



Developing EGL for CICS

Contents

Developing EGL programs for the CICS environment 1

Understanding CICS terminology	1
File techniques in CICS programs	3
Using temporary storage	4
Using transient data queues	6
Using spool files in z/OS CICS	7
Using spool files in VSE CICS	9
Using VSAM files	13
Using recordName.resourceAssociation	14
Printing techniques in CICS	15
Using transient data queues for printer output	15

Using spool files for printer output on z/OS CICS	16
Using VSE/POWER files for printer output	17
Using converseVar.printerAssociation.	18
Debugging CICS programs	19
Setting the recovery unit of work	20
Using CICS functions from EGL programs	20
Communicating between multiple CICS transactions	24
Inter-transaction affinity considerations in a CICSplex	25

Index 29

Developing EGL programs for the CICS environment

There are specific design and development considerations when creating EGL programs for CICS® for z/OS and CICS for VSE.

You can perform the following tasks using EGL to develop CICS programs:

- Define programs for CICS.
- Test CICS program logic using the EGL debugger.
- Generate COBOL programs to run in the CICS environment.

The following sections contain additional information on developing programs for the CICS environment:

- Accessing multiple DB2 plans in z/OS CICS
- Developing segmented programs in EGL

Refer to the EGL documentation for the following information about CICS:

- Transferring control in CICS environments.
- CICS considerations in the *IBM Rational COBOL Runtime Guide for zSeries*.

Related information

“Understanding CICS terminology”

Familiarize yourself with terms that have special meanings in CICS.

“File techniques in CICS programs” on page 3

Defining CICS programs is much the same as defining programs for other environments. There are some file technique considerations you should note that are specific to CICS.

“Printing techniques in CICS” on page 15

CICS handles the **printer** file differently on different platforms.

“Debugging CICS programs” on page 19

Test CICS programs by first fixing logic errors in the EGL debugger, then generating the programs for your CICS test region.

“Setting the recovery unit of work” on page 20

The **sysLib.commit()** system function notifies EGL that the current recovery unit of work is complete and a new unit of work is to be started.

“Using CICS functions from EGL programs” on page 20

You can use CICS functions in EGL programs.

“Communicating between multiple CICS transactions” on page 24

“Inter-transaction affinity considerations in a CICSplex” on page 25

Understanding CICS terminology

Familiarize yourself with terms that have special meanings in CICS.

Transaction

A unit of processing, consisting of one or more programs.

Task The processing of a transaction for a program user.

Conversational

The CICS term for running a program in nonsegmented mode. A conversational program consists of a sequence of alternating entries and

responses between a user and the program. File and database position and locks, and storage resources are held across the terminal I/O operation.

Pseudoconversational

The CICS term for running a program in segmented mode. A pseudoconversational program consists of a series of single CICS tasks designed to appear to the user as a continuous conversation. File and database position and locks, and storage resources are released across the terminal I/O operation. The program must save conversation status before terminal output and restore it on terminal input.

Communication area (COMMAREA)

A data area used to transfer information between two programs within a transaction or between two transactions from the same terminal. When one program transfers to another, the COMMAREA can be any data area the transferring program can access. The transferred-from program can both pass data to that area and receive results in the area. The data area is usually the working storage area of that program.

Transaction Work Area (TWA)

A fixed length storage area allocated for each transaction task control area. Generated EGL programs use a 1024 byte section of the TWA. The offset of the EGL section of bytes is controlled by the **twasOffset** build descriptor option.

Resource Definitions Online (RDO)

Definitions of resources used or managed by the CICS system. Each definition is created by using resource definition online (RDO). The following list shows the types of definitions:

TDQUEUE

Used to define transient data destinations for the system.

FILE Used to define files used by the system.

PROGRAM

Contains information about each program. The Rational COBOL Runtime for z/Series programs, generated COBOL programs, libraries, and servers, FormGroup online print services programs, FormGroup format modules, and DataTables must be defined. Alternatively, if you use the CICS autoinstall feature for programs you do not need to create PROGRAM definitions.

TRANSACTION

Defines the transaction identifiers that can be entered by program users. For each transaction, it also defines the related program that starts the processing for the transaction.

DB2CONN, DB2ENTRY, and DB2TRAN

Describe the interface between the CICS region and DB2®, including the association of transaction codes with DB2 program plans.

PROFILE, TYPETERM, and TERMINAL

Contain descriptions of terminals, their features, and operating information.

Temporary storage queue

A CICS managed file for storing intermediate results. Records in a temporary storage queue can be accessed serially or by a relative record number. Descriptions of the two types of temporary storage follow:

Auxiliary

A temporary storage queue that is stored on DASD. It can be recovered and maintained from one CICS run to the next.

Main A temporary storage queue that exists in the CICS address space. It is not recoverable and is not maintained from one CICS run to the next.

Transient data queue

A CICS managed file that is serially organized. Descriptions of the two types of transient data queues follow:

Extrapartition transient data queue

A CICS managed serial file in a system sequential data set or tape. The file can be an input file or an output file but not both. Extrapartition queues are not recoverable.

Intrapartition transient data queue

A transient data queue that is accessible to transactions running in a CICS region. The queue can be used for both input and output. On z/OS, the queue is stored in a VSAM entry sequenced data set. Intrapartition queues can be recovered.

Related information

“Developing EGL programs for the CICS environment,” on page 1

There are specific design and development considerations when creating EGL programs for CICS for z/OS and CICS for VSE.

“File techniques in CICS programs”

Defining CICS programs is much the same as defining programs for other environments. There are some file technique considerations you should note that are specific to CICS.

“Printing techniques in CICS” on page 15

CICS handles the **printer** file differently on different platforms.

File techniques in CICS programs

Defining CICS programs is much the same as defining programs for other environments. There are some file technique considerations you should note that are specific to CICS.

Related information

“Using temporary storage”

“Using transient data queues” on page 6

You can use transient data queues for many of the same purposes as an auxiliary temporary storage queue.

“Using spool files in z/OS CICS” on page 7

EGL programs generated for CICS for z/OS can access JES SPOOL files if the serial or print file is associated with the **spool** file type at generation.

“Using spool files in VSE CICS” on page 9

EGL programs generated for VSE batch or CICS for VSE can create and write to a VSE/POWER queue member by setting the **fileType** property to **spool** in the resource association for the serial or print file at generation.

“Using VSAM files” on page 13

An EGL program generated for CICS can access VSAM files if the serial, indexed or relative file is associated with the **vsam** file type at generation.

“Using recordName.resourceAssociation” on page 14

You can dynamically change the physical file associated with a record at run time.

Using temporary storage

In CICS, temporary storage is the primary method for storing data that must be available to multiple transactions. Data items in temporary storage are placed in queues with names assigned dynamically by the program storing the data. Temporary storage is implemented in two different ways: main temporary storage and auxiliary temporary storage:

- Main indicates that the queue is stored in space taken from the dynamic storage area.
- Auxiliary indicates that the queue is written to an entry-sequenced VSAM data set.

Main and auxiliary storage have the following characteristics:

- CICS maintains an index of items in main storage.
- Main temporary storage requires more virtual storage than does auxiliary. It should be used for small queues that have short lifetimes or are accessed frequently.
- Auxiliary temporary storage is designed for large amounts of data that must be stored for a long time or are accessed infrequently.
- Queues can be recovered in auxiliary temporary storage.

Note: Only one transaction at a time can update a recoverable temporary storage queue. Keep in mind the probability of enqueues as you design your program. You should also ensure that there are enough VSAM strings to eliminate as much contention as possible.

- If a task attempts to write to temporary storage and the space is not available, CICS suspends the task. The task is not resumed until another task frees the needed space in main storage or in the VSAM data set.
- Rational COBOL Runtime for zSeries uses temporary storage to save information about the program during a segmented **converse** or to save a copy of the form during a transfer using a **show** statement. You can use the **workDBType** build descriptor option to specify whether the main or auxiliary temporary storage is to be used.

Accessing temporary storage from EGL

An EGL program generated for the CICS environment can access CICS temporary storage as a serial or relative record. The following I/O statements are valid when you access temporary storage:

- **add**
- **close**
- **delete**
- **get**
- **get next**
- **get forUpdate**
- **replace**

The resource association for the file must have the EGL file type specified as **tempaux** (auxiliary storage file) or **tempmain** (main storage file) when the program is generated. The system resource name (**systemName** property) is the queue name associated with the temporary storage file.

Temporary storage files can be used by only one task at a time. EGL generates the following CICS commands for you:

- When the queue is first accessed, EGL enqueues with a CICS ENQ command (NOSUSPEND option) on the resource name *EZETEMP-queueName*.
- When the file is closed (**close** statement or end of program) or when recoverable resources are committed, EGL dequeues with a CICS DEQ command.

Non-EGL programs that access the same file should enqueue on the same system resource name while accessing the file.

Records in temporary storage have an additional byte added to the front of the record that indicates the status of the record:

X'01' indicates that the record has been logically deleted.

X'00' indicates that the record logically exists in the file.

The additional byte is added to the record definition and managed by Rational COBOL Runtime for z/Series. Do not include the additional byte in the EGL record definition. However, if the temporary storage file is also used by a non-EGL program, the non-EGL program must allocate space for the byte, interpret the byte, and update it as EGL does. Processing of the additional byte is as follows:

add or replace

The byte is set to **X'00'**.

delete The byte is set to **X'01'** and the record length is set to 1.

get next

Records with a value of **X'01'** are skipped.

get or get forUpdate

Records with a value of **X'01'** cause a **noRecordFound** record state to be set.

The **close** statement does not delete temporary storage files. Use the **sysLib.purge()** system function to delete the file. EGL enqueues by generating a CICS ENQ command with the NOSUSPEND option on resource name *EZETEMP-queueName* when **sysLib.purge()** is used and dequeues (DEQ command) after the queue is deleted.

Related information

“Developing EGL programs for the CICS environment,” on page 1

There are specific design and development considerations when creating EGL programs for CICS for z/OS and CICS for VSE.

“File techniques in CICS programs” on page 3

Defining CICS programs is much the same as defining programs for other environments. There are some file technique considerations you should note that are specific to CICS.

“Using transient data queues”

You can use transient data queues for many of the same purposes as an auxiliary temporary storage queue.

“Using spool files in z/OS CICS” on page 7

EGL programs generated for CICS for z/OS can access JES SPOOL files if the serial or print file is associated with the **spool** file type at generation.

“Using spool files in VSE CICS” on page 9

EGL programs generated for VSE batch or CICS for VSE can create and write to a VSE/POWER queue member by setting the **fileType** property to **spool** in the resource association for the serial or print file at generation.

“Using VSAM files” on page 13

An EGL program generated for CICS can access VSAM files if the serial, indexed or relative file is associated with the **vsam** file type at generation.

“Using recordName.resourceAssociation” on page 14

You can dynamically change the physical file associated with a record at run time.

Using transient data queues

You can use transient data queues for many of the same purposes as an auxiliary temporary storage queue.

Like temporary storage, intrapartition transient data consists of data queues in a single data set with an index in main storage. Transient data queues differ from auxiliary temporary storage queues in the following ways:

- Transient data queue names must be defined in the RDO TDQUEUE entry before CICS is started. Transient data queues do not have the same random access characteristics as temporary storage queues.
- Transient data queues must be read sequentially, and each item can be read only once. After a transaction reads an item, the item is removed from the queue and is not available to any other transaction.
- Items in a transient data queue cannot be changed.
- Transient data queues are always written to a data set.
- Writing items to a transient data queue can initiate a specific transaction when the trigger level for the queue is reached.
- A transient data queue can be physically or logically recoverable, and you can specify that you want areas of the entry sequenced data set (ESDS) that have been written and read to be reused for new data.
- You can direct print output to a transient data queue but not to a temporary storage queue.
- Because the commands for intrapartition and extrapartition data sets are the same, you can switch between the internal CICS facility and an external data set. You need change only the RDO TDQUEUE entry.

Accessing transient data queues from EGL

An EGL program generated for the CICS environment can access CICS transient data queues as a serial record. The following I/O statements are valid when you access a transient data queue:

- **add**
- **close**
- **get next**

The resource association for the file must have the EGL file type specified as **transient** when the program is generated. The system resource name (**systemName** property) is the name of the transient data queue as it is defined in the corresponding TDQUEUE entry.

You can also use the resource association to direct print output to a transient data queue.

Related information

“Developing EGL programs for the CICS environment,” on page 1

There are specific design and development considerations when creating EGL programs for CICS for z/OS and CICS for VSE.

“File techniques in CICS programs” on page 3

Defining CICS programs is much the same as defining programs for other environments. There are some file technique considerations you should note that are specific to CICS.

“Using temporary storage” on page 4

“Using spool files in z/OS CICS”

EGL programs generated for CICS for z/OS can access JES SPOOL files if the serial or print file is associated with the **spool** file type at generation.

“Using spool files in VSE CICS” on page 9

EGL programs generated for VSE batch or CICS for VSE can create and write to a VSE/POWER queue member by setting the **fileType** property to **spool** in the resource association for the serial or print file at generation.

“Using VSAM files” on page 13

An EGL program generated for CICS can access VSAM files if the serial, indexed or relative file is associated with the **vsam** file type at generation.

“Using recordName.resourceAssociation” on page 14

You can dynamically change the physical file associated with a record at run time.

Using spool files in z/OS CICS

EGL programs generated for CICS for z/OS can access JES SPOOL files if the serial or print file is associated with the **spool** file type at generation.

The system resource name for a spool file depends on whether the file is an input or output file:

Input file

Maximum 10-byte name in the format *userid.class*.

userid

A 4- to 8-character external writer name or an asterisk. If you use an external writer name, CICS requires that the first 4 characters of the external writer name be the same as the first 4 characters of the CICS APPLID used to identify the CICS region to ACF/VTAM.

class

An optional 1-character spool class; the default is "A".

Output file

Maximum 19-byte name in the format *nodeid.userid.class*.

nodeid

A 1- to 8-character system node ID. You can use an asterisk for *nodeid*.

userid

A 1- to 8-character system user ID. You can use an asterisk for *userid*. If you do not specify *class*, *userid* is also optional and defaults to the CICS user id (the same value that is stored in **sysVar.userID**).

class

An optional 1-character spool class; the default is "A".

Refer to the CICS customization manual for more information.

Do not use spool files as temporary files that a program writes to and then reads. You can specify the same resource name for the output and input file, but in this case the resource name represents a destination rather than a specific file. If you write to a spool destination and close the file, the file might not be immediately available for input file from that destination and might be queued behind other files sent to the same destination.

For more information on spool file access in CICS, refer to the CICS customization manual.

Spool files are opened on first access and closed in one of the following circumstances:

- The program ends.
- A **close** statement refers to the file.
- Recoverable resources are committed (**sysLib.commit()**, **sysLib.rollback()**, end of transaction or segment).

Related information

“Developing EGL programs for the CICS environment,” on page 1

There are specific design and development considerations when creating EGL programs for CICS for z/OS and CICS for VSE.

“File techniques in CICS programs” on page 3

Defining CICS programs is much the same as defining programs for other environments. There are some file technique considerations you should note that are specific to CICS.

“Using temporary storage” on page 4

“Using transient data queues” on page 6

You can use transient data queues for many of the same purposes as an auxiliary temporary storage queue.

“Using spool files in VSE CICS”

EGL programs generated for VSE batch or CICS for VSE can create and write to a VSE/POWER queue member by setting the **fileType** property to **spool** in the resource association for the serial or print file at generation.

“Using VSAM files” on page 13

An EGL program generated for CICS can access VSAM files if the serial, indexed or relative file is associated with the **vsam** file type at generation.

“Using recordName.resourceAssociation” on page 14

You can dynamically change the physical file associated with a record at run time.

Using spool files in VSE CICS

EGL programs generated for VSE batch or CICS for VSE can create and write to a VSE/POWER queue member by setting the **fileType** property to **spool** in the resource association for the serial or print file at generation.

A serial or print output file associated as a **spool** file type can be created and routed to the RDR, LST, or PUN VSE/POWER queue. EGL programs generated for CICS for VSE can also read from a VSE/POWER queue member by associating a serial file as the **spool** file type at generation.

The first **add** statement for a record that is associated with a spool file creates a new VSE/POWER queue member and adds the data to the beginning of the file. Later **add** statements place data following the previously added data until the file is closed. A **close** statement issued for a spool file closes the VSE/POWER queue member.

Once a spool file is closed, a later **add** statement for a record with the same file name creates a new VSE/POWER queue member. When adding data to a spool file that is to be routed to the LST VSE/POWER queue, you must be aware of the following: VSE/POWER LST queue members are opened by IBM Rational COBOL Runtime for zSeries with the ASA option. This specifies that the report is created using an ANSI printer-control character at the beginning of each line of data. If the file is a serial file, you must ensure that valid carriage control characters are used. If the file is a print file, then Rational COBOL Runtime adds the printer-control characters for you.

A spool file is closed in any of the following circumstances:

- The program ends.
- A **close** statement references the file.
- Recoverable resources are committed (**sysLib.commit()**, **sysLib.rollback()**, end of transaction or segment).

Any close indicates the end of the file.

Because you have a choice of a VSE/POWER queue destination when creating an output spool file, you have the ability to create a file that is placed on the VSE/POWER RDR queue as a batch job. Note that a **close** statement that refers to a spool file that is a RDR queue member indicates the end of the file. A subsequent **add** to the RDR queue file creates a new RDR queue member to be processed as a separate batch job. Also note that when creating jobs, if a POWER EOJ statement is output, the POWER job is made available to run before the spool file is closed.

System resource name format for VSE spool file

In z/VSE, the system resource name for a spool file depends on whether the file is used for input or output. An input spool file can only be used for CICS for z/VSE, not for zVSE batch.

Note: Rational COBOL Runtime performs no error checking to ensure that a correct combination of values is specified for the qualifiers in the system resource name. Rational COBOL Runtime sends the values for each of the system resource name qualifiers "as is" to VSE/POWER. If the **v60ExceptionCompatibility** program property is set to YES, Rational COBOL Runtime places the return code from VSE/POWER in **sysVar.errorCode**. If the **v60ExceptionCompatibility** program property is set to NO, a file I/O exception is thrown.

Input file

Maximum 10-byte name in the format *userid.class*

userid

A 1- to 8-character VSE/POWER identifier of the program or user that will process the report. The identifier must not include blank or null characters. There is no default for this value.

class

A 1-character spool class. The *class* component is optional; the default is "A". You cannot use an asterisk to cause EGL to use a default *class* for an input spool file. The input spool file is read from the VSE/POWER PRT or PUN queue.

Output file

Maximum 65-byte name in the format

jobname.queue.class.disp.form.node.userid.parm

Note: The *jobname* parameter must be specified, or the default must be explicitly requested by specifying an asterisk (*). All other parameters can request the default value by specifying an asterisk (*) or a blank. However, if you request a default value for a parameter by using a blank, default values are used for all subsequent parameters.

jobname

A 1- to 8-character name that defines the jobname for the VSE/POWER queue member. This value is used when the CICS Report Control Facility (RCF) is being used; an asterisk (*) in this field causes the EGL file name to be used for the record or EZEPRINT for a print file. When RCF is not being used (for example, when the queue is PUN or LST), the value in *jobname* is ignored, and the VSE/POWER queue member jobname is the CICS for z/VSE program ID.

queue

A 3-character name that identifies the destination VSE/POWER queue for the file. The following values are valid:

- **RDR** for job output
- **LST** for list output
- **PUN** for punch output
- **PRT** for list output (using RCF)

Using any other characters for queue causes a spool name error. An asterisk (*) or a blank in this field causes the PRT queue to be used.

When *queue* is set to RDR or PRT, RCF is used to access the file. When *queue* is set to PUN or LST, basic CICS SPOOL support is used. Note that both LST and PRT specify that the file is to be a part of the VSE/POWER LST queue, but PRT uses RCF commands while LST does not.

If you attempt to use RCF when you do not have RCF installed on your CICS system, CICS returns an error message. This might be an AEY9 transaction abend, a NOSPOOL condition, or the message "SPOOLING SYSTEM IS NOT AVAILABLE".

When the queue value is PRT or LST, Rational COBOL Runtime for z/VSE opens the file with the ASA option. This option specifies that the report is created using an American National Standard printer-control character at the beginning of each line of data. If you are using a serial file, you must ensure that valid carriage control characters are used. If the file is a print file, Rational COBOL Runtime for z/VSE automatically adds the American National Standard printer-control characters for you.

class

A single character that specifies class. An asterisk (*) or blank in this field causes the default of a null string to be used.

disp

A single character that specifies the VSE/POWER disposition status of the queue member once it is closed. The following values are valid:

- | | |
|----------|--|
| D | Process the job and delete it after processing. |
| H | Hold the job in queue until released. |
| K | Process the job and keep it in the queue after processing. |
| L | Let the job stay in the queue until released. |

Using any other character for *disp* causes a spool name error. This field is not applicable when *queue* is LST or PUN. An asterisk (*) or blank in this field causes the default of "D" to be used.

form

A 4-character name that identifies the form number for print output. An asterisk (*) or a blank in this field causes the default of your location's standard form to be used.

node

A 1- to 8-character name that specifies the system node ID. An asterisk (*) or a blank in this field defaults to the current system node ID.

userid

A 1- to 8-character VSE/POWER identifier of the program or user that will process the report. It must not include blanks or null characters.

Under CICS for z/VSE, an asterisk (*) or a blank in *userid* is handled in the following way:

- If you are signed on to CICS, then *userid* defaults to the contents of **sysVar.userID**.
- If you are not signed on to CICS, then *userid* remains an asterisk (or, if blank, is changed to an asterisk).
- If *userid* is set to ANY, it is reset to an asterisk.

Under z/VSE Batch, an asterisk (*) or blank in the *userid* defaults to ANY.

parm

A string of characters used to specify additional information when *queue* is set to LST or PRT. The format of *parm* varies in the following way depending on the runtime environment and the value of *queue*:

- If the runtime environment is VSECICS:
 - If *queue* is set to LST, *parm* is set to *outdescr* as described later in this section.
 - If *queue* is set to PRT, *parm* is set to *fcfname.copies* as described later in this section.
- If the runtime environment is VSEBATCH:
 - If *queue* is set to LST or PRT, *parm* is set to *fcfname.copies*.

outdescr

A string that is passed to CICS in the OUTDESCR option of the CICS for z/VSE SPOOLOPEN OUTPUT command. The characters must be specified in the correct format for the OUTDESCR option. The parameters use the same keywords and values as are used on the VSE/POWER LST statement for program-user-defined output operands, but the syntax varies slightly. The following example shows how you can use FORMDEF FORM1 and PAGEDEF PAGE1 as the *parm* string:

```
FORMDEF(FORM1) PAGEDEF(PAGE1)
```

The following example shows how that *parm* might be used in the corresponding spool file name:

```
JOBNAME1.LST.*.*.*.*.FORMDEF(FORM1) PAGEDEF(PAGE1)
```

Rational COBOL Runtime for z/VSE handles the calculation and insertion of the length area at the beginning of the string as required by CICS for z/VSE. The maximum length of the *parm* string is variable and depends on the length of the spool file specification up to this point; the total length of the spool file specification cannot be over 65 characters.

fcfname

The name of the FCB-image phase which VSE/POWER uses for printing the related job output. The name phase must be cataloged in a sublibrary accessible from the VSE/POWER partition. The name can be up to 8 alphanumeric characters (letters, numbers, and special characters). If omitted, the system default FCB is used. The default name can be specified with an asterisk (*).

copies

A number from 1 to 255 that specifies the number of copies to be printed. The default is 1.

Do not use spool files as temporary files for a program that writes to a file and then reads the file. You can specify the same resource name for an output and input file, but the resource name represents a destination rather than a specific file. If you write to a spool destination and close the file, the file might not be immediately available as an input file from that destination and might be queued behind other files sent to the same destination.

For more information on spool file access in CICS, see the CICS customization manual.

Spool files are opened on first access and closed in one of the following circumstances:

- The program ends.
- A **close** statement refers to the file.
- Recoverable resources are committed (**sysLib.commit()**, **sysLib.rollback()**, end of transaction or segment).

Related information

“Developing EGL programs for the CICS environment,” on page 1

There are specific design and development considerations when creating EGL programs for CICS for z/OS and CICS for VSE.

“File techniques in CICS programs” on page 3

Defining CICS programs is much the same as defining programs for other environments. There are some file technique considerations you should note that are specific to CICS.

“Using temporary storage” on page 4

“Using transient data queues” on page 6

You can use transient data queues for many of the same purposes as an auxiliary temporary storage queue.

“Using spool files in z/OS CICS” on page 7

EGL programs generated for CICS for z/OS can access JES SPOOL files if the serial or print file is associated with the **spool** file type at generation.

“Using VSAM files”

An EGL program generated for CICS can access VSAM files if the serial, indexed or relative file is associated with the **vsam** file type at generation.

“Using recordName.resourceAssociation” on page 14

You can dynamically change the physical file associated with a record at run time.

Using VSAM files

An EGL program generated for CICS can access VSAM files if the serial, indexed or relative file is associated with the **vsam** file type at generation.

The system resource name (**systemName** property) is the RDO FILE name for the data set as it is defined to CICS.

For CICS, the **close** statement does not actually close the data set. The **close** statement releases record locks and position in the file.

When accessing the same indexed data set using two file names for the same physical data set or two file names that access a base data set and its alternate

index, an indefinite deadlock can occur in CICS that does not raise the **deadlock** condition. When the file is not defined with LSRPOOLID equal to NONE in the RDO FILE entry and one I/O statement in a program has performed a **get next** on a file and another I/O statement performs a **get...forUpdate** or **add** statement for the same file (or alternate index) without ending the **get next**, this deadlock can occur. If you design this type of file access into your programs, make sure LSRPOOLID for the file is set to NONE to avoid the deadlock.

Related information

“Developing EGL programs for the CICS environment,” on page 1

There are specific design and development considerations when creating EGL programs for CICS for z/OS and CICS for VSE.

“File techniques in CICS programs” on page 3

Defining CICS programs is much the same as defining programs for other environments. There are some file technique considerations you should note that are specific to CICS.

“Using temporary storage” on page 4

“Using transient data queues” on page 6

You can use transient data queues for many of the same purposes as an auxiliary temporary storage queue.

“Using spool files in z/OS CICS” on page 7

EGL programs generated for CICS for z/OS can access JES SPOOL files if the serial or print file is associated with the **spool** file type at generation.

“Using spool files in VSE CICS” on page 9

EGL programs generated for VSE batch or CICS for VSE can create and write to a VSE/POWER queue member by setting the **fileType** property to **spool** in the resource association for the serial or print file at generation.

“Using VSAM files” on page 13

An EGL program generated for CICS can access VSAM files if the serial, indexed or relative file is associated with the **vsam** file type at generation.

“Using recordName.resourceAssociation”

You can dynamically change the physical file associated with a record at run time.

Using recordName.resourceAssociation

You can dynamically change the physical file associated with a record at run time.

Move the system resource name of the new file to *recordName.resourceAssociation*. The new system resource is used in the next I/O statement associated with the record. If another resource is already open for the record, that file is closed before the new file is accessed. The new resource must have the same file type as the file type specified for the record file when the program was generated.

Related information

“Developing EGL programs for the CICS environment,” on page 1

There are specific design and development considerations when creating EGL programs for CICS for z/OS and CICS for VSE.

“File techniques in CICS programs” on page 3

Defining CICS programs is much the same as defining programs for other environments. There are some file technique considerations you should note that are specific to CICS.

“Using temporary storage” on page 4

“Using transient data queues” on page 6

You can use transient data queues for many of the same purposes as an auxiliary temporary storage queue.

“Using spool files in z/OS CICS” on page 7

EGL programs generated for CICS for z/OS can access JES SPOOL files if the serial or print file is associated with the **spool** file type at generation.

“Using spool files in VSE CICS” on page 9

EGL programs generated for VSE batch or CICS for VSE can create and write to a VSE/POWER queue member by setting the **fileType** property to **spool** in the resource association for the serial or print file at generation.

“Using VSAM files” on page 13

An EGL program generated for CICS can access VSAM files if the serial, indexed or relative file is associated with the **vsam** file type at generation.

Printing techniques in CICS

CICS handles the **printer** file differently on different platforms.

Programs write printer data when the program processes a **print** statement for a print form. The printer output is in line character format with ANSI printer-control characters. The printer data is written to a logical file named **printer**.

- For z/OS CICS, you can associate the **printer** file with either a transient data queue (**transient** file type) or a JES spool file (**spool** file type) at generation.
- For VSE CICS, you can associate the **printer** file with either a transient data queue (**transient** file type) or a VSE/POWER file (**spool** file type) at generation.

Related information

“Using transient data queues for printer output”

If you associate **printer** with a transient data queue at generation, the system resource name (**systemName** property) is the RDO TDQUEUE name for the queue.

“Using spool files for printer output on z/OS CICS” on page 16

If the **printer** file is associated with the **spool** file type at generation, the system resource name (**systemName** property) identifies the node, user or external writer identifier, and class that you want to spool the file.

“Using VSE/POWER files for printer output” on page 17

“Using converseVar.printerAssociation” on page 18

You can set the value of the **converseVar.printerAssociation** system variable to change the print file destination (transient data queue or spool file name) at run time.

Using transient data queues for printer output

If you associate **printer** with a transient data queue at generation, the system resource name (**systemName** property) is the RDO TDQUEUE name for the queue.

You can define the destination for the queue as a system printer, a terminal printer, or a data set. If the destination is a terminal printer, you need to define a transaction that is started when data is written to the queue. The transaction runs the Rational COBOL Runtime program FZETPRT. FZETPRT reads the queue and writes the data to the terminal printer identified in the RDO TDQUEUE entry.

The program does not actually write the printer output to the transient data queue until the print file is closed. The printed output is accumulated in temporary storage. When the file is closed (**close** statement or end of transaction), Rational COBOL Runtime carries out the following steps:

1. Enqueues on the transient data queue using the value of the **systemName** property as the resource name
2. Copies the printer output to the queue
3. Dequeues

The maximum number of print records that can be accumulated in the transient data queue is 32765. Your program must close the print file before 32765 records are accumulated.

Related information

"Developing EGL programs for the CICS environment," on page 1

There are specific design and development considerations when creating EGL programs for CICS for z/OS and CICS for VSE.

"Printing techniques in CICS" on page 15

CICS handles the **printer** file differently on different platforms.

"Using spool files for printer output on z/OS CICS"

If the **printer** file is associated with the **spool** file type at generation, the system resource name (**systemName** property) identifies the node, user or external writer identifier, and class that you want to spool the file.

"Using converseVar.printerAssociation" on page 18

You can set the value of the **converseVar.printerAssociation** system variable to change the print file destination (transient data queue or spool file name) at run time.

Using spool files for printer output on z/OS CICS

If the **printer** file is associated with the **spool** file type at generation, the system resource name (**systemName** property) identifies the node, user or external writer identifier, and class that you want to spool the file.

The name is in the following format:

nodeid.userid.class

nodeid

A 1- to 8-character system node ID.

userid

A 1- to 8-character system user ID.

class

A 1-character spool class.

You can specify an asterisk to use the default *userid* and *nodeid*. The *class* component is optional and defaults to "A". If you do not specify *class*, *userid* is also optional and defaults to the CICS user id (the same value as stored in **sysVar.userID**). The maximum name size is 19 bytes.

Refer to the CICS customization manual for more information. The spool file is opened as needed on **print** statements and closed on **close** statements for a print form, or when recoverable resources are committed (**sysLib.commit()**, **sysLib.rollback()**, or end of transaction or segment).

Related information

“Developing EGL programs for the CICS environment,” on page 1

There are specific design and development considerations when creating EGL programs for CICS for z/OS and CICS for VSE.

“Printing techniques in CICS” on page 15

CICS handles the **printer** file differently on different platforms.

“Using VSE/POWER files for printer output”

“Using **converseVar.printerAssociation**” on page 18

You can set the value of the **converseVar.printerAssociation** system variable to change the print file destination (transient data queue or spool file name) at run time.

Using VSE/POWER files for printer output

Printing is initiated when a program processes a **print** statement for an EGL-defined print form. Printing can also be initiated when the program processes an **add** statement, in the circumstances described later in this topic.

When you initiate printing through a **print** statement and print form, the printer output is routed to the file that is specified as the resource associated with **printer**. You can specify the resource associated with **printer** at generation or at run time (using the **converseVar.printerAssociation** system variable).

On CICS for z/VSE, if the resource associated with **printer** is a **spool** file, it is spooled to VSE/POWER. You can specify that queue as LST or PRT, causing the **spool** file to become a VSE/POWER LST queue member. You can specify the jobname, class, disp, form, node, and userid of the VSE/POWER LST queue member. You specify these values using the system resource name format for the spool file. For more information, see “Using spool files in VSE CICS” on page 9.

You can use the CICS Report Controller in conjunction with Rational COBOL Runtime printer functions to provide ease of handling printed output.

Printing can also be initiated when the program processes an **add** statement for a serial record that is associated with the **spool** file type and with a **systemName** property that specifies a destination queue of LST. The **systemName** property specifies the VSE/POWER queue destination. If the queue is specified as LST or PRT, the file becomes a VSE/POWER LST queue member.

Related information

“Developing EGL programs for the CICS environment,” on page 1

There are specific design and development considerations when creating EGL programs for CICS for z/OS and CICS for VSE.

“Printing techniques in CICS” on page 15

CICS handles the **printer** file differently on different platforms.

“Using spool files for printer output on z/OS CICS” on page 16

If the **printer** file is associated with the **spool** file type at generation, the system resource name (**systemName** property) identifies the node, user or external writer identifier, and class that you want to spool the file.

“Using `converseVar.printerAssociation`”

You can set the value of the **converseVar.printerAssociation** system variable to change the print file destination (transient data queue or spool file name) at run time.

Using `converseVar.printerAssociation`

You can set the value of the **converseVar.printerAssociation** system variable to change the print file destination (transient data queue or spool file name) at run time.

To change the print file destination, the program sets the **converseVar.printerAssociation** system variable to the new system resource name for the print file before the **print** statement is run. The new resource must have the same file type as the one specified for **printer** when the program was generated.

Multiple print files can be open at the same time. A **print** statement writes to the resource named in **converseVar.printerAssociation** at the time the statement is run. A **close** statement for a print form closes only the resource named in **converseVar.printerAssociation**. Any files not explicitly closed are closed at the end of the transaction or segment, or commit point for spool files.

The default value for **converseVar.printerAssociation** is the system resource name specified for the **printer** file at generation. If you set the **printDestination** build descriptor option to **TERMINALID** and the program was started with a **vgLib.startTransaction()** that had the *termID* parameter set to binary zeros and that specified a *prID* parameter, then **converseVar.printerAssociation** is initialized to the value in the *prID*.

If the program was started by a non-EGL program that specified the **RTERMID** on the **START** command, then **converseVar.printerAssociation** is initialized to the value specified for **RTERMID**.

Related information

“Developing EGL programs for the CICS environment,” on page 1

There are specific design and development considerations when creating EGL programs for CICS for z/OS and CICS for VSE.

“Printing techniques in CICS” on page 15

CICS handles the **printer** file differently on different platforms.

“Using spool files for printer output on z/OS CICS” on page 16

If the **printer** file is associated with the **spool** file type at generation, the system resource name (**systemName** property) identifies the node, user or external writer identifier, and class that you want to spool the file.

“Using VSE/POWER files for printer output” on page 17

“Debugging CICS programs”

Test CICS programs by first fixing logic errors in the EGL debugger, then generating the programs for your CICS test region.

Debugging CICS programs

Test CICS programs by first fixing logic errors in the EGL debugger, then generating the programs for your CICS test region.

The EGL debugger runs only on Windows and Linux. There are steps you must take in debugging because you cannot directly access CICS resources from the debugger:

- Set your resource association **fileType** for the debugger to **seqws** (workstation sequential file) for a serial file that you plan to implement as a CICS transient data queue or spool file when you generate for CICS. Do the same thing for a print file that you plan to implement as a CICS transient data queue or spool file at runtime.
- Set your resource association **fileType** to **ibmcobol** (remote VSAM file on the z/OS or VSE host) for a relative file that you plan to implement as a CICS temporary storage queue when you generate for CICS. Note that Linux does not support remote VSAM files.
- Set your resource association **fileType** to **ibmcobol** (remote VSAM file on the z/OS or VSE host) for an indexed or relative file that you plan to implement as a VSAM file when you generate for CICS. Note that Linux does not support remote VSAM files.

When you have fixed any logic errors in your program, generate your program again, this time for your CICS test region. In the CICS test region, you can verify interactions that must be set up specifically for CICS, including I/O with the following files and devices:

- temporary storage queues
- transient data queues
- spool files
- printers

Related reference

“Developing EGL programs for the CICS environment,” on page 1

There are specific design and development considerations when creating EGL programs for CICS for z/OS and CICS for VSE.



Resource associations part

“Setting the recovery unit of work”

The **sysLib.commit()** system function notifies EGL that the current recovery unit of work is complete and a new unit of work is to be started.

Setting the recovery unit of work

The **sysLib.commit()** system function notifies EGL that the current recovery unit of work is complete and a new unit of work is to be started.

This system function issues a CICS SYNCPOINT command.

There is an implicit **sysLib.commit()** upon completion of a program (**exit program** or a **transfer** or **show** statement). You can explicitly call **sysLib.commit()** at any time during the execution of a program. Best practice is to call **sysLib.commit()** from within programs that have **get forUpdate**, **replace**, **delete**, or **add** statements with a logic loop at the completion of a logical unit of work so that the program does not hold multiple locks.

Take care to prevent deadlocks when you access data sets using VSAM Local Shared Resources (LSR) on CICS from a generated EGL program. When using LSR and using a **get next** or **get previous** statement on a file, call **sysLib.commit()** at the completion of a logical unit of work. Do this prior to attempting a **get forUpdate**, **replace**, **delete**, or **add** statement that requires exclusive use of resources. This releases the **get next** position and allows later updates to the data set.

If you are defining **transfer to program** statements to replace **transfer to transaction** statements, note that the recovery unit of work holds across the **transfer to program** and you must call **sysLib.commit()** if you want to end the unit of work.

A SYNCPOINT occurs on a **transfer to program** statement under the following conditions:

- If a transfer to a non-EGL program occurs and a PSB is scheduled
- If the **synchOnPgmTransfer** build descriptor option is set to "YES" and a PSB is scheduled
- If both of the following conditions are true:
 - The **synchOnPgmTransfer** build descriptor option is set to "NO" for the transferring program
 - The transferring program had scheduled a PSB and different PSBRecord names were specified for the two programs

Related information

“Developing EGL programs for the CICS environment,” on page 1

There are specific design and development considerations when creating EGL programs for CICS for z/OS and CICS for VSE.

“Using CICS functions from EGL programs”

You can use CICS functions in EGL programs.

Using CICS functions from EGL programs

You can use CICS functions in EGL programs.

The following table lists the CICS functions that you can use in EGL programs. The table also summarizes how to use these functions.

Table 1. CICS functions and how to represent them in EGL

CICS Function	EGL Function	Comments
Type of Program		
Conversational	segmented program property = NO	
Pseudoconversational	segmented program property = YES	The workDBType build descriptor option specifies whether to use a main (MAIN) or auxiliary (AUX) temporary storage queue for saving status across the terminal I/O.
Pseudoconversational, different transaction names	Segmented converse with sysVar.transactionID set to the next transaction name	The workDBType build descriptor option specifies whether to use a main (MAIN) or auxiliary (AUX) temporary storage queue for saving status across the terminal I/O.
Terminal and Printer Support		
Communication with the terminal	converse for input/output, display for output, or a show statement with a form for output and input specified in the inputForm property.	
System printer support	print statement; transient file type for printer file	Associate RDO TDQUEUE for transient data queue with system printer
Terminal printer support	print statement; transient file type for printer file	Associate RDO TDQUEUE for transient data queue with terminal printer and trigger FZETPRT transaction to write to printer.
JES SPOOL file for printer output for z/OS CICS	print statement; spool file type for printer file	systemName in EGL resource association entry identifies node, spool writer or user identifier, and class
VSE/POWER SPOOL file for printer support for VSE CICS	print statement; spool file type for printer file on VSE environment	systemName in EGL resource association entry identifies jobname, queue, class, disp, form, node, userid, and parm of the VSE/POWER LST queue member.
Dynamic printer support	converseVar.printerAssociation set to alter print destination for print statement	
Database and File Support		

Table 1. CICS functions and how to represent them in EGL (continued)

CICS Function	EGL Function	Comments
DL/I database definition and access	PSBRecord definition, DL/I segment definition, and normal I/O statements as provided for DL/I database access.	EGL creates default SSAs and sets the default PCB number.
PSB Scheduling	dliVar.dliPsbName identifies the PSB to be scheduled.	Scheduling is done automatically prior to the first DL/I operation in the unit of work.
PSB Termination	Done automatically on sysLib.commit() , sysLib.rollback() , or end of transaction or segment.	A CICS SYNCPOINT occurs on a transfer to program statement under the following conditions: <ul style="list-style-type: none"> • If transfer to a non-EGL program occurs and a PSB is scheduled • If synchOnPgmTransfer = "YES" and a PSB is scheduled • If synchOnPgmTransfer = "NO" for the transferring program, and the transferring program had scheduled a PSB and different PSBRecord names were specified for the two programs.
Program restart following abnormal termination due to deadlock in queueing on database records	dliVar.cicsRestart	Variable indicating whether the program was restarted
DB2 database definition and access	SQLRecord definition and normal I/O statements as provided for relational database access.	
VSAM file support	vsam file type for serial, relative, and indexed files	RDO FILE entry required for file; systemName in EGL resource association entry matches FILE entry name
Transient data queue support	fileType property set to transient for serial files	RDO TDQUEUE entry required for file; systemName in EGL resource association entry matches TDQUEUE entry name
Function shipping for VSAM data sets and transient data queues	remoteFile type for the fileLink element in the linkage options entry for the file	

Table 1. CICS functions and how to represent them in EGL (continued)

CICS Function	EGL Function	Comments
Specifying SYSID when function shipping.	remoteFile and locationSpec = PROGRAMCONTROLLED for the fileLink element in the linkage options entry for the file; sysLib.remoteSystemID to dynamically set the remote system name	
Main temporary storage queue support	tempmain file type for serial or relative file	Records have extra control byte in byte 1
Auxiliary temporary storage queue support	tempaux file type for serial or relative file	Records have extra control byte in byte 1
JES SPOOL file support for z/OS CICS	spool file type for serial file	systemName in EGL resource association entry identifies node (output only), spool writer or user identifier, and class
VSE/POWER SPOOL file for VSE CICS	spool file type for serial file	systemName in EGL resource association entry identifies spool writer or user identifier and class (input only); or jobname, queue, class, disp, form, node, userid, and parm for a VSE/POWER RDR, PUN, or LST queue member (output only).
Program Communications		
START transaction	A transfer to program statement or call to vgLib.startTransaction()	
RETURN TRANSID	A show statement or a segmented converse statement	
Function shipping for START transaction	vgLib.startTransaction() with asynchLink element, type=remoteAsynch in linkage options entry for the record specified in vgLib.startTransaction()	
Specifying SYSID when function shipping	asynchLink element, type = remoteAsynch and locationSpec = PROGRAMCONTROLLED in linkage options entry; sysLib.remoteSystemID to dynamically set the remote system name	
XCTL to another EGL program	A transfer to transaction statement	If a record is specified, it is transferred in the COMMAREA.

Table 1. CICS functions and how to represent them in EGL (continued)

CICS Function	EGL Function	Comments
XCTL to a non-EGL program	A transfer to transaction statement with externallyDefined option	If a record is specified, it is transferred in the COMMAREA.
LINK to program with data in COMMAREA	call statement; callLink element, type = localCall , linkType = "CICSLINK" , and parmForm = "COMMDATA" for linkage options entry for called program	
Distributed program LINK to program with data in COMMAREA	call statement; callLink element, type = remoteCall , linkType = "CICSLINK" , and parmForm = "COMMDATA" for linkage options entry for called program	
Specifying SYSID on distributed program LINK	callLink element, type=remoteCall and serverID = "serverName" on linkage options entry for called program	
Specifying TRANSID on distributed program LINK	callLink element, type=remoteCall and serverID = "transactionName" on linkage options entry for called program	
Specifying SYNCONRETURN on distributed program LINK	callLink element, type=remoteCall and luwControl ="SERVER" on linkage options entry for called program	
Miscellaneous		
SYNCPPOINT	sysLib.commit()	
SYNCPPOINT ROLLBACK	sysLib.rollback()	
JOURNAL call	sysLib.audit()	

Related information

"Developing EGL programs for the CICS environment," on page 1

There are specific design and development considerations when creating EGL programs for CICS for z/OS and CICS for VSE.

"Communicating between multiple CICS transactions"

Communicating between multiple CICS transactions

Programs running in a CICS environment can communicate with other programs in the same CICS region using shared DataTables. DataTables defined as shared cause all programs in the same CICS region to use the same copy of the DataTable until a new copy is requested. In CICS environments, shared DataTables can be modified at run time. Because of this, multiple EGL programs running in the same CICS region could use a shared DataTable as a shared communications area. This use of DataTables might have synchronization considerations depending on the specific CICS platform and the way the data is modified in the DataTable.

For CICS for z/OS and CICS for VSE, modifications to shared DataTables are not synchronized across **call** statements or I/O statements.

Related information

“Developing EGL programs for the CICS environment,” on page 1

There are specific design and development considerations when creating EGL programs for CICS for z/OS and CICS for VSE.

“Inter-transaction affinity considerations in a CICSplex”

Inter-transaction affinity considerations in a CICSplex

A *CICSplex* consists of two or more CICS regions that are linked using CICS intercommunication facilities. A CICS function known as *dynamic transaction routing* supports load balancing by dynamically routing a transaction from a terminal to any of the regions that have the resources to process the transaction.

Inter-transaction affinity occurs when two or more CICS transactions pass information to one another in a way that requires the transactions to run in the same CICS region. When inter-transaction affinity exists, you must define the transactions to CICS so that they are routed to the same region.

The following topics describe transaction routing considerations for CICS programs generated using EGL. For a more complete discussion of transaction routing, see *Dynamic Routing in a CICSplex*, SC33-1012.

Segmented programs

Segmented programs use a temporary storage queue (the work database) for saving the state of the program conversation during a **converse** or **show** statement. All segments of the conversation must have access to the same temporary storage queue *and* must continue to use the same terminal ID.

Sharing EGL DataTables for update

Programs can update shared EGL DataTables in the CICS environment. The shared DataTable is stored in memory obtained through a CICS GETMAIN; any updates are accessible only to programs running in the same region. Any transactions dependent on passing information through a shared DataTable must be routed to the same region.

Temporary storage queues

EGL support for temporary storage queues requires that access to the queues be serialized. The generated program does this by using CICS ENQ and DEQ with the queue name as the resource name. ENQ and DEQ are effective only within the scope of a single region. To ensure that access to the queue is serialized, do one of the following:

- Define the temporary storage queue as a local queue.
- Route all transactions that access the queue to same region.
- Use a queue naming convention that includes the terminal ID from **sysVar.terminalID** as part of the queue name, so that a different queue is used for each terminal. Since only one transaction is active from any one terminal at a time, access to the queue is serialized.

Refer to the CICS manual for more information on using temporary storage queues with transaction routing.

Using a transient data queue for printed output

Printed output can be routed to a transient data queue. The program accumulates the printed output in a temporary storage queue. When the output is complete, the program copies the output to the transient data queue, using ENQ/DEQ to ensure that output from multiple transactions in the same system is not interspersed.

Because ENQ/DEQ are effective only within a region, define the queue as a local queue to prevent interspersed output from multiple regions.

Also, if you have defined the queue to trigger the FZETPRT terminal printing program, define the transaction for FZETPRT as a local transaction in the region where the queue resides.

Error destination queue

Error messages from Rational COBOL Runtime can be directed to a transient data queue called the error destination queue. Define the queue as a local queue to each region in which an EGL program can run to ensure that messages related to a single error are not interspersed with messages related to another error occurring at the same time in another region.

Disable on run unit failure

One of the options that can be specified using the diagnostic control utility is disabling a transaction whenever a run-unit error is detected for that transaction.

The disable action is implemented using the CICS SET function and is effective only for the region in which the error occurred.

CICS utility function region affinity

The Rational COBOL Runtime CICS utilities perform functions that have region affinity; therefore, you must ensure that the transaction is routed to the desired region based on the user identifier, LU name, or alternate transaction name. The following table lists the utilities, default transaction identifier, and function description.

Table 2. Region Dependent Utilities

Utility	Default ID	Function Description
CICS Utilities Menu	ELAM	Menu for selecting the other utilities (except trace).
New Copy	ELAN	Loads new copy of program, library, service, FormGroup, or DataTable in region
Diagnostic Message Print	ELAU	Prints error message queue associated with the region.

Table 2. Region Dependent Utilities (continued)

Utility	Default ID	Function Description
Diagnostic Control Options	ELAC	Sets error reporting options for Rational COBOL Runtime. Is region dependent if option file (RDO FILE name ELACFIL) is defined as local to each region; is not region dependent if ELACFIL is defined as a shared file accessed through a file owning region.
Trace	ELAZ	Trace options set by this utility are saved in memory obtained through a CICS GETMAIN in the region and are effective only in the region in which the ELAZ transaction ran.

Related information

“Developing EGL programs for the CICS environment,” on page 1

There are specific design and development considerations when creating EGL programs for CICS for z/OS and CICS for VSE.

Index

C

CICS 20
 CICSplex 25
 communicating between
 transactions 24
 file techniques
 spool files (VSE) 9
 VSAM files 13
 printing techniques
 converseVar.printerAssociation 18

CICS (*continued*)
 printing techniques (*continued*)
 overview 15
 spool files 16
 transient data queues 16
 VSE/POWER output 17
 using functions from EGL 21
CICS program development
 file techniques
 overview 4
 spool files (z/OS) 7

CICS program development (*continued*)
 file techniques (*continued*)
 temporary storage 4
 transient data queues 6
 overview 1
 terminology 1

R

recovery unit of work 20