log.
- 3 - Record the scenario under which this message occurs.
- 4 - Obtain the COBOL listing and source for the failing application.
- 5 - Use your electronic link with IBM Service (for example, IBMLINK) if one is available, or contact the IBM Support Center.

GEN0232 Map %1 in map group %2 is not defined or is not supported.

Explanation: The specified form does not exist or is not defined for the type of device being used.

User Response: Have the application developer either define the form for your device type or select the device for the form. Generate the form group again.

GEN0233 %1 error on file %2, EZERT8 = %3.

Explanation: An I/O operation failed for the specified file. This message specifies the COBOL verb performed and the EGL file name associated with the operation.

sysVar.errorCode contains either the COBOL status key value or EGL file return code.

User Response: Use the appropriate COBOL publication or the EGL reference guide to diagnose the error, and take the recommended corrective action.

GEN0260 %1 bytes of UI record won't fit in %2 byte buffer.

Explanation: The program issued a statement for presenting a UI record. There was not enough room in the communications buffer for the record. The buffer needs space for the record plus any message information written using function sysLib.setError.

User Response: Modify the program to reduce the size of the user interface record or write fewer or smaller error messages.

GEN0261 EZEUIERR message information and inserts won't fit in %1 byte buffer.

Explanation: The program issued one or more calls to the system function sysLib.setError to write messages associated with a UI record. The information associated with the last message written will not fit in the buffer used by the program for communicating with user.

User Response: Modify the program to write fewer or smaller error messages.

GEN0262 Web transaction program and user interface record bean %1 are incompatible.

Explanation: An action program was started with information from a UI record bean that isn't known to the program or whose definition is not compatible with the UI record declaration with which the program was generated.

User Response: Insure that the specified bean is defined as the inputPageRecord for the program. Regenerate the program and the Java Beans from the same user interface record declaration.

GEN0263 Number of occurs value %1 is out of range for record array at offset %2

Explanation: An action program could not write a UI record because the ″number of elements″ value set by the program for an array was less than 0 or greater than the maximum number of elements defined for the array.

User Response: Correct the program logic so that it sets the number of elements to a value within the allowed range.

GEN0264 Input data entered by the user doesn't fit in user interface record.

Explanation: An action program received input data from the Web server that doesn't fit in the UI record. The transaction program and the Java Bean associated with the UI record record may have been generated at different times with incompatible UI record declarations.

User Response: Regenerate the program and the Java Beans from the same definitions. Contact your IBM representative if this doesn't correct the problem.

GEN0265 Segmented converse is not supported within current function stack.

Explanation: The program issued a converse statement with sysVar.segementedMode set to 1 (segmented converse) and at least one of the functions in the current function stack uses parameters or local items or records. The generated program is not able to save parameters or local storage data over a segmented converse.

User Response: Modify the program so that the converse statement is not used within a function that has parameters or local data.

GEN0266 MQ function %1, Completion Code %2, Reason Code %3.

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)

User Response: Please refer to the MQSeries Application Programming Reference for further information on MQSeries completion and reason codes.

GEN0267 Queue Manager Name %1.

Explanation: This is the name of the queue manager associated with the failing MQ function call listed in message GEN0266.

If the failing MQ function was MQOPEN, MQCLOSE, MQGET, or MQPUT, the name identifies 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.

User Response: Please refer to the MQSeries Application Programming Reference for further information on the MQSeries completion and reason code listed message GEN0266.

GEN0268 Queue Name %1.

Explanation: This is the name of the queue object associated with the failing MQ function call listed in message GEN0266.

User Response: Please refer to the MQSeries Application Programming Reference for further information on MQSeries completion and reason codes reported in message GEN0266.

GEN2001 The table %1 is not valid for application %2

Explanation: The reason code is %3. The explanations follow:

- 1 - The dataTable version is not compatible with the current level of IBM EGL Server and the running application.
- 2 - The dataTable was generated for an ASCII-based EGL runtime environment.
- 3 - The data itself is corrupted.
- 4 - The dataTable could not be opened.

User Response: Have the application developer replace dataTable %1 with a correctly generated version.

If the reason code indicates that the table data is corrupted, ensure that the table was transmitted to the host system as a binary image file.

If the reason code indicates the table was generated for an ASCII-based host system, ensure that the table is regenerated for the same target system as the application attempting to use it.

If the reason code indicates the table could not be opened see previous messages in the job log.

GEN2002 EGL Server does not support DBCS data type.

Explanation: EGL Server does not support the DBCHAR data type because COBOL does not support DBCHAR.

User Response: Have the application developer

change EGL DBCHAR primitive types to MBCHAR data types and regenerate the application.

GEN2004 Character conversion from CCSID %1 to %2 is not supported

Explanation: Character conversion is not supported between the two Coded Character Set IDs (CCSID) %1 and %2.

User Response: Have the application developer verify that the specified Coded Character Sets IDs (CCSID) are valid and that conversion between the two CCSIDS is supported. The EGL application may have to be regenerated.

GEN2005 Error %1 occurred when converting record %2.

Explanation: sysLib.convert encountered error code %1 during the call.

User Response: Have the application developer verify that the application logics record %2 with data that matches its definition. The EGL application then needs to be regenerated.

GEN2006 The map group %1 is not valid for application %2.

Explanation: The reason code is %3. The explanations follow:

- 1 - Reserved.
- 2 - Reserved.
- 3 - The form group data is corrupted.
- 4 - The form group could not be opened.

User Response: Have the application developer replace form group %1 with a correctly generated version.

If the reason code indicates that the form group data is corrupted, ensure that the form group was transmitted to the host system as a binary image file.

If the reason code indicates the form group could not be opened see previous messages in the job log.

GEN2007 Press Enter to continue.

Explanation:

User Response:

GEN7025 Error encountered allocating memory.

Explanation: An error was encountered while allocating memory. The system has run out of memory.

User Response: Make sure that you have enough memory on your system as specified in the Software/hardware requirements for the product. Stop the execution of some of your other applications on your system.

Do as follows:

1. Record the message number and the message text. (The error message includes the information on where the error occurred and the type of internal error.)
2. Record the situation in which this message occurs.
3. Contact your IBM representative.

GEN7030 The format of the data descriptor is incorrect. The hex value of the data descriptor in error is %1.

Explanation: The format of the data descriptor is incorrect. A header descriptor is found within the data descriptor.

User Response: Do as follows:

1. Record the message number and the message text. (The error message includes the information on where the error occurred and the type of internal error.)
2. Record the situation in which this message occurs.
3. Contact your IBM representative.

GEN7035 The format of the data descriptor is incorrect.

Explanation: The format of the data descriptor is incorrect. An End Of Description descriptor is not found.

User Response: Do as follows:

1. Record the message number and the message text. (The error message includes the information on where the error occurred and the type of internal error.)
2. Record the situation in which this message occurs.
3. Contact your IBM representative.

GEN7040 The format of the data descriptor is incorrect. An unknown data code %1 was found.

Explanation: The format of the data descriptor is incorrect. An unknown data code was found in the data description.

User Response: Do as follows:

1. Record the message number and the message text. (The error message includes the information on where the error occurred and the type of internal error.)
2. Record the situation in which this message occurs.
3. Contact your IBM representative.

GEN7055 The Conversion Descriptor structure is not valid.

Explanation: The Conversion Descriptor structure CMCVOD required by the conversion routine is incorrect.

User Response: Do as follows:

1. Record the message number and the message text. (The error message includes the information on where the error occurred and the type of internal error.)
2. Record the situation in which this message occurs.
3. Contact your IBM representative.

GEN7065 The data descriptor for parameter %1 is not valid.

Explanation: The data descriptor for the parameter is not valid.

User Response: Do as follows:

1. Record the message number and the message text. (The error message includes the information on where the error occurred and the type of internal error.)
2. Record the situation in which this message occurs.
3. Contact your IBM representative.

GEN9003 EGL Server encountered a critical internal processing error.

Explanation: A critical internal processing error was detected. This may include such things as corrupted run unit control blocks, an unexpected return code from an internal function, or illogical code path entry.

Diagnostic messages preceding this message in the job log explain the nature of the error. In most cases, the application or system programmer will need to adjust your application system to correct the problem.

User Response: Either print the job log or record the messages along with the following:

- - The from program name.
- - The to program name.
- - The instruction numbers.

You can view or print the job log with the DSPJOBLOG command. If no diagnostic messages precede this message, ensure that your job logged all messages by checking the 'Message Logging' or LOG value of your job definition or job description, depending on whether the job is interactive or batch. For interactive jobs, command DSPJOB OPTION(*DFNA) will display the 'Message Logging' value.

Contact your application or system programmer with the information you gathered.

GEN9004 EGL Server COBOL error handler was invoked to end the run unit.

Explanation: A function check has caused the run unit to end. A database rollback has been issued and heap storage released.

Diagnostic messages preceding this message in the job log explain the nature of the error. In most cases, the application or system programmer will need to adjust your application system to correct the problem.

User Response: Either print the job log or record the messages along with the following:

- - The from program name.
- - The to program name.
- - The instruction numbers.

You can view or print the job log with the DSPJOBLOG command. If no diagnostic messages precede this message, ensure that your job logged all messages by checking the 'Message Logging' or LOG value of your job definition or job description, depending on whether the job is interactive or batch. For interactive jobs, command DSPJOB OPTION(*DFNA) will display the 'Message Logging' value.

Contact your application or system programmer with the information you gathered.

Chapter 7. Notices

Note to U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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 106-0032, 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
Lab Director
IBM Canada Ltd. Laboratory
8200 Warden Avenue
Markham, Ontario, Canada L6G 1C7

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 information 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 measurements 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.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to 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 illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to

IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. 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:

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

- DB2
- IBM
- iSeries
- OS/400
- SystemView
- VisualAge
- WebSphere

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product or service names, may be trademarks or service marks of others

Program Number: 5724-D46

Printed in USA

SC31-6841-01