CICS® Transaction Server for VSE/ESA™

Recovery and Restart Guide

Release 1

SC33-1666-00

CICS® Transaction Server for VSE/ESA™

Recovery and Restart Guide

Release 1

SC33-1666-00

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 151.

First Edition (June 1999)

This edition applies to Release 1 of CICS Transaction Server for VSE/ESA, program number 5648-054, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

The CICS for VSE/ESA Version 2.3 edition remains applicable and current for users of CICS for VSE/ESA Version 2.3.

Order publications through your IBM representative or the IBM branch office serving your locality.

At the back of this publication is a page entitled “Sending your comments to IBM”. If you want to make any comments, please use one of the methods described there.

Contents

Preface

What this book is about

This book contains guidance about determining your CICS recovery and restart needs, deciding which CICS facilities are most appropriate, and implementing your design on your CICS system.

The information in this book is generally restricted to a single CICS system. For guidance on intersystem communication (ISC) and multiregion operation (MRO), see the *CICS Intercommunication Guide*. For information about XRF systems, see the *CICS XRF Guide*. However, the Extended Recovery Facility (XRF) takeover is based on emergency restart processing, so the information in this book is relevant to XRF.

4 This book does not describe recovery and restart for the CICS Front End
4 Programming Interface. For information on this topic, see the *CICS Front End Programming Interface User's Guide*.

Who should read this book

This book is for those responsible for restart and recovery planning, design, and implementation—either for a complete system or for a particular subject.

What you need to know to understand this book

To understand this book, you should have experience of installing and generating a CICS system and the products with which it is to work, or of writing CICS application programs or exit programs. You should also understand your application requirements well enough to be able to make decisions about realistic recovery and restart needs, and the trade-offs between those needs and the performance overhead they incur.

How to use this book

This book deals with a wide variety of topics, all of which contribute to the recovery and restart characteristics of your system. It is unlikely that you would have to implement all the possible techniques discussed in this book, so use the table of contents to find the sections relevant to your work. If you are new to recovery and restart, you should find Part 1 helpful, because it introduces the basic concepts.

Notes on terminology

In this book, the following terms are used:

- **CICS** refers to CICS Transaction Server for VSE/ESA
- **MB** equals 1 048 576 bytes.

Book structure

Part 1, "Overview" on page 1

Describes:

- The reasons and types of error that make it important for recovery and restart to be considered

- The facilities that CICS provides for data recovery, communication recovery, and system recovery.

Part 2, “Recovery and restart processes” on page 15

Describes the processes which CICS goes through at restart, and the processes used for recovery in a running system. The emphasis is on the parts of the processes that you can influence by your recovery strategy and implementation.

Part 3, “Implementing your recovery and restart strategy” on page 55

Describes how to implement the functions of recovery and restart. Each chapter deals in detail with a particular subject, referring back to information about design or processes when necessary.

Notes on terminology

The terms listed in Table 1 are commonly used in the CICS Transaction Server for VSE/ESA Release 1 library. See the *CICS Glossary* for a comprehensive definition of terminology.

Table 1 (Page 1 of 2). Commonly used words and abbreviations in CICS Transaction Server for VSE/ESA Release 1

Term	Definition (and abbreviation if appropriate)
$(the dollar symbol)	In the character sets and programming examples given in this book, the dollar symbol ($) is used as a national currency symbol and is assumed to be assigned the EBCDIC code point X'5B'. In some countries a different currency symbol, for example the pound symbol (£), or the yen symbol (¥), is assigned the same EBCDIC code point. In these countries, the appropriate currency symbol should be used instead of the dollar symbol.
BSM	BSM is used to indicate the basic security management supplied as part of the VSE/ESA product. It is RACROUTE-compliant, and provides the following functions: • Signon security • Transaction attach security
C	The C programming language
CICSplex	A CICSplex consists of two or more regions that are linked using CICS intercommunication facilities. Typically, a CICSplex has at least one terminal-owning region (TOR), more than one application-owning region (AOR), and may have one or more regions that own the resources accessed by the AORs
CICS Data Management Facility	The new CICS Transaction Server for VSE/ESA Release 1 facility to which all statistics and monitoring data is written, generally referred to as “DMF”
CICS/VSE	The CICS product running under the VSE/ESA operating system, frequently referred to as simply “CICS”
COBOL	The COBOL programming language
DB2 for VSE/ESA	Database 2 for VSE/ESA which was previously known as “SQL/DS”.

Table 1 (Page 2 of 2). Commonly used words and abbreviations in CICS Transaction Server for VSE/ESA Release 1

Term	Definition (and abbreviation if appropriate)
ESM	ESM is used to indicate a RACROUTE-compliant external security manager that supports some or all of the following functions: • Signon security • Transaction attach security • Resource security • Command security • Non-terminal security • Surrogate user security • MRO/ISC security (MRO, LU6.1 or LU6.2) • FEPI security.
FOR (file-owning region)—also known as a DOR (data-owning region)	A CICS region whose primary purpose is to manage VSAM and DAM files, and VSAM data tables, through function provided by the CICS file control program.
IBM C for VSE/ESA	The Language Environment version of the C programming language compiler. Generally referred to as “C/VSE”.
IBM COBOL for VSE/ESA	The Language Environment version of the COBOL programming language compiler. Generally referred to as “COBOL/VSE”.
IBM PL/I for VSE/ESA	The Language Environment version of the PL/I programming language compiler. Generally referred to as “PL/I VSE”.
IBM Language Environment for VSE/ESA	The common runtime interface for all LE-conforming languages. Generally referred to as “LE/VSE”.
PL/I	The PL/I programming language
VSE/POWER	Priority Output Writers Execution processors and input Readers. The VSE/ESA spooling subsystem which is exploited by the report controller.
VSE/ESA System Authorization Facility	The new VSE facility which enables the new security mechanisms in CICS TS for VSE/ESA R1, generally referred to as “SAF”
VSE/ESA Central Functions component	The new name for the VSE Advanced Function (AF) component
VSE/VTAM	“VTAM”

Part 1. Overview

This part of the book describes:

- The reasons and types of error that make it important for recovery and restart to be considered
- The facilities that CICS® provides for data recovery, communication recovery, and system recovery.

Chapter 1. Introduction to recovery and restart

This chapter describes some of the basic concepts of the recovery and restart facilities provided by CICS.

The principal topics discussed are:

Faults and their effects

Among the failures that can occur in a data processing system are:

- Communication failures (in online systems)
- Data set or database failures
- Application or system program failures
- Processor failures
- Power supply failures

Comparison of batch and online systems

All these problems are potentially more severe in an online system than in a system that performs only batch processing.

In batch systems, input data is usually prepared before processing begins, and jobs can be rerun, either from the start of the job or from some intermediate checkpoint.

In online systems, input is usually created dynamically by terminal operators, and arrives in an unpredictable sequence from many different sources. If a failure occurs, it is generally not possible simply to rerun the application, because the content and sequence of the input data is unknown. And, even if it is known, it is usually impractical for operators to reenter a day's work.

Online applications therefore require a system with special mechanisms for recovery and restart which batch systems do not require. These mechanisms ensure that each resource associated with an interrupted online application returns to a known state so that processing can restart safely.

In mixed systems, where both batch and online processing can occur against data at the same time, the recovery requirements for batch processing and online systems are similar.

Recovery requirements in an online system

An online system requires mechanisms that, together with suitable operating procedures, provide **automatic** recovery from failures and allow the system to restart with the minimum of disruption.

The two main recovery requirements of an online system are:

- To maintain the integrity of data
- To minimize the effect of failures

Maintaining the integrity of data

"Data integrity" means that the data is in the form you expect and has not been corrupted. The whole object of recovery operations on files, databases, and similar data resources is to maintain and restore the integrity of the information. Ideally, it should be possible to restore the data to a consistent, known state following any type of failure, with a minimum loss of previous valid updating activity.

Logging changes

One way of doing this is to keep a record, or log, of all the changes made to a resource while the system is executing normally. If a failure occurs, the logged information can help recover the data.

You can use the information in two ways:

1. It can be used to back out incomplete or invalid changes to one or more resources. This is called **backward recovery**, or backout. For backout, it is necessary to record the contents of a data element *before* it is changed. These records are called before-images. In general, backout is applicable to processing failures that prevent one or more transactions (or a batch program) from completing.
2. It can be used to reconstruct changes to a resource, starting with a backup copy of the resource taken earlier. This is called **forward recovery**. For forward recovery, it is necessary to record the contents of a data element *after* it is changed. These records are called after-images.

   In general, forward recovery is applicable to data set failures, or failures in similar data resources, which cause data to become unusable because it has been corrupted or because the physical storage medium has been damaged.

Note: In many cases, a data set failure also causes a processing failure. Then, forward recovery must be followed by backward recovery. If CICS is shut down to perform the forward recovery, a CICS emergency restart performs the backward recovery.

Minimizing the effect of failures

Any online system should limit the effect of any failure. Where possible, a failure that affects only one user, one application, or one data set, should not halt the entire system. Furthermore, if processing for one user is forced to stop prematurely, it must be possible to back out any changes made to any data sets (as if the processing had not started).

If processing for the entire system stops, there may be many users whose updating work is interrupted. On a subsequent startup of the system, only those data set

updates in process (in flight) at the time of failure should be backed out. Backing out only the in-flight updates makes restart quicker, and reduces the amount of data to reenter.

The role of CICS

CICS provides many of the recovery and restart functions needed in an online system.

Automatic backout can be used for most CICS resources (such as databases, files, and auxiliary temporary storage queues), either following a transaction failure or during emergency restart of CICS. CICS also handles all the logging needed for backout. If the backout of a VSAM file fails, CICS backout failure control closes down the base cluster and all affected files. Then, a forward recovery and backout utility can recover the data set offline, and the failed data set can be reset to normal for CICS usage.

CICS message protection performs logging of input and output messages for VTAM® terminals, and enables the messages to be recovered following a system failure.

CICS logs the information required for the forward recovery of DL/I databases (after-images).

VTAM persistent sessions considerations

Persistent session support improves the availability of CICS. It benefits from VTAM 4.2 persistent LU–LU session improvements to provide restart-in-place of a failed CICS without rebinding.

CICS support of persistent sessions includes the support of all LU–LU sessions except LU0 pipeline and LU6.1 sessions. CICS determines for how long the sessions should be retained from the PSDINT system initialization parameter. This is a user-defined time interval. If a failed CICS is restarted within this time, it can use the retained sessions immediately—there is no need for network flows to rebind them. **Note that the "Inter-Enterprise" variant of VSE/VTAM is required for persistent session support**.

You can change the interval using the CEMT or EXEC CICS SET VTAM command, but the changed interval is not stored in the CICS global catalog, and therefore is not restored in an emergency restart.

If CICS is terminated by means of a CEMT or EXEC CICS PERFORM SHUTDOWN IMMEDIATE command or if CICS fails, the CICS sessions are held by VTAM in "recovery pending" state, and may be recovered during startup by a newly starting CICS system.

During emergency restart, CICS restores those sessions pending recovery from the CICS global catalog and the CICS system log to an "in session" state. This happens when CICS opens its VTAM ACB.

Before specific terminal types and levels of service are discussed, note that many factors can affect the performance of a terminal at takeover, including:

- The type of terminal
- The total number of terminals connected
- What the end-user is doing at the time of takeover
- The type of failure of the CICS system
- How the terminal is defined

Subsequent processing is LU dependent: cleanup and recovery for non-LU6 persistent sessions are similar to those for non-LU6 backup sessions under XRF. Cleanup and recovery for LU6.2 persistent sessions maintain the bound session when possible, but there are cases where it is necessary to unbind and rebind the sessions; for example, where CICS fails during a session resynchronization.

The end user of a terminal sees different symptoms of a CICS failure following a restart, depending on whether VTAM persistent sessions are in use:

- If CICS is running without VTAM persistent sessions and fails, the user sees the VTAM logon panel followed by the “good morning” message (if AUTOCONNECT(YES) is specified for the RDO TYPETERM resource definition).
- If CICS does have persistent session support and fails, and the user enters data while CICS is recovering, the user’s perception is that CICS is “hanging”; the screen on display at the time of the failure remains until persistent session recovery is complete. Use of the RDO TYPETERM RECOVOPTION and RECOVNOTIFY keywords allows you to customize the CICS system so that a successful emergency restart can either be transparent to the end user, or the end user can be notified of the CICS failure, allowing the appropriate recovery actions to be taken.

  If APPC sessions are active at the time CICS fails, APPC partners will also perceive the persistent sessions recovery as CICS “hanging”. Requests issued by the APPC partner will be saved by VTAM, and passed to CICS when the persistent recovery is complete. After a successful emergency restart, the options defined in PSRECOVERY of the RDO CONNECTION definition and RECOVOPTION of the RDO SESSIONS definition take effect. If the appropriate recovery options have been selected (see the *CICS Resource Definition Guide*), and the APPC sessions are in the correct state, CICS will perform an ISSUE ABEND (see the *CICS Distributed Transaction Programming Guide*) to inform the partner that the current conversation has been abnormally terminated.

Unbinding sessions

Sessions held by VTAM in a recovery pending state are not always reestablished by CICS. CICS (or VTAM) unbinds recovery pending sessions in the following situations:

- If CICS does not restart within the specified persistent session delay interval
- If a COLD start is performed after a CICS failure
- If CICS restarts with XRF=YES (when the failed CICS was running with XRF=NO)
- If CICS cannot find a terminal control table terminal entry (TCTTE) for a session (for example, because the terminal was autoinstalled with AIRDELAY=0 specified)

- If an RDO TERMINAL or SESSIONs resource definition is defined with the recovery option (RECOVOPTION) set to UNCONDREL or NONE
- If CICS determines that it cannot recover the session without unbinding and rebinding it
- If an RDO CONNECTION resource definition is defined with the persistent session recovery option (PSRECOVERY) set to NONE.

In all these situations, the sessions are unbound, and the result is as if CICS has restarted following a failure without VTAM persistent session support.

There are some other situations where APPC sessions are unbound. For example, if a bind was in progress at the time of the failure, sessions are unbound.

Sessions not retained

There are some circumstances in which VTAM does not retain LU–LU sessions:

- VTAM does not retain sessions after a VTAM, VSE, or processor (CPC) failure
- VTAM does not retain CICS sessions if you close VTAM with any of the following CEMT or EXEC CICS commands:
  - SET VTAM FORCECLOSE
  - SET VTAM IMMCLOSE
  - SET VTAM CLOSED
- VTAM does not retain CICS sessions if you close the CICS node with the VTAM command VARY,NET,INACT ID=applid
- VTAM does not retain CICS sessions if your CICS system performs a normal shutdown (with a PERFORM SHUTDOWN command)

For further information on persistent session support, see the *CICS System Definition Guide*.

Backward recovery (backout)

Backward recovery, or backout, is a way of “undoing” changes made to resources such as files or databases.

Backout is one of the fundamental recovery mechanisms of CICS. It relies on recovery information recorded while CICS and its transactions are running normally.

Recovery information for backout is recorded in the following way. Before a change is made to a resource, a before-image is recorded on both the CICS system log and a dynamic log. A before-image is a record of what the resource was like before the change.

If a transaction fails, information is needed to back out the changes the transaction made while the rest of the CICS system continues normally. This is dynamic transaction backout.

For dynamic transaction backout, CICS writes the information to a dynamic log in main storage. There is one dynamic log for each task.

If the CICS system fails, information is needed to back out the changes made by all tasks that were in-flight at the time of failure. This backout happens during emergency restart.

In readiness for backout during CICS emergency restart, CICS writes recovery information to a journal, the CICS system log.

Recoverable resources

In CICS, a recoverable resource is any resource with recorded recovery information that can be recovered by backout.

The following resources can be made recoverable:

- CICS files that relate to:
  - VSAM data sets
  - DAM data sets
- Data tables
- The CICS system definition (CSD) file
- Intrapartition transient data destinations
- Auxiliary temporary storage queues
- Messages
- Resource definitions dynamically installed using resource definition online (RDO)
- DL/I databases

Logical units of work and synchronization points

When one or more resources are being changed, there comes a point when the changes are “complete” and do not need backout if a failure occurs later.

Logical unit of work

The period between the start of a particular set of changes and the point at which they are complete is called a logical unit of work (LUW). The LUW is a fundamental concept of all CICS backout mechanisms.

From the application designer’s point of view, an LUW is a sequence of actions that needs to be complete before any of the individual actions can be regarded as complete.

For the CICS backout mechanisms, an LUW is simply that part of a transaction’s work that, when complete, is regarded as committed. Committed changes do not have to be backed out if the transaction or the system fails.

Synchronization points

The end of a logical unit of work is indicated to CICS by a synchronization point (usually abbreviated to syncpoint).

A syncpoint arises in the following ways:

- Implicitly at the end of a transaction, signaled by an EXEC CICS RETURN command at the highest logical level. This means that a logical unit of work cannot span tasks.
- Explicitly by EXEC CICS SYNCPOINT commands issued by the application programmer at appropriate points in the transaction.
- Implicitly through a DL/I VSE program specification block (PSB) termination (TERM) call or command. This means that only one DL/I VSE PSB can be scheduled within a logical unit of work.

  Note that an explicit EXEC CICS SYNCPOINT command, or an implicit syncpoint at the end of a task, implies a DL/I PSB termination call.
- Implicitly when a batch DL/I VSE program issues a DL/I VSE checkpoint call. This can occur when the batch DL/I VSE program is sharing a database with CICS applications through multiple partition support (MPS).

It follows from this that an LUW starts:

- At the beginning of a task
- Whenever an implicit or explicit syncpoint is issued and the transaction does not end.

An LUW that does not change a recoverable resource has no meaningful effect for the CICS recovery mechanisms. Nonrecoverable resources are never backed out.

Examples

In Figure 1, task A is a nonconversational (or pseudoconversational) task with one LUW, and task B is a multiple-LUW task (typically a conversational task in which each LUW accepts new data from the user). The figure shows how LUWs end at syncpoints. During the task, the application program can issue syncpoints explicitly, and at the end, CICS issues a syncpoint.

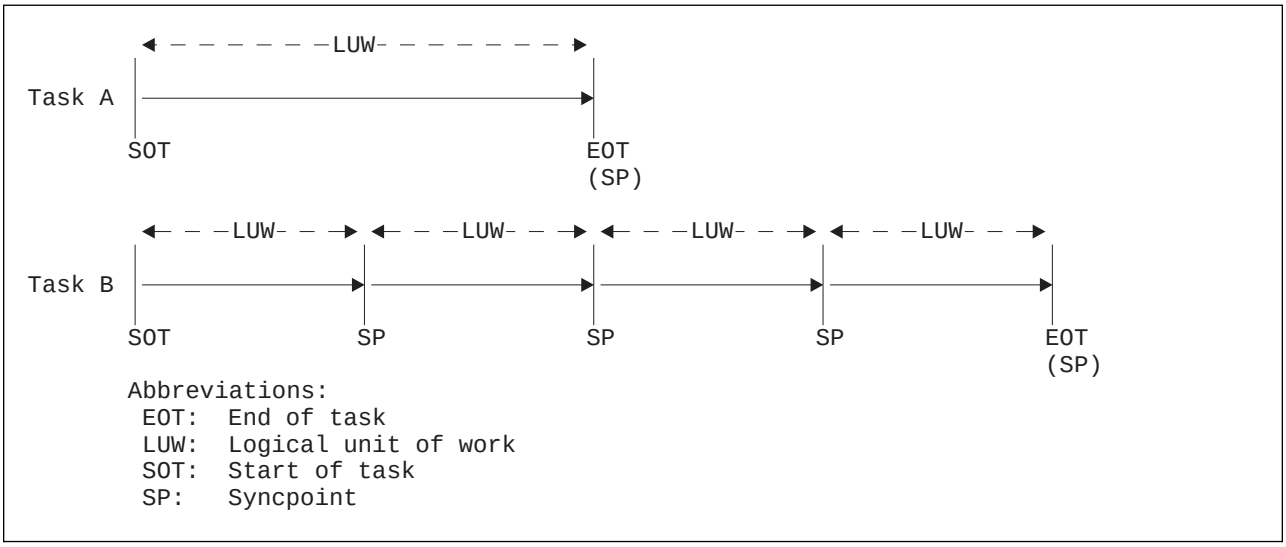

Figure 1. Logical units of work (LUWs) and syncpoints

Figure 2 on page 10 shows that database changes made by a task are not committed until a syncpoint is executed. If task processing is interrupted because of a failure of any kind, changes made within the abending LUW are automatically backed out.

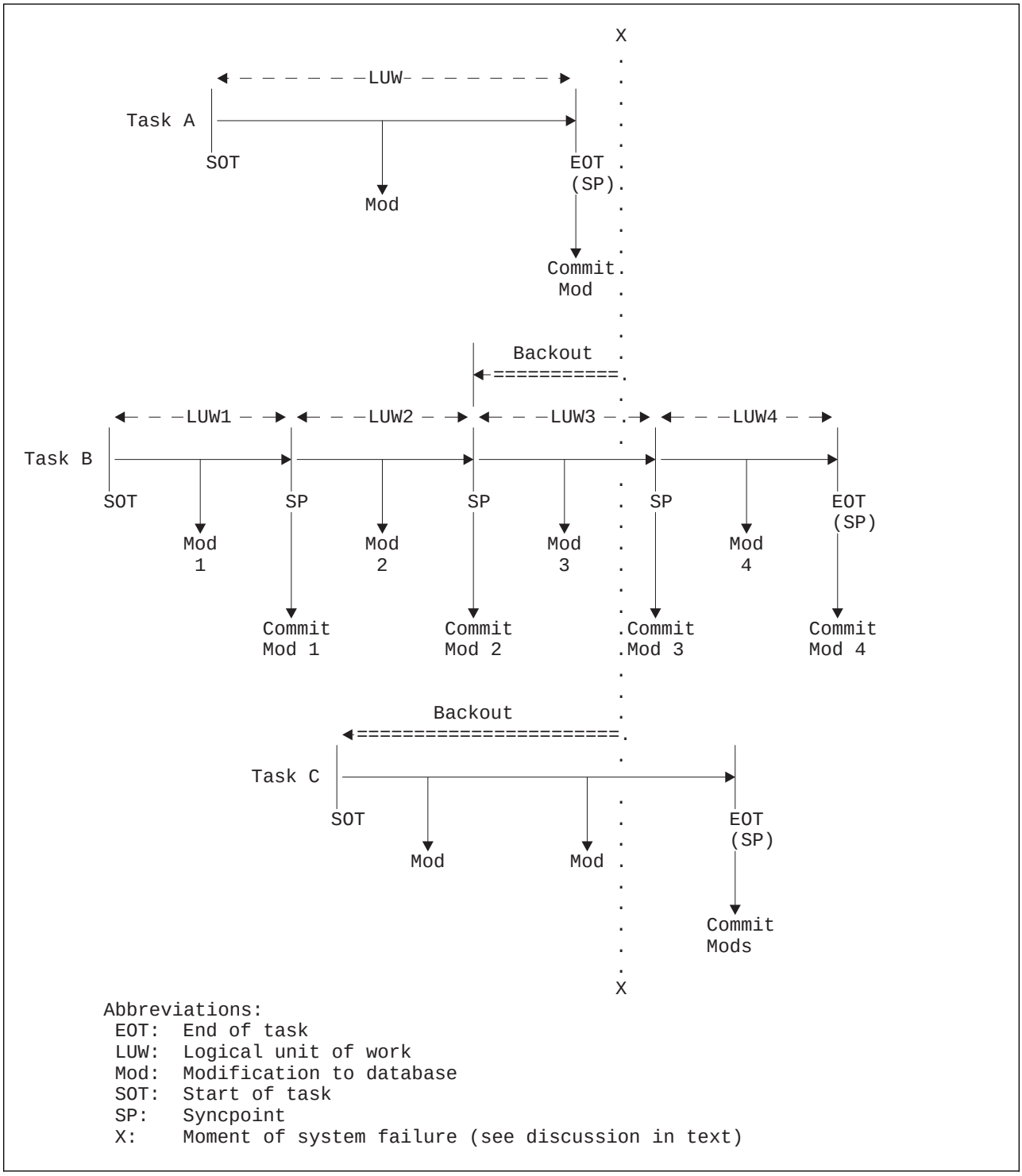

Figure 2. Backout of logical units of work

If there is a system failure at time X:

- The changes made in task A have been committed and are therefore not backed out.
- In task B, the changes shown as Mod 1 and Mod 2 have been committed, but the change shown as Mod 3 is **not** committed and is backed out.
- All the changes made in task C are backed out.

Forward recovery

Some types of data set failure cannot be corrected by backward recovery; for example, failures that cause physical damage to a database or data set. Recovery from failures of this type is usually based on the following actions:

1. Take a backup copy of the data set at regular intervals.
2. Record an after-image of every change to the data set on the system log or any other journal.
3. After the failure, use the information recorded on the system log or other journal to bring the backup copy to the most up-to-date condition possible.

These operations are known as *forward recovery*.

Forward recovery of local DL/I databases

CICS writes after-images of DL/I VSE database changes to the system log. These records are available for forward recovery operations.

Forward recovery of CICS data sets

CICS supports forward recovery of VSAM data sets updated by CICS file control (that is, by files or data tables defined by a CICS RDO FILE definition).

CICS writes the after-images of changes made to a data set on a journal, which can be the system log. You specify the journal number in the file definition. You can define the journal to use automatic archiving, that is, CICS automatically submits a batch job to copy a journal when it is closed. You may then use the archived journals with offline forward-recovery utilities. The file-definition options that are required to implement forward recovery are explained further in "Implementing recoverability of files" on page 74. See Chapter 2, "Recording of recovery information" on page 17 for more information about automatic archiving.

Recovery of VTAM messages

You can nominate transactions that work with VTAM terminals to be *message protected* (see "Specifying message-protection options for VTAM terminals" on page 81). For such transactions, this means that CICS is responsible for logging input and output messages; after a system failure, CICS makes these logged messages available so that application programs can reestablish communication with the terminals.

In addition, for VTAM terminals that support the set-and-test-sequence number (STSN) command, CICS can check SNA sequence numbers after a system failure and retransmit output messages if necessary.

Failures that require CICS recovery processing

The following sections briefly describe CICS recovery processing after:

- Communication failure
- Transaction failure
- System failure

Whenever possible, CICS attempts to contain the effects of a failure—typically by terminating only the offending task while all other tasks continue normally. The updates performed by a prematurely terminated task can be backed out automatically (see “CICS recovery processing following a transaction failure” on page 13).

CICS recovery processing following a communication failure

Causes of communication failure include:

- Terminal failure
- Printer terminal running out of paper
- Power failure at a terminal
- Invalid SNA status

During normal processing, CICS does not store any data to use for recovery from a communication failure. However, for an intersystem communication (ISC) link between CICS and IMS™ or between two CICS systems, CICS stores the inbound and outbound SNA sequence numbers in the relevant TCTTE control block, and on the system log.

If the link fails and is later reestablished, CICS and IMS or CICS and CICS use the SNA set-and-test-sequence numbers (STSN) command to find out what they were doing (backout or commit) at the time of link failure. For further information on link failure, see the *CICS Intercommunication Guide*.

If communication fails, the communication system access method either retries the transmission or notifies CICS after several. attempts. If a retry is successful, CICS is not informed. Information about the error can be recorded by the operating system. If the retries are not successful, CICS is notified.

When CICS detects a communication failure, it gives control to one of two programs:

- The node error program (NEP) for VTAM logical units
- The terminal error program (TEP) for non-VTAM terminals

Both dummy and sample versions of these programs are provided by CICS. The dummy versions do nothing; they simply allow the default actions selected by CICS to proceed. The sample versions show how to write your own NEP or TEP to change the default actions.

The types of processing that might be in a user-written NEP or TEP are:

- Logging additional error information. CICS provides some error information when an error occurs.
- Retrying the transmission. This is not recommended because the access method will already have made several attempts.
- Leaving the terminal out of service. This means that it is unavailable to the terminal operator until the problem is fixed and the terminal is put back into service by means of a master terminal transaction.
- Abending the task if it is still active (see “CICS recovery processing following a transaction failure” on page 13).
- Reducing the amount of error information printed.

"Your own NEP processors" on page 99, and "Your own TEP code" on page 100, have more information about the sample NEPs and TEPs. For programming information about coding your own NEPs and TEPs, see the *CICS Customization Guide*. More general information is in Chapter 6, "Communication error processing" on page 53.

CICS recovery processing following a transaction failure

Causes of a transaction failure include:

- A program check in the application program. CICS intercepts operating system calls for an abend (provided the abend code is included in the system recovery table (SRT)) and, in turn, abends the task.
- An invalid request to CICS from an application, causing an abend.
- A task issuing an ABEND request.
- I/O errors on the data set.

During normal execution of a transaction working with recoverable resources, CICS stores recovery information in a *dynamic log*. If the transaction fails, CICS uses the dynamic log information to back out the changes made by the interrupted LUW. Recoverable resources are thus not left in a partially updated or inconsistent state. Backing out an individual transaction is called **dynamic transaction backout** (DTB).

After DTB has completed, the transaction can restart automatically without the operator being aware of it happening. This function is especially useful in those cases where the cause of transaction failure is temporary and an attempt to rerun the transaction is likely to succeed (for example, DL/I program isolation deadlock). The conditions when a transaction can be automatically restarted are described under "Abnormal termination of a task" on page 47.

If DTB fails, perhaps because of an I/O error on a VSAM data set, CICS backout failure control quiesces all activity on all files referencing data sets that have failed backout. Forward recovery and backout utilities can then recover the data sets offline while CICS remains running.

Chapter 5, "Abend processing" on page 45 gives more details about CICS processing a transaction failure.

CICS recovery processing following a system failure

Causes of a system failure include:

- Processor failure
- Loss of electrical power supply
- Operating system failure
- CICS failure.

During normal execution, CICS stores recovery information on a system log, which can be on disk or tape. After a system failure, CICS is restarted by a special procedure called emergency restart.

During emergency restart, CICS reads the system log backward and extracts information that it places on the restart data set.

CICS then uses the information in the restart data set to:

- Back out recoverable resources
- Recover VTAM messages
- Recover resource definitions installed using the CEDA transaction
- Recover resource definitions installed using EXEC CICS CREATE commands

More details of CICS processing following a system failure are in “Emergency restart” on page 34. You might also review “Forward recovery” on page 11.

Part 2. Recovery and restart processes

This part of the book describes the CICS recovery and restart processes, and indicates where to add user processing to influence these processes. The way you design for, implement, and extend these functions is described in the later parts of this book.

This part contains the following chapters:

For DL/I VSE information, see Chapter 19, “Recovery in a DL/I VSE environment” on page 139.

Chapter 2. Recording of recovery information

This chapter describes where CICS stores information for recovery and restart purposes, including:

When DL/I VSE runs with CICS, all logging for DL/I VSE recovery is directed to the CICS dynamic and system logs. Do not use the batch log that is normally created in DL/I VSE batch processing when running DL/I VSE under CICS.

Recording on the catalogs

CICS uses two catalogs:

- The global catalog (DFHGCD)
- The local catalog (DFHLCD)

The global catalog filename is DFHGCD and the local filename is DFHLCD. In an XRF configuration, the active and alternate CICS each have a local catalog and share the global catalog. The *CICS System Definition Guide* tells you how to create and initialize the CICS catalog data sets.

While CICS is running, the catalogs receive information passed from one execution of CICS, through a shutdown, to the next execution of CICS. This information is not only for warm and emergency restarts, but also for a cold start. If the global catalog fails for any reason, the control record and vital resource information are lost, and it becomes impossible to perform a warm or emergency start.

Take backups of the catalogs periodically (perhaps at the end of each CICS run) to limit the damage that could be caused by a catalog failure during a CICS run.

The next two sections list the types of information recorded on the catalogs.

Global catalog

The global catalog contains information needed at restart, including:

- The control record. After any type of startup, CICS sets an indicator in the control record to "emergency restart needed". If CICS terminates normally, this indicator is changed to "warm start possible". Then, for an automatic start (START=AUTO), if the indicator says "warm start possible", CICS performs a warm start. If the indicator says "emergency restart needed", CICS performs an emergency restart.

  CICS performs a cold start when using the catalog for the first time or, if it is unable to read the catalog.

- Warm keypoint information (described in "Warm keypoints" on page 28).

- Details of the open/closed status of the system log. When CICS terminates, normally or abnormally, it tries to close the system log. If this is successful, the system-log-status indicator is updated.
- Details of the status (ready for use/not ready for use/current) of all data sets in all **disk** journals (including system log) defined without the automatic journal archiving facility.

  This status is retained across a restart, thus maintaining the protection against the reuse of data sets (provided by specifying the PAUSE option in the JCT).

  For journals (including the system log) defined with automatic journal archiving, see “Journal archive control data set” on page 25.
- Resource information. The following information is recorded on the global catalog during CICS execution (see “Recovering dynamically added resource definitions” on page 39), and when CICS is shut down normally (when a warm keypoint is taken):
  - Installed program and transaction resource definitions
  - Installed terminal entries
  - Installed autoinstall terminal models
  - Installed partner definitions
  - The file control table (and, for VSAM data sets that have suffered a backout failure, CICS sets a backout-failed status in a record on the CICS global catalog)
  - DL/I VSE status information
  - Destination control table (intrapartition entries)
  - Dump table information
  - Transient data information
  - Temporary storage information
  - Interval control elements and automatic initiate descriptors at system termination time
  - Unit of recovery descriptors (URDs) at normal shutdown
  - Communications network operating system (CNOS) information during normal CICS operations so that the values can be restored during a persistent sessions restart
- Statistics information, so that restart may restore the same statistics
- Monitoring information, so that the same monitoring options apply at restart

Local catalog

The local catalog contains the essential information for the domains to reinitialize. It also contains the dump data set status record. This records the last dump data set in use. If the DUMPDS=AUTO system initialization parameter has been specified, CICS needs this information at startup to determine which dump data set to open.

Dump options set by CEMT are also recorded, and saved across restarts.

Restart data set

During emergency restart, CICS reads the system log backward and copies selected information to the restart data set. This is the only use of the restart data set. The information is used during emergency restart. You should ensure that the restart data set is large enough to hold all the data copied (see “Recovery control processing” on page 36).

Dynamic log (for dynamic transaction backout)

For resources defined as recoverable, CICS stores a copy of all changes that might be needed for dynamic transaction backout on a dynamic log. To back out the changes made to recoverable resources by a failing transaction, the before-images of such records must be retrieved from the log. The dynamic log is maintained in addition to the system log because the backout data on the system log cannot be read without interfering with other transactions that are writing to it.

Characteristics of a dynamic log

The dynamic log resides in main storage above the 16MB line. The size of the allocation depends on the value specified in the DBUFSZ system initialization parameter and the storage used by previous invocations of the transaction. If the allocation is insufficient, extra storage for spilled dynamic log buffers is allocated above the 16MB line.

Each dynamic log relates to only one transaction. Information that is no longer required is deleted at a syncpoint.

Information recorded in a dynamic log

The information recorded in a dynamic log includes:

- Changes to recoverable files:
  - Before-image of each updated or deleted record
  - Key and data of each new record.
- Changes to DL/I databases:
  - After-image of a database change except for a physical replace record
  - Before-image of a database change
  - KSDS insert log records
- The first VTAM input message for each LUW (for message-protected tasks only)
- The contents of the following areas as they existed at the start of the task (not just the current LUW):
  - The terminal input/output area (TIOA), which contains the initial input that initiated the task
  - The terminal control table user area (TCTUA)
  - The communication area (COMMAREA) as left by a previous task communicating with the same terminal

  These areas are only for transactions that have RESTART(YES) set in the RDO TRANSACTION resource definition.

Note: Even though no information is recorded on the dynamic log for recoverable intrapartition transient data queues or recoverable auxiliary temporary storage queues, these resources and their associated tables can be recovered during dynamic transaction backout. This is because the necessary information is retained in the destination control table, the temporary storage table, and in the queues themselves (see “Dynamic transaction backout (DTB)” on page 47).

System log (journal 1)

The CICS system log is a CICS journal (with a journal identification of 01) that can reside on disk or tape. The following sections describe:

- The information that is recorded on the system log
- The characteristics of the system log on disk
- The characteristics of the system log on tape

The system log is the only place where CICS records backout information for use in emergency restart processing.

Chapter 8, “Logging and journaling” on page 65 tells you how to set up the system log.

Information recorded on the system log

The information recorded on the system log is sufficient to allow backout of changes made to recoverable resources by transactions that were running at the time of failure, and to restore the recoverable part of CICS system tables. Typically, this includes before-images of database records and after-images of recoverable parts of CICS tables—for example, transient data cursors or TCTTE sequence numbers.

In addition, records may be written to the system log (journal 01) by explicit journal requests in the user’s application program; for example, EXEC CICS WRITE JOURNALNUM. You may also choose to place forward recovery information on the system log (see “Defining journals” on page 67).

User-written log records allow you to provide your own recovery process for resources that CICS does not recover itself. The DFHUSBP program, which is invoked during backout, processes these log records and so allows these resources to be backed out (see “XRCINIT exit” on page 92).

CICS also writes “backout-failed” records to the system log (and global catalog) if a failure occurs in backout processing of a VSAM data set during dynamic backout or at emergency restart.

In the event of an uncontrolled termination of CICS, records on the system log are used as input to the emergency restart process as described in “Emergency restart” on page 34.

System activity keypoints

The CICS system log may reside on disk data sets that “wrap around”; that is, when the end of the data set is reached, writing resumes at the start. This means that data does not remain on the system log indefinitely; it will eventually be overwritten. For backout data this is not usually a problem, because the active records should never be older than the longest task that was running at the time of failure. You should take care with exceptionally long-running conversational transactions, however.

On the other hand, the (forward) recovery of CICS tables requires data written by the last **completed** task that changed the table. This data could have been overwritten, but the activity keypointing mechanism prevents its loss by periodically copying the latest committed versions of CICS tables to the system log. In addition, the current tasks are identified in activity keypoints, allowing emergency restart to work out where to stop its backward scan of the system log. Frequently taken activity keypoints can therefore reduce restart time, at the expense of extra processing during normal running.

Frequency of taking activity keypoints: The first activity keypoint of a CICS session is written during system initialization (cold start, warm start, or emergency restart).

The recording of subsequent activity keypoints can be initiated in the following ways:

- By the CSKP transaction, which is attached after every *nn* physical writes to the system log (where *nn* is specified in the AKPFREQ system initialization parameter–for further information, see the *CICS System Definition Guide*).
- Every time logging starts on a new disk data set or tape volume (unless an activity keypoint is already being written).

Characteristics of the system log on disk

The system log can be implemented with **one** disk data set (DFHJ01A) or **two** disk data sets (DFHJ01A and DFHJ01B), as defined by the JTYPE option in the JCT.

A disk system log is designed to wrap around and reuse its data sets if necessary. If only one data set is being used and it becomes full, logging continues at the beginning of the same data set and overwrites information already recorded there. If two data sets are being used, and the data set in use becomes full, logging switches to the beginning of the other data set and overwrites information already recorded there.

Automatic archiving

To ensure that data sets are not overwritten before the contents have been archived for recovery purposes, you may specify automatic archiving of filled data sets with the DFHJCT JOUROPT=AUTOARCH option (for two data sets only). For further information about automatic archiving, see “Preserving the system log (automatic archiving)” on page 65.

An alternative is to use the DFHJCT JOUROPT=PAUSE option, which requests a response from the processor console operator before reusing a data set. This gives the operator a chance to archive the data set (using a batch job) before reusing it. If you use the PAUSE option on a single data set system log, transactions that write to the log must wait while the data set is copied.

The data set used and the position where logging starts when CICS starts depends on whether the system log data sets have been formatted for this CICS run.

Table 2 illustrates where logging starts on two-disk system log data sets that have not been formatted for this CICS run.

Table 3 on page 23 illustrates where logging starts on two-disk system log data sets that have been preformatted by the DFHJCJFP utility, before the start of this CICS run. The *CICS System Definition Guide* tells you more about DFHJCJFP and formatting journals.

Note: You should not format the system log before an emergency restart because you will destroy your recovery data and make restart impossible.

Table 2. Where logging starts on a system log specified with two disk data sets

Type of start	DFHJCT JOUROPT=AUTOARCH	DFHJCT JOUROPT=PAUSE
cold or warm	At start of whichever data set is READY for use. If both are READY, at start of DFHJ01A. If neither is READY, DFHJ01A is requested.	After the last record written to DFHJ01A or DFHJ01B during previous run of CICS. (See also note 2.)
emergency	After last record written to DFHJ01A or DFHJ01B during previous run. (See also note 3.)	

Notes:

1. Journaling will start at the beginning of the next data set if:
   - the last data set used is near the end of volume
   - the data set chosen conflicts with the information on the global catalog
   - the data set is flagged 'not ready for use'

   If you specify JSTATUS=RESET, the status of the journal on the CICS global catalog from the previous run is ignored. In this case, positioning always starts after the last record written, unless this is near the end of volume when the next data set is selected.
2. If the last data set used is near the end of volume, or the data set chosen conflicts with the CICS global catalog, or the data set is flagged 'not ready for use', journaling will start at the beginning of the next data set. However, if you specify JSTATUS=RESET, the status of the journal on the global catalog from the previous run is ignored. In this case, positioning always starts after the last record written, unless this is near the end of volume when the next data set is selected.
3. If you specify neither JOUROPT=AUTOARCH nor JOUROPT=PAUSE, journaling starts in the same place as if you had specified JOUROPT=PAUSE, except that an extent would never be flagged 'not ready for use'.
4. If the previous run of CICS was terminated with an IMMEDIATE shutdown, journal control closes the system log. In this case, an archive request was submitted and positioning is the same as for a cold or warm start.
5. For more information about the JOUROPT options, see page 65.
6. You cannot specify warm or emergency starts. They depend on the START=AUTO system initialization parameter.

Table 3. Where logging starts on a reformatted system log specified with two disk data sets		
Type of start	**DFHJCT JOUROPT=AUTOARCH**	**DFHJCT JOUROPT=PAUSE**
cold or warm	At start of whichever data set is READY for use. If both are READY, at start of DFHJ01A. If neither is READY, DFHJ01A is requested.	At start of DFHJ01A. (See also note 2.)
emergency	Not possible.	Not possible.

Note: If the data set chosen conflicts with the information on the global catalog, or the data set is flagged 'not ready for use', journaling will start at the beginning of the next data set. However, if you specify JSTATUS=RESET, the status of the journal on the CICS global catalog from the previous run is ignored, so positioning always starts at the beginning of DFHJ01A.

Characteristics of the system log on tape

When implemented on tape, the system log consists of a series of tape volumes.

One or two tape drives

CICS supports the use of one or two tape drives. File names are associated with the tape drives as follows:

- For one tape drive, the file name is DFHJ01A. When one tape volume has been filled, another tape volume is mounted and recording continues using the same TLBL name.
- For two tape drives, the CICS TLBL names are DFHJ01A and DFHJ01B. When one tape volume has been filled, journaling continues to the volume on the other tape drive. Thus the series of volumes is recorded by using the two tape drives, and the two file names, alternately.

Journals 2 through 99

Journals 2 through 99 have three purposes: user journaling, automatic journaling, and recording after-images for use with a forward recovery utility:

- User journaling is under your control; it is not used for recovery purposes by CICS.

  You can create user journal records by executing EXEC CICS WRITE JOURNALNUM commands in transactions.

- Automatic journaling means that CICS (on behalf of the user) automatically writes records to any journal, including the system log, as a result of:
  - Records read from or written to files (before-images and after-images).
  - Input or output messages from terminals accessed through VTAM. These are requested by options of an RDO TRANSACTION resource definition. These messages can be used to create audit trails. Remember that syncpoint records are written only to the system log.

  You can request automatic journaling by using options of an RDO FILE resource definition or by using DFHFCT TYPE=FILE macro operands. Automatic journaling is used for user-defined purposes, for example, for an audit trail, or for a forward recovery program. It is not used for CICS recovery purposes.

- CICS records after-images of updates made to CICS files for use with a forward recovery utility.

  You specify which journal is to receive this data by the FWDRECOVLOG option of an RDO FILE resource definition. You can use any journal, including the system log, for this purpose.

Like the system log, you can define user journals for one or two disks or tape. Table 4 indicates where disk journaling (for two data sets) begins for each type of start.

Table 4. Where journaling starts on journals 2 through 99 specified with DISK2

	Type of start	DFHJCT JOUROPT=AUTOARCH	No automatic archiving
Did not format with DFHJCJFP utility before start	cold or warm	At start of whichever data set is READY for use. If both are READY, at start of DFHJnnA. If neither is READY, DFHJnnA is requested.	After the last record written to DFHJnnA or DFHJnnB during the previous run of CICS. (See note 1.)
	emergency	After the last record written to DFHJnnA or DFHJnnB during the previous run of CICS.	
Did reformat with DFHJCJFP utility before start	cold or warm	At start of whichever data set is READY for use. If both are READY, at start of DFHJnnA. If neither is READY, DFHJnnA is requested.	At start of DFHJnnA. (See note 2.)
	emergency	Journaling starts as for cold/warm starts, but formatting means that the data is lost. (See note 3.)	

Notes:

1. Journaling will start at the beginning of the next data set if:

   - The last data set used is near the end of volume.
   - The data set chosen conflicts with the information on the global catalog.
   - The data set is flagged ‘not ready for use’

   However, if you specify JSTATUS=RESET, the status of the journal on the CICS global catalog from the previous run is ignored. In this case, positioning always starts after the last record written, unless this is near the end of volume when the next data set is selected.

2. If the data set chosen conflicts with the information on the CICS global catalog, or the data set is flagged ‘not ready for use’, journaling will start at the beginning of the next data set. However, if you specify JSTATUS=RESET, the status of the journal on the CICS global catalog from the previous run is ignored, so positioning always starts at the beginning of DFHJnnA.

3. If the previous run of CICS was terminated with an IMMEDIATE shutdown, CICS journal control will have closed the user journal. In this case, an archive will have been submitted and positioning is as for a cold or warm start.

Chapter 8, “Logging and journaling” on page 65 provides information about implementing journals.

Journal archive control data set

For journals defined with automatic journal archiving DFHJCT TYPE=ENTRY macro (JOUROPT=AUTOARCH option), details of their status are kept on the journal archive control data set (DFHJACD). This is a VSAM relative record data set (RRDS). For more information about defining the DFHJACD data set, see the *CICS System Definition Guide*.

Chapter 3. CICS shutdown

This chapter describes the various ways CICS can shut down, both normally and abnormally. It also describes the ways that CICS, during shutdown, records information needed for its restart. It covers the following topics:

CICS can stop executing as a result of:

- A normal (controlled) shutdown requested by the master terminal operator
- A normal shutdown requested by an EXEC CICS command in an application program
- Cancelation at the end of emergency restart

CICS can also stop executing in the following (abnormal) ways:

- An immediate shutdown requested by the master terminal operator
- An immediate shutdown requested by an EXEC CICS command in an application program
- A request from the operating system (arising, for example, from a program check or system abend)
- An uncontrolled shutdown (caused, for example, by a machine check or power failure)
- A CICS system module encountering an irrecoverable error
- The START=LOGTERM system initialization parameter

Normal shutdown processing (PERFORM SHUTDOWN)

Normal shutdown is initiated by the master terminal operator, or by an EXEC CICS command in an application program. It takes place in three quiesce stages as described below.

First quiesce stage

During the first quiesce stage of shutdown all terminals are active, all CICS facilities are available, and the following activities are performed concurrently:

- Tasks that already exist complete.
- Tasks that are automatically initiated are run—if they start before the second quiesce stage.
- Any programs listed in the first part of the shutdown program list table (PLT) are run sequentially. (The shutdown PLT suffix is specified in the PLTSD system initialization parameter, which may be overridden by the PLT option of the CEMT or EXEC CICS PERFORM SHUTDOWN command.)

- A new task started as a result of terminal input is allowed to start only if its transaction code is listed in the current transaction list table (XLT) or has been defined as SHUTDOWN(ENABLED) in the transaction definition (RDO). The XLT list of transactions restricts the tasks that can be started by terminals and allows the system to shut down in a controlled manner. The current XLT is the one specified by the XLT=xx system initialization parameter, which may be overridden by the XLT option of the CEMT or EXEC CICS PERFORM SHUTDOWN command.

  Certain CICS-supplied transactions are, however, allowed to start whether their code is listed in the XLT or not. These transactions are CEMT, CESF, CLS1, CLS2, CSAC, CSTE, and CSNE.

The first quiesce stage is complete when the last of the programs listed in the first part of the shutdown PLT has executed and all user tasks are complete.

Note: Long-running tasks (such as conversational tasks) must terminate before CICS shutdown can proceed.

Second quiesce stage

During the second quiesce stage of shutdown:

- Terminals are not active.
- No new tasks are allowed to start.
- Programs listed in the second part of the shutdown PLT (if any) run sequentially. These programs cannot communicate with terminals or make any request that would cause a new task to start.

The second quiesce stage ends when the last of the programs listed in the PLT has completed executing.

Third quiesce stage

During the third quiesce stage of shutdown:

- CICS closes all files that are defined to CICS file control. However, CICS does not catalog the files as UNENABLED; they can then be opened implicitly by the first reference after a subsequent CICS restart.
- CICS writes statistics to the CICS data management facility (DMF).
- CICS writes the following information to the CICS global catalog:
  - A warm keypoint (see “Warm keypoints”).
  - A warm-start-possible indicator. This status applies on the next initialization of CICS if START=AUTO is specified.
- CICS stops executing.

Warm keypoints

The CICS-provided warm keypoint program (DFHWKP) writes a warm keypoint to the CICS global catalog during the third quiesce stage of shutdown processing when all system activity is quiesced. The warm keypoint contains information used to restore the operating environment during a subsequent warm start or emergency restart.

The information listed under “Warm start” on page 32 includes that recorded by a warm keypoint.

Immediate shutdown processing (PERFORM SHUTDOWN IMMEDIATE)

When the master terminal operator or a program requests an immediate shutdown of CICS, processing is different from a normal shutdown in the following important ways:

- User tasks are not guaranteed to complete.
- None of the programs listed in the shutdown PLT is executed.
- CICS does *not* write a warm keypoint or a warm-start-possible indicator to the CICS global catalog.
- CICS does not close files defined to CICS file control.
- Sessions wait for the restart system to initialize or the expiry of the interval specified in the PSDINT system initialization parameter

The next initialization of CICS must be an emergency restart in order to preserve data integrity. An emergency restart is certain if the next initialization of CICS specifies START=AUTO, because an emergency-restart-needed indicator is written to the CICS global catalog whenever CICS is initialized. This indicator remains until the next startup, provided you do not reinitialize the CICS global catalog.

Shutdown requested by the operating system

This type of shutdown can be initiated by the operating system as a result of a program check or an operating system abend. A program check or system abend can cause either an individual transaction to abend or CICS to terminate. (For further details, see “Processing of operating system abends and program checks” on page 51.)

A CICS termination caused by an operating-system request:

- Does not guarantee that user tasks will complete
- Does not allow shutdown PLT programs to execute
- Does not write a warm keypoint or a warm-start-possible indicator to the CICS global catalog
- Takes a system dump as specified by the DUMP system initialization parameter
- Does not close any open files. VSAM files are automatically verified by VSAM on the next open

The next initialization of CICS *must* be an emergency restart in order to preserve data integrity. An emergency restart is certain if the next initialization of CICS specifies START=AUTO, because of the emergency-restart-needed indicator written to the CICS global catalog whenever CICS is initialized.

Uncontrolled termination

An uncontrolled shutdown of CICS can be caused by:

- Power failure
- Machine check
- Operating-system failure

In each case, CICS cannot perform any shutdown processing. In particular, CICS does not write a warm keypoint or a warm-start-possible indicator to the CICS global catalog.

The next initialization of CICS *must* be an emergency restart in order to preserve data integrity. An emergency restart is certain if the next initialization of CICS specifies START=AUTO system initialization parameter, because of the emergency-restart-needed indicator written to the CICS global catalog whenever CICS is initialized.

Printing the dump data set

Most uncontrolled shutdowns will produce a transaction dump. One step of the restart procedure is to print the dump data set. If CICS is initialized using a different dump data set, the print job can be run in parallel with the initialization. If the local catalog stores the name of the dump data set in use when the shutdown occurred, the restart can automatically choose to open a second dump data set.

This is done by specifying the DUMPDS=AUTO system initialization parameter, and defining both dump data sets, DFHDMPA and DFHDMPB, to CICS.

On a warm or emergency start, CICS selects the dump data set that was not in use when the previous CICS run terminated. On a cold start, CICS selects DFHDMPA.

Chapter 4. CICS startup

This chapter describes the CICS startup processing specific to recovery and restart. It is divided into the following sections:

Types of initialization

You can specify any of these system initialization START options:

- START=AUTO, which results in:
  - A warm start, if the previous termination was normal
  - An emergency restart, if the previous termination was not normal
  - A cold start if CICS is running for the first time after initializing the catalogs.

  When START=AUTO is specified, CICS inspects the control record on the CICS global catalog. If it finds an emergency-restart-needed indicator, it performs an emergency restart. If it finds a warm-start-possible indicator, it performs a warm start. If it does not find an indicator (when the CICS catalogs are used for the first time), it performs a cold start.
- START=COLD, which results in a cold start.
- START=STANDBY, for XRF only, which identifies the system as an alternate CICS system. An active CICS system is started, like a non-XRF system, using START=AUTO or COLD.
- START=LOGTERM, which stops CICS at the beginning of emergency restart before backout processing, to allow offline recovery processing.

The use, at restart, of the catalogs, the system log, and user journals is described in Chapter 2, "Recording of recovery information" on page 17.

The CICS initialization process for cold, warm and emergency restarts is described below.

Recovery of system log and user journals

For all types of startup, CICS recovers the status of the system log and user journals as follows:

- For a journal defined to use automatic archiving, CICS recovers the status from the journal archive control data set (DFHJACD). If, for some reason, you want to override the status information, redefine the DFHJACD data set. The *CICS System Definition Guide* tells you how to do this.

- For a disk journal that does not use automatic archiving, CICS recovers the status from the CICS global catalog. To ignore this status information at startup, use the JSTATUS=RESET system initialization parameter.

Cold start

In a cold start, CICS initializes with limited reference to any system activity recorded in the CICS catalogs. A cold start occurs only when the catalogs are newly initialized.

Resource definition information comes from:

- The program library for those tables specified in system initialization parameters (such as DCT=xx).
- The CICS system definition (DFHCSD) file for those resources defined by resource definition online (RDO). The GRPLIST system initialization parameter specifies the particular groups to be used.

Note: If a failure occurs during a cold start, do *not* attempt to do an emergency restart, because the information needed for emergency restart may not have been written to the CICS global catalog. When the cause of the failure has been corrected, initiate another cold start.

User processing can be added to a cold start through the use of programs listed in the program list table (PLT) to run at initialization (see “Using initialization (PLTPI) programs” on page 84).

Note that, on a cold start:

- CICS recovers the status of the system log and user journals (see “Recovery of system log and user journals” on page 31).
- CICS does not use any system log or warm keypoint information from an earlier execution. If you use a cold start after a failure, you might lose data integrity.
- For VSAM data sets that have suffered a backout failure that has not been corrected, the backout-failed status is kept on the CICS global catalog.
- Data on intrapartition transient-data and on auxiliary temporary storage is lost.
- Dump table entries are lost.
- The value of the SVA system initialization parameter is retained on the local catalog across a cold start, unless it is overridden.

Warm start

A warm start restores certain elements of CICS to the status recorded in the warm keypoint of the previous normal shutdown (see “Warm keypoints” on page 28).

In a warm start:

- Resource definition information comes from the program library for those tables specified in system initialization parameters (such as DCT=xx). Resources defined by RDO are restored from the CICS global catalog. The resource information is then updated with information from the warm keypoint.

Between the previous shutdown and the warm start, if you place on the program library new versions of control tables containing attributes of any entry to be warm started, be aware that there might be a conflict between the information in the warm keypoint and that in the control table. This might cause problems later.

- CICS recovers the status of the system log and user journals (see “Recovery of system log and user journals” on page 31).

User processing can be added to a warm start through the use of programs listed in the program list table (PLT) to run at initialization (see “Using initialization (PLTPI) programs” on page 84).

Unless you specify COLD in any of the system initialization parameters that have that option, the following items are warm started—that is, they return to the state they were in at the previous normal shutdown:

- Selected fields from the CSA.
- Intrapartition transient data. At a warm start, destinations may be added, changed, or deleted by changing the DCT load module if the DCT=(xx,COLD) system initialization parameter is coded. You might, however, lose data if you change or delete a destination.
- FCT information. Note that specifying the FCT=xx system initialization parameter has no effect at warm start, because all file definitions are restored from the CICS global catalog.

  If a VSAM data set has suffered a failure during dynamic transaction backout (DTB) or emergency restart, and if the failure has not yet been corrected, the backout-failed status is preserved across a warm start.
- Installed transactions and profiles. Variable information (such as counters and indicators) is reset—except for the enabled/disabled status and the transaction priority, which retain the status recorded in the warm keypoint.
- Installed programs and mapsets. Variable information (such as counters and indicators) is reset—except for the enabled/disabled status, which is restored to the state at the time CICS was shut down.
- Program definitions created by program autoinstall are restored only if they are cataloged. This depends on the autoinstall PGAICTLG system initialization parameter, as follows:

  PGAICTLG=NONE
  autoinstalled program definitions are not cataloged.

  PGAICTLG=MODIFY
  If you code this, or allow it to default, autoinstalled program definitions are cataloged only if the program definition is modified by a SET PROGRAM request subsequent to the autoinstall.

  PGAICTLG=ALL
  Autoinstalled program definitions are written to the global catalog at the time of the autoinstall and following any subsequent modification.
- TCT information using information in the warm keypoint.

  1. Autoinstalled terminal entries are not recovered at warm start except in the following situation. If an autoinstalled terminal is logged off when there is a logoff delay (indicated by the AILDELAY system initialization parameter), it

is possible that the time will not expire before CICS is terminated. If this is the case, and the terminal definition has been cataloged (whenever the AIRDELAY parameter is not zero), the terminal will be recovered at warm or emergency restart, and will be deleted after the period specified by AIRDELAY on the restart JCL. If the JCL used to restart the CICS system specifies AIRDELAY=0, the terminal is recovered, but is deleted as soon as CICS restart is complete.

2. Only the CICS global catalog is referenced for RDO-eligible terminals at a warm start. To add or change a terminal, use RDO. If you want to install and delete terminals, use autoinstall.
3. For terminals not eligible for RDO, to change terminal definitions you must restart CICS with a new terminal control table.
4. Defined APPC connections are warm started. Autoinstalled single-session APPC connections (via CINIT) are subject to the same rules as autoinstalled terminals. Autoinstalled parallel-session APPC connections and single-session APPC connections via a BIND are not warm-started because they are not cataloged.

- Auxiliary temporary storage information. The READ pointers are recovered.
- Control information in the form of interval control elements (ICEs) for outstanding START TRANSID commands and equivalent interval control requests generated internally (by BMS, for example).
- Basic mapping support (BMS) information.
- Details of unit-of-recovery descriptors (URDs) for both external resource managers and APPC conversations.
- Statistics (the collection interval and option, and the logical end-of-day time).
- Monitoring status.
- Dump options set by CEMT or by a program using CICS system programming commands.
- System and transaction dump table entries added by CEMT or by a program using CICS system programming commands.
- The value of the SVA system initialization parameter.

Partial warm start

A partial warm start is similar to a complete warm start, except that some selected CICS facilities are cold-started, as specified in the system initialization parameters. Information comes from the warm keypoint written at the previous normal shutdown, and is applicable only for those facilities that were not specified to be cold-started. The remaining facilities are cold-started.

Emergency restart

Following an abnormal shutdown, an emergency restart returns recoverable resources to their *committed* states; that is:

- Changes to recoverable resources made by logical units of work that were interrupted, are backed out.

For a DL/I VSE database, you do not normally need to run the DLZBACK0 batch backout utility before the emergency restart. But, if backout of the DL/I VSE database fails during emergency restart, any further attempt to perform the backout will also fail unless the batch backout utility has been run before the emergency restart. The *DL/I VSE Resource Definition and Utilities* manual tells you how and when to do it.

If backout for a VSAM data set fails, CICS makes the data set unavailable, and you may run a batch backout utility.

- Messages associated with message-protected tasks are preserved.
- Dynamically added VTAM TCT resource (terminal, typeterm, sessions, and connection) definitions that were committed during execution of a CEDA INSTALL or EXEC CICS CREATE command, are preserved (see "Recovering dynamically added resource definitions" on page 39).

Do not make changes to recoverable resources between the abnormal shutdown and the emergency restart. To do so endangers successful emergency restart processing.

Do not attempt to do an emergency restart if a failure has occurred during a cold start. This is because information needed for emergency restart may not have been written to the CICS global catalog.

Emergency restart processing uses as input the records accumulated on the system log during the previous execution (see "Information recorded on the system log" on page 20). To make emergency restart processing possible, specify a nonzero value for the AKPFREQ system initialization parameter.

During emergency restart, CICS recovers the status of the system log and user journals (see "Recovery of system log and user journals" on page 31).

CICS repositions the latest system log data set. Emergency restart reads the system log backward, and copies to the restart data set the system log records for those LUWs that were processing when the abnormal termination of CICS occurred. (This book normally refers to such tasks as **in-flight tasks** or **in-flight LUWs**.)

CICS backout processing uses the information on the restart data set to remove the effects of data-set modifications made by in-flight tasks. CICS performs initialization, recovery of resource definitions, and then backout processing.

User processing can be added to emergency restart processing in several ways as described in:

- "Using initialization (PLTPI) programs" on page 84
- Chapter 11, "User exits for transaction backout during emergency restart" on page 91.

Resource definition information is obtained from:

- The program library for those tables specified in system initialization parameters (such as DCT=xx). The FCT is an exception, and is not referred to during emergency restart.

  Information about RDO-eligible terminals is taken only from the last warm keypoint (see "Warm keypoints" on page 28). Any terminals installed after

CICS wrote the last warm keypoint will not be recovered. If CICS cannot find a warm keypoint, it installs resources that were active at the last cold start. To make changes to these terminals, always use RDO.

- The CICS global catalog. The CICS global catalog is also used to restore information about statistics gathering and monitoring status, in the same way as for a warm start. Dump options and SVA system initialization parameter status are reapplied from the local catalog.

  The CICS global catalog contains autoinstalled program definitions if the PGAICTLG system initialization parameter has been coded with MODIFY or ALL.

- The system log, activity keypoints, and syncpoint log records for temporary storage and intrapartition transient data.

Recovery control processing

Recovery control reads the system log backward at least as far as the most recent activity keypoint, and copies recovery information to the restart data set. Backout processing uses the information on the restart data set later in the emergency restart process.

The following information is collected:

- Information relating to **in-flight** LUWs and tasks.
- Information relating to **completed** LUWs and tasks, for example:
  - Committed output messages.
  - Tasks that have (1) completed since the last activity keypoint and (2) have the high-order bit set as specified in the JTYPEID operand of an EXEC CICS WRITE JOURNALNUM command (see “User records on the system log” on page 38).
- Information relating to **committed** resource definition changes made using RDO.

When all the above information has been copied from the system log, summary information is recorded on the restart data set, and is available for user-written programs (see Chapter 11, “User exits for transaction backout during emergency restart” on page 91).

Backout processing

After CICS has written the backout information to the restart data set, transaction backout processing can begin. The effects of inflight tasks on the following resources are backed out:

- Recoverable transient data destinations.
- Recoverable temporary storage queues. The READ pointers are set to zero.

  Records older than a specified limit are purged. A parameter (TSAGE) in the temporary storage table (TST) can be used to specify an interval beyond which the queue is to be purged.

  Those start operations that were initiated with data (EXEC CICS START command with FROM, RTRANSID, RTERMID, or QUEUE) are recovered, as

long as you specify a REQID with the same name as a recoverable temporary storage queue.

- Files. File access methods that do not support delete requests (VSAM-ESDS and DAM) are a special case:
  - An application program may choose to delete a record logically by performing a get-for-update followed by a write-update of the same record but with a marked-for-deletion flag.

    Backout processing for such a deletion is exactly the same as for any other updated record. The record marked for deletion is overwritten with the before-image—that is, the same record, but not marked for deletion. (For this reason, these types of data sets must not be reorganized between an abnormal termination and an emergency restart.)
  - To back out a record added to the file, backout processing cannot, on its own, perform the necessary deletion because (1) no delete request is available, and (2) backout processing does not know the user's marked-for-deletion code.

    Therefore, the record must be marked for deletion in a XRCFCER backout exit program, (see "XRCFCER exit" on page 94).

    If no exit program is available, data set integrity for VSAM files is maintained by making the data set unavailable.

  Any alterations made to the data-set name of a file are applied to the installed file definition before transaction backout opens the file. Thus, the data-set name is the same as at the time of the failure, and the file is opened against the correct data set.

  If file backout fails to open a VSAM file for any reason, the operator is prompted to GO or to CANCEL CICS. If GO is specified, backout failure processing takes place. If, however, the file does not open because CICS has already detected a backout failure, there is no operator prompt but the open error exit, XRCOPER, is still taken. CICS flags the backout failure and makes the affected data set unavailable. You may use a batch backout utility to recover the data set offline.

- DL/I VSE databases. If DL/I VSE backout processing fails, the global user exit, XRCDBER, is driven. XRCDBER can return either to ignore the error and continue with the next database, or prompt the operator to GO, or CANCEL CICS.

- Data tables. A CICS-maintained data table has the same recovery/restart properties as the source data set, because CICS always keeps a data table and its source data set in step with each other. If recovery action is required during an emergency restart, the source data set is opened but the loading of the data table is not initiated at that time. This is because there has not yet been any opportunity to activate user exits to control the insertion of entries into the table. CSFU, the system transaction that is responsible for opening files defined with the RDO FILE resource definition option, OPENTIME(STARTUP), or the DFHFCT TYPE=FILE macro operand FILSTAT=OPENED, initiates the loading of any data tables left open after restart recovery.

  In contrast, the recovery attributes of a user-maintained data table and its source data set are independent of each other. Recovery support is provided for user-maintained data tables, but only for dynamic backout. Because no records are written to the system log, there is no recovery at emergency restart.

When a user-maintained data table is opened after a CICS restart, it is loaded with the contents of the source data set. Thus the same recovery support is given whether you specify RECOVERY(ALL) or RECOVERY(BACKOUTONLY) on the RDO FILE resource definition.

- Message-protected tasks. Recovery of message-protected tasks involves reading message texts from the restart data set into message caches for use by user programs. CICS does not read or purge the contents of a message cache.

  A message cache is created only if the task is invoked from a VTAM terminal, under conditions explained in “Interpreting the contents of a message cache” on page 124. A message cache is a temporary storage queue with a DATAID of “DFHMXXXX”, where XXXX is the identification of the logical unit.

- User records on the system log. User-journaled records are written to a journal with the 2-byte JTYPEID set to X'nnFF', where ‘nn’ is a 1-byte function identifier. If this journal is the system log, the records written by LUWs in flight at the time of failure are written to the restart data set. In addition, if the high-order bit of the function identifier byte of JTYPEID is set (JTYPEID=X'80FF', for example), these records are also copied to the restart data set for all tasks completed after the last activity keypoint.

  During emergency restart, the records on the restart data set are processed by the DFHUSBP user backout program. DFHUSBP presents each record to the XRCINPT exit point as it is read from the restart data set. You may add an exit program to recover and process this journaled data. For information about the exit, see “XRCINPT exit” on page 93.

Completion of emergency restart

CICS takes a syncpoint that commits the processing performed during backout. CICS can then continue.

CICS takes an activity keypoint that ensures that there is at least one activity keypoint on the new system log data set. It will show that there are no in-flight tasks, and thus mark the backward scan of the system log on a subsequent emergency restart, in case no other activity keypoint is written during this execution of CICS.

Recovery of specific items

This section describes the recovery at emergency restart of file states, databases, dynamically added resources, and VTAM messages.

Recovering file states

During emergency restart, the state of a file is restored from the global catalog to its state at the time of the shutdown. For example, changes made by EXEC CICS or CEMT SET FILE commands during the last CICS run are restored in the FCT entry.

This applies, in particular, to the ENABLED/DISABLED state and to the SERVREQ options (UPDATE, DELETE...), but does not apply to the opened or closed state.

The file is opened at first reference or after initialization, in accordance with the RDO FILE resource definition option OPENTIME or the DFHFCT macro operand

FILSTAT, regardless of the open or closed state of the file at the end of the last CICS run.

Note: All files defined to CICS file control are closed during a normal shutdown, but they are not defined as UNENABLED in the global catalog. This allows each file to be implicitly opened on the first reference to the file after the CICS restart.

For VSAM files that have suffered a backout failure (during either dynamic transaction backout (DTB) or a previous emergency restart) that has not been corrected, the backout-failed status is carried across an emergency restart (as for other types of start).

Note that the recovery of file states is not synchronized with other recoverable changes in the way that file data recovery is. If a file state change is in-flight at the time of a CICS failure, it is not defined whether the change takes effect or not. There is no backout of in-flight LUWs for file state recovery.

Backout processing for DL/I databases

Changes that were made to databases by inflight LUWs are backed out in the following ways:

- Segments that were updated are overwritten by their before-images.
- Segments that were deleted are added.
- Segments that were added are deleted.

Recovering dynamically added resource definitions

This section describes the mechanism used during emergency restart for recovering resource definitions that were added using the CEDA transaction.

CICS has two ways of installing and committing resource definitions:

- VTAM TCT resource definitions (CONNECTION, SESSIONS, TERMINAL and TYPETERM) are installed in groups and committed at the group level (**group commit**).
- Other resources definitions are installed in groups but committed at the individual resource level (**commit immediate**).

The CICS global catalog keeps a record of the status of the RDO-supported resource definitions. If a CEDA INSTALL for a group of VTAM TCT resources is successful, CICS writes the changed resource definitions to the CICS global catalog during commit processing, when the changes become visible to other CICS tasks.

For resources other than VTAM TCT resources, CICS writes each single resource definition to the CICS global catalog as soon as the corresponding resource is installed. If CICS does not succeed in installing the entire group, it does not back out the individual installed resources. They are, in effect, committed individually.

If CICS fails after this commit processing has completed, it may recover committed resource definitions from the CICS global catalog on a subsequent emergency restart.

If CICS fails before commit processing has started for the group, it will, on a subsequent emergency restart, recover any resources (except VTAM TCT

resources) from the CICS global catalog. CICS will back out any VTAM TCT resources in the uncommitted group install.

Committing the changes to VTAM TCT resources at the group level requires the install process to write the definitions to the system log so that CICS can complete an in-flight commit at emergency restart. Because CICS commits all other RDO resources immediately during install, it does not need to write these to the system log.

Before committing changes to VTAM TCT resource definitions, CICS writes the changed resource definitions to the system log.

If CICS fails while commit processing is taking place, the system log contains VTAM TCT resource definitions that are to be committed, but are not on the CICS global catalog. Other resources in the group that were installed before CICS failed are on the CICS global catalog.

During the subsequent emergency restart, the resource manager creates its set of resource definitions from the CICS global catalog. The resource manager then asks recovery control to pass it the VTAM TCT resource definitions logged during the CEDA INSTALL where commit processing started but did not complete. The resource manager reinstalls such definitions, making them visible to the CICS system, and writes to the CICS global catalog the definitions read from the system log.

After installing each resource, CICS sends a message to the CSDL log. If, after an emergency restart, you are in any doubt about the state of a resource, you should install the whole group again.

Recovering autoinstalled terminals: Autoinstalled terminal entries are recovered at an emergency restart, but not at a warm start. After a delay period (the default is seven minutes) specified by the AIRDELAY system initialization parameter, any autoinstalled terminal that was recovered but is not in session again is deleted. The terminal is deleted even if it has outstanding work scheduled, such as an AID (automatic initiate descriptor).

AIRDELAY=0 means that autoinstalled terminals are not written to the CICS global catalog and are therefore not recovered—this applies to terminals and APPC single session via a CINIT. Also, autoinstalled single sessions via a BIND and parallel sessions are not recovered.

If you code AUTOCONNECT=YES as an autoinstalled terminal model, terminals using such a model establish sessions as soon as CICS takes control. They are not deleted after the delay period. You should take care when you select terminals with an AUTOCONNECT=YES model. Such a terminal might be autoconnected and in session after an emergency restart, and the terminal user might not be present. This could considerably impair your virtual storage saving.

Recovering autoinstalled programs: See page 33 for information about when autoinstalled programs are cataloged in the CICS global catalog.

Recovering program definitions: Program definitions created by program autoinstall are restored only if they are cataloged. This depends on the autoinstall PGAICTLG system initialization parameter.

Recovering dynamic changes to transient data queue attributes

During normal operation, CICS allows you to change specific attributes of DCT resources. You make such changes by using the CEMT INQUIRE TDQUEUE or CEMT SET TDQUEUE command. The *CICS-Supplied Transactions* manual tells you how to use these transactions.

When you perform an emergency restart, CICS restores changes made to DCT entries for recoverable transient data queues. The attributes restored for each DCT entry are the automatic transaction initiation (ATI) trigger level, terminal identifier, and transaction identifier.

Resynchronization and re-presentation of VTAM messages

When LU-LU sessions are reestablished after an emergency restart, CICS participates in a resynchronization protocol with logical units to find out if any messages, in either direction, were lost when CICS terminated. Lost messages are retransmitted either by the LU or by CICS from a resend slot in temporary storage. Resend slots are deleted when the temporary storage is cold started, or at the next emergency restart if it is not recoverable, or when a program deletes the temporary storage.

The logical units that require resynchronization are marked in the terminal control table terminal entries (TCTTEs) during backout processing. Resynchronization is not attempted if:

- The terminal is acquired with COLDACQ specified.
- The session is a pipeline session.
- The TCTTE is marked to cold start the session by the TCT assembly process. This is done for terminals, such as 3270 terminals, that do not support the set and test sequence number (STSN) command.[1]

If the previous session abended, do not use COLDACQ, because this overrides CICS integrity control, and could lead to data integrity problems. Also, check the CSMT log for an activity keypoint after the restart of a session following a CICS failure. If there is no activity keypoint, issue COLDACQ again after the next emergency restart.

Comparison of the types of restart

Table 5 compares aspects of the three types of restart. Note that you do not specify warm and emergency starts; they come from the START=AUTO system initialization parameter. For clarity, the figure does not compare aspects of resource definition. That comparison is in Table 6 on page 42.

[1] Further information on STSN commands can be found in the appropriate CICS subsystem guides.

Table 5. Comparison of types of CICS restart

	Cold Start	Warm Start	Emergency Restart
Information from system log of previous run?	Not used	Not used	Used
Auxiliary temporary storage retained?	No	Yes—all data	Yes (assuming queue names recoverable)
Intrapartition transient data destination retained?	No	Yes—all data	Yes (assuming destinations logically or physically recoverable)
Backout performed?	No	No	Yes
Message recovery?	No	No	Yes
User control blocks reinitialized? (TCTUA, Comm. Area, CWA).	No	No	No
Post-initialization PLT processing possible?	Yes	Yes	Yes

Table 6. Sources of resource definition information for different types of start

Source of resource definition (RD) information:	**Cold Start**	**Warm Start**	**Emergency Restart**
RD information in all tables referenced by system initialization parameters	Obtained from program library	Obtained from program library	Obtained from program library
RD information contained in warm keypoint of previous run	Not used	Used to update RD information from program library	Not available
RD information in the groups in the list(s) named by the GRPLIST system initialization parameter for THIS initialization	Taken from CICS system definition file (CSD) and merged with information from the program library. See Note 1.	Not used	Not used
RD information in the groups in the list(s) named by the GRPLIST system initialization parameter for the PREVIOUS initialization	Not applicable	Obtained from CICS global catalog	Obtained from CICS global catalog
RD information in groups that have been INSTALLed since the last cold start	Not applicable	Obtained from CICS global catalog	Obtained from CICS global catalog (and system log for VTAM TCT resources)
Autoinstalled terminals	Not applicable	CICS global catalog if AID outstanding	CICS global catalog
Autoinstalled programs	Not applicable	Obtained from CICS global catalog. See Note 2.	Obtained from CICS global catalog. See Note 2.

Notes:

1. For more information about the CSD, see the *CICS Resource Definition Guide*.
2. In the case of autoinstalled programs, these may or may not have been recorded on the CICS global catalog depending on the PGAICTLG system initialization parameter specified on the *previous* run of CICS.

User programs at initialization

After any type of startup (cold, warm, or emergency), and before CICS finally takes control, any programs listed to run at initialization execute sequentially. You list these programs in the program list table (PLT), defined by the PLTPI system initialization parameter.

Following execution of the initialization programs, CICS takes a syncpoint that commits changes made to recoverable resources and releases enqueues on them.

For more information about PLT programs, see “Using initialization (PLTPI) programs” on page 84.

Chapter 5. Abend processing

This chapter describes abend processing under the following headings:

- "Requests for an abend"
- "Transaction abend processing"
- "Processing of operating system abends and program checks" on page 51.

Requests for an abend

The following events can request CICS to abend a transaction:

- A transaction ABEND request issued by a CICS management module
- An EXEC CICS ABEND request issued by a user program
- Certain commands issued from the master terminal, such as CEMT SET TASK PURGE or FORCEPURGE
- Certain commands issued from an application program, such as EXEC CICS SET TASK PURGE or FORCEPURGE
- A transaction abend request issued by DFHZNEP or DFHTEP following a communication error

Transaction abend processing

If, during transaction abend processing, another abend occurs and CICS continues, there is a risk of a transaction abend loop and further processing of a resource that has lost integrity (because of uncompleted recovery). If CICS detects that this is the case, the CICS system abends with message DFHPC0402, DFHPC0405, DFHPC0408, or DFHPC0409.

How CICS handles transaction abends

The action taken by CICS on the abend exit code can:

- Terminate the task **normally**
- Terminate the task **abnormally**.

"Abnormal termination of a task" on page 47 describes the processing that may follow the abnormal termination of a task.

Exit code

Exit code can be written either in **programs** (separate modules defined by CEDA DEFINE PROGRAM commands) or **routines** within the application program. Exit code, if activated, can gain control when a task abend occurs.

Exit code can be activated, deactivated, or reactivated by EXEC CICS HANDLE ABEND commands; for programming information on these, see the *CICS Application Programming Reference* manual.

Only one abend exit can be active at any given logical level within a task. This means that:

1. When one application program uses the LINK command to pass control to another program, the program sending control and the program receiving control can each have one active exit.
2. When an exit is activated (at a particular program level), any other exit already active at the same level becomes deactivated automatically.

Reasons that an application programmer might have for coding a program level abend exit, and functions that might be incorporated, are discussed in "Handling abends and program level abend exits" on page 111.

When an abend request is issued for a task, CICS immediately passes control to the exit that is active at the current logical level[2]:

- If no exit is active at the current logical level, CICS checks progressively up through higher logical levels and passes control to the first exit code found to be active.
- If CICS finds no active exit at, or higher than, the current logical level, the task terminates abnormally (see "Abnormal termination of a task" on page 47).

When control is transferred to any exit code, CICS deactivates the exit before any of its code is executed. (This means that, in the event of another abend request, the exit will not be reentered, and control is passed to activated exit code (if any) at the next higher level.)

The exit code then executes as an extension of the abending task, and runs at the same level as the program that issued the EXEC CICS HANDLE ABEND command that activated the exit.

After any program level abend exit code has been executed, the next action depends on how the exit code ends:

- If the exit code ends with an EXEC CICS ABEND command, CICS gives control to the next higher level exit code that is active. If no exit code is active at higher logical levels, CICS terminates the task **abnormally**. The next section describes what may happen after abnormal termination of a task.
- If the exit code ends with an EXEC CICS RETURN command, CICS returns control to the next higher logical level at the point following the EXEC CICS LINK command (not to any exit code that may be active) just as if the EXEC CICS RETURN had been issued by the lower level application program. This leaves the task in a normal processing state and it does not terminate at this point.

  In the special case of an EXEC CICS RETURN command being issued by exit code at the highest logical level, CICS regains control and terminates the task **normally**. This means that:

  1. Dynamic transaction backout is **not** performed.
  2. An end-of-task syncpoint record is written to the system log.

[2] The program receiving control is said to be at a *lower logical level* than the program that issues the LINK command. The concept of logical levels is explained in the *CICS Application Programming Guide*.

Note: If a transaction updates recoverable resources and, therefore, requires dynamic transaction backout to be performed in the event of a task abend, the exit code *must* end with an EXEC CICS ABEND command.

Abnormal termination of a task

If the exit code ends with an ABEND command, abnormal termination of a task starts after all active program-level abend exits (if any) have executed. The sequence of actions during abnormal termination of a task depends on the following factors:

- Code in the transaction restart program (DFHREST)
- The transaction has freed the principal facility
- Backout is successful.

Transaction restart

The transaction restart user-replaceable program (DFHREST) enables you to participate in the decision as to whether a transaction should be restarted or not.

For programming information about how to provide your own code for DFHREST, see the *CICS Customization Guide*.

Notes:

1. CICS invokes DFHREST only when RESTART(YES) is specified in a transaction's resource definition.
2. When transaction restart occurs, a new task is attached that invokes the initial program of the transaction. This is true even if the task abended in the second or subsequent LUW, and DFHREST requested a restart.
3. Statistics on the total number of restarts against each transaction are kept.
4. Emergency restart does not restart any tasks.
5. Making a transaction restartable involves slightly more overhead than dynamic transaction backout because more items are logged; such items are logged only on the dynamic log.
6. In some cases, the benefits of transaction restart can be obtained instead by using the EXEC CICS SYNCPOINT ROLLBACK command. Although use of the ROLLBACK command is not usually recommended, it does keep all the executable code in the application programs (except for DFHDBP exit code). For more information about the use of the ROLLBACK option when working in an ISC or MRO environment, see the *CICS Intercommunication Guide*.

Dynamic transaction backout (DTB)

Assuming that the resources affected by the abending task are recoverable, CICS performs dynamic transaction backout (DTB).

DTB backs out the effects of a transaction that terminates abnormally. The resources specified as recoverable are restored to the state they were in at the beginning of the interrupted LUW (that is, at the most recent synchronization point or start of task). The resources are thus restored to a consistent state.

DTB is similar in effect to the backout of in-flight tasks during emergency restart (following a CICS failure). The most important differences are that DTB operates on a single abnormally terminating transaction and that the backout is carried out

online (that is, while the rest of the CICS system continues to run normally). DTB thus provides immediate recovery of data integrity following a transaction failure.

User exits are provided for errors (see “Global user exits in DFHDBP” on page 88).

To restore the resources to the state they were in at the beginning of the LUW, a description of their state at that time must be preserved. For tables maintained by CICS (the destination control table and the temporary storage unit table), information is held in the tables themselves. For transient data and auxiliary temporary storage, deleted records or the before-images of records that have changed are saved on the transient data or temporary storage data sets themselves. For DL/I VSE databases or CICS files, the before-images of deleted or changed records are recorded on a **dynamic log** (described in “Dynamic log (for dynamic transaction backout)” on page 19). The first input messages from message-protected VTAM terminals are also held on this log.

DTB backs out changes made by the abending transaction to the following resources:

CICS files
In the special case of the file access methods that do not support delete requests (VSAM-ESDS and DAM), records to be deleted should be marked for deletion in an XDBFERR exit program (see “Global user exits in DFHDBP” on page 88). (Such records can be truly deleted when the data set is subsequently reorganized offline by a user-supplied utility.) If you do not have an exit program, backout failure processing is entered.

If backout of a VSAM file fails, CICS:

- Notes the backout-failed status in the base cluster block
- Logs a backout-failed record in the CICS system log
- Sets a backout-failed status in the CICS global catalog
- Closes the FCT entries open against the base cluster, to prevent further updates on the damaged data set.

CICS then informs the operator of the status of the data set, and a batch backout utility may be run using the information provided by CICS, a copy of the data set restored from the backup copy, and archived logs. For more information about running batch backout, see Chapter 16, “Backout failure” on page 129.

DL/I VSE databases
If DL/I VSE backout processing fails, all potentially affected databases are stopped to preserve data integrity, but CICS continues to run.

Intrapartition transient data (logical recovery only)
Intrapartition destinations specified as **logically** recoverable are restored by DTB.

Physical recovery, which may be specified for emergency restart, is not part of DTB. This means that:

- Any records retrieved by the abended LUW are not available to be read by another task, and are therefore lost.

- Any records written by the abending LUW are not backed out. This means that these records **are** available to be read by other tasks, although they might be invalid.

Recovery of extrapartition queues is not supported.

Auxiliary temporary storage

DTB recovers temporary storage data written to or released from auxiliary storage. It does not recover temporary storage data in main storage.

Terminal messages

For message-protected tasks, the transmission of any deferred output messages, which would normally occur after syncpoint processing, is suppressed by DTB. The first input message after the last synchronization point is recovered from the dynamic log and presented to the XDBIN exit of DFHDBP.

EXEC CICS START requests

Recovery of START requests during DTB depends on whether the following operands are coded with the START request:

- The PROTECT operand (which ensures that the new task cannot START execution until the START-issuing task has passed its next syncpoint)
- The FROM and LENGTH operands (which pass data through temporary storage to the STARTed task).

Recovery of START requests during DTB is described below for different combinations of these operands on a START request that has already been issued.

Simple START request (without PROTECT, FROM, and LENGTH operands)

DTB has no effect; the new task starts at its specified time (and may already be executing when the START-issuing task backs out). Abending the START-issuing task does not abend the started task.

START request with PROTECT (but without the FROM and LENGTH operands)

DTB of the START-issuing task cancels the START request. The new task will not have started yet because the START-issuing task being backed out will not have reached the syncpoint.

START request that passes data to the new task by means of the FROM and LENGTH operands (but without the PROTECT operand)

Assuming that the temporary storage queue used for START request data is designated as recoverable by a DFHTST TYPE=RECOVERY macro, DTB of the task also backs out the data being transferred to the new task. The new task still starts at its specified time, but the data is not available to the started task and will therefore raise a NOTFND condition.

START request with PROTECT, FROM and LENGTH operands

DTB of the START-issuing task backs out the data being transferred to the new task (assuming temporary storage is designated as recoverable) and cancels the START request. The new task therefore never gets started.

Note: Recovery of temporary storage (whether or not PROTECT is specified) does not cause dynamic restart of the new task. (It may qualify for restart like any other task, if RESTART(YES) is coded on the RDO TRANSACTION resource definition.) On emergency restart, the START

command restarts only tasks started with data written to a *recoverable* temporary storage queue.

Basic mapping support (BMS) messages

DTB recovery of BMS messages affects those BMS operations that store data on temporary storage. They are:

- BMS commands that specify the PAGING operand
- The BMS ROUTE command
- The message switching transaction (CMSG).

Backout of these BMS operations is based on backing out START requests (because, internally, BMS uses the START mechanism to implement the operations listed above). You request backout of these operations by marking the temporary storage DATAIDs that carry the messages as recoverable in the DATAID operand of the DFHTST TYPE=RECOVERY macro. For more information about this operand, see the *CICS Resource Definition Guide*.

Application programmers can override the default temporary storage DATAIDs by specifying the following operands:

- REQID operand in the EXEC CICS SEND MAP command
- REQID operand in the EXEC CICS SEND TEXT command
- REQID operand in the EXEC CICS ROUTE command
- PROTECT operand in the CMSG transaction.

Note: If DTB fails, restart is not attempted regardless of the setting of the restart program.

Actions taken at abnormal task termination

The CICS abnormal condition program is invoked during abnormal task termination unless the task is to be restarted.

The principal action of this program is to send, if possible, an abend message to the terminal connected to the abending transaction. It also sends a message to the master terminal destination.

Before sending the message to the master terminal, the abnormal condition program links to the user-replaceable program error program (DFHPEP). DFHPEP is given control through a LINK from the CICS abnormal condition program. This occurs after all program-level abend exit code has been executed by the task that abnormally terminates, and after dynamic transaction backout (if any) has been performed.

Notes:

1. DFHPEP is not given control when the task abend is part of the processing done by CICS to avoid a system stall.
2. DFHPEP processing takes place after a transaction dump has been taken. DFHPEP cannot prevent the taking of a dump.
3. DFHPEP is not given control when the task is terminated because of an attach failure. Examples are when the transaction does not exist or when a security violation is detected.

The CICS-provided DFHPEP program executes no functions, but you can include in it your own code to carry out installation-level action following a transaction abend

(see “Program error program (DFHPEP)” on page 121). There is only one program error program for the whole system.

All CICS facilities are available to the DFHPEP program. You can:

- Send messages to the terminal
- Send messages to the master terminal
- Record information or statistics about the abend
- Request the disabling of the transaction entry associated with this task.

Processing of operating system abends and program checks

There is a limit to the processing you can attempt after an operating-system abend or a program check.

If the abend is associated with any domain other than the application domain, there is no further user involvement in processing the error.

If the abend is in the application domain, one of the following can occur:

- CICS terminates (see “Shutdown requested by the operating system” on page 29).
- CICS remains operational, but the CICS task currently in control can terminate.

If a program check occurs when a user task is processing, the task abends with an abend code of ASRA. If a program check occurs when a CICS system task is processing, CICS terminates.

If an operating-system abend has occurred, processing continues by searching the system recovery table, DFHSRT. The SRT is a table containing a set of operating-system abend codes that you want CICS to recover from. CICS searches the SRT looking for the system abend code issued by the system.

- If a match *is not* found, CICS is terminated.
- If a match *is* found, and a CICS system task is processing, CICS is terminated.
- If a match **is** found, and a user task is processing, the default action is to abend the task with an abend code of ASRB. However, you can change this action by coding a global user exit program at exit point XSRAB. The value of the return code from XSRAB determines which of the following happens next:
  - The task terminates with the ASRB abend code.
  - The task terminates with the ASRB abend code and CICS cancels any program-level abend exits that are active for the task.
  - CICS terminates.

  For programming information about the XSRAB exit point, see the *CICS Customization Guide*.

CICS supplies an SRT that has a default set of abend codes; and you can add to, delete from, or modify the default list of abend codes. For more information about the SRT, see the *CICS Resource Definition Guide*.

Note: Because it is possible to introduce recursions between program checks and abends, take great care when coding a global user exit program at the XSRAB exit point.

Chapter 6. Communication error processing

This chapter describes the main CICS programs that participate in communication error processing:

- Node error program (DFHZNEP)
- Terminal error program (DFHTEP).

CICS controls terminals by using VTAM (in conjunction with NCP for remote terminals). These communication access methods detect transmission errors between the central processing complex (CPC) and a remote terminal, and automatically invoke error recovery procedures, if specified. These error recovery procedures generally involve:

- Retransmission of data a defined number of times or until data is transmitted error-free.
- Recording of information about the error on a data set or internally in control blocks. You can, at times, access data recorded in control blocks using communication system commands.

If the data is not transmitted successfully after the specified number of retries:

- CICS terminal management is notified.
- One of the following CICS terminal error transactions is initiated:
  - Control can pass to a node error program (DFHZNEP) provided by yourself.
  - Control can pass to a terminal error program (DFHTEP) provided by yourself.

Chapter 12, “Handling communication errors” on page 97 is a starting point for coding your own error programs.

Node error program (DFHZNEP)

You can specify your own processing for VTAM errors in a node error program (NEP). You can use the sample NEP supplied, change the sample, or write your own.

The NEP is entered once for each terminal error; therefore it should be designed to process only one error for each invocation. (The types of processing that might be done are discussed in “Your own NEP processors” on page 99.)

In some circumstances, VTAM communication system errors can be passed to an application program. If you issue an EXEC CICS HANDLE command with the TERMERR condition specified, the application program can decide on the action to take in response to the error condition. The TERMERR condition is raised if the DFHZNEP program, (if you have one), schedules an ABTASK action (ATNI abend) for a terminal error while the task is attached.

Note: The TERMERR is raised for the current or next terminal control request. If the task is executing normally and performing non-terminal operations when the VTAM network error occurs, the task is unaware of the error and continues processing until it attempts the next terminal control request. It is at this point that

the task receives the TERMERR. If the task does not issue any further terminal-type request it will not receive the TERMERR or ABEND.

Terminal error program (DFHTEP)

You can specify your own processing for non-VTAM communication errors in a terminal error program (TEP). You can use the sample TEP supplied with CICS (DFHXTEP), change the sample, or write your own.

The TEP is entered once for each terminal error and therefore should be designed to process only one error for each invocation.

The in-doubt window

When different CICS systems are connected by MRO or across an ISC (LU6.1 or APPC) link, tasks can communicate across the connection and can update resources in a logically interdependent way. If the connection or either system fails between syncpoints, both systems can back out any updates of recoverable resources either dynamically or on emergency restart.

If a failure occurs during the syncpointing process, the situation is less clear. For an interval of time called the in-doubt window, neither system “knows” if the other has committed its updates and, therefore, whether it should commit its own. The possibility of failure during the in-doubt window should be taken into account when designing applications.

The processing of a distributed syncpoint involves a complicated set of flows and protocols. Different concepts are involved in MRO, LU6.1, and APPC syncpointing. See the *CICS Intercommunication Guide* for descriptions of each of these.

The processing between two syncpoints is called a logical unit of work (LUW) and is identified by a unique identifier. This identifier is written to the system log by each task when the task makes its first change to a recoverable resource. It is also included in any of the messages generated in diagnosing a failure during the in-doubt window. A user-written log-scanning utility can read all log records for the LUW in the affected CICS regions, and determine what action is needed to bring the databases into synchronization. Programming information about this is given in the *CICS Customization Guide*.

Part 3. Implementing your recovery and restart strategy

This part describes the way you implement your recovery and restart strategy.

It contains these chapters:

Chapter 7. Starting to specify recovery and restart facilities

This chapter describes how to specify the basic CICS recovery facilities in the following topics:

In addition to the information in other parts of this book, for reference information about resource definition, see the *CICS Resource Definition Guide*, and for further information about system initialization parameters, see the *CICS System Definition Guide*.

Questions relating to recovery requirements

For ease of presentation, the following questions assume a single application.

Note: If a new application is added to an existing system, the effects of the addition on the whole system need to be considered.

Question 1: Does the application update data in the system? If the application is to perform **no** updating (that is, it is an inquiry-only application), recovery and restart functions are not needed within CICS. (But you should take backup copies of non-updated data sets in case they become unreadable.) The following questions assume that the application does perform updates.

Question 2: Will this application be used concurrently by more than one user? If two or more users are to run this application concurrently, you must take special steps to avoid interference between multiple executions of the application.

Question 3: Does this application update data sets that other online applications access? If yes, does the business require updates to be made **online**, and then to be immediately available to other applications—that is, as soon as the application has made them? This could be a requirement in an online order entry system where it is vital for inventory data sets[3] to be as up-to-date as possible for use by other applications at all times.

Alternatively, can updates be stored temporarily and used to update the data set(s) later—perhaps using offline batch programs? This might be acceptable for an application that records only data not needed immediately by other applications.

[3] In the context of these questions, the term "data sets" includes databases.

Question 4: Does this application update data sets that batch applications access? If yes, establish whether the batch applications are to access the data sets concurrently with the online applications. (If accesses made by the batch applications are limited to read-only, it is possible for the data sets to be shared between online and batch applications, although read integrity may not be guaranteed. If you intend to update data sets concurrently from both online and batch applications, you may wish to consider using DL/I, which ensures both read and write integrity.)

Question 5: Does the application access any confidential data? Files that contain confidential data, and the applications that have access to those files, must be clearly identified at this stage. You may need to ensure that only authorized users may access confidential data when service is resumed after a failure, by asking for reidentification in a sign-on message.

Question 6: If a data set becomes unusable, should all applications be terminated while recovery is performed? If degraded service to any application has to be preserved while recovery of the data set takes place, include procedures to do this.

Question 7: Which of the files to be updated are to be regarded as vital files? Identify any files that are so vital to the business that they must always be recoverable.

Question 8: How long can the business tolerate being unable to use the application in the event of a failure? Indicate (approximately) the maximum time that the business can allow the system to be out of service after a failure. Is it minutes or hours? The time allowed may have to be negotiated according to the types of failure and the ways in which the business can continue without the online application.

Question 9: How is the user to continue or restart entering data after a failure? This is an important part of a recovery requirements statement because it can affect the amount of programming required. The user's restart procedure will depend largely on what is feasible—for example:

- Is it necessary for the user to continue business by other means—for example, manually?
- Does the user still have source material (papers, documents) that allow the continued entry (or reentry) of data? If the source material is transitory (received over the telephone, for example), this will require slightly more complex procedures.
- Even if the user does still have the source material, does the quantity of data preclude its reentry?

Such factors define the point where the user restarts work. This could be at a point that is as close as possible to the point reached before the system failure (which might be implemented with the aid of a progress transaction[4]). Or it could be at some point earlier in the application—even at the start of the transaction.

These considerations should be in the external design statement.

---

4 A progress transaction here means one that enables users to determine the last actions performed by the application on their behalf.

Question 10: During what periods of the day are online applications expected to be available? This is an important consideration when applications (online and batch) require so much of the available computer time that difficulties can arise in scheduling precautionary work for recovery (taking backup copies, for example). See “Daily and weekly schedules” on page 131.

Validate the recovery requirements statement

After considering the above questions, produce a formal statement of application and recovery requirements. Before any design or programming work begins, all interested parties should agree on the statement—including:

- Those responsible for business management
- Those responsible for data management
- Those who are to use the application—including the end users, and those responsible for computer and online system operation.

Designing the end user’s restart procedure

Decide how the user is to restart work on the application after a system failure. Points to consider are:

- The need for users to reidentify themselves to the system in a signon message (dictated by security requirements, as discussed under “Question 5: Does the application access any confidential data?” on page 58).
- The availability of appropriate information for users, so that they know what work has and has not been done. Consider the possibility of a progress transaction (as discussed under Progress transaction on page 101).
- How much or how little rekeying will be needed when resuming work (dictated by the feasibility of rekeying data, as discussed under “Question 9: How is the user to continue or restart entering data after a failure?” on page 58).

The design of the user’s restart procedure (including the progress transaction, if used) should include precautions to ensure that each input data item is processed once only.

End user’s standby procedures

Decide how application work might continue in the event of a prolonged failure of the system. For example, for an order-entry application, it might be practical (for a limited time) to continue taking orders offline—by pencil-and-paper methods. If such an approach is planned, you need to specify how the offline data is to be subsequently entered into the system; it may be necessary to provide a catch-up function.

Note: If the user is working with a terminal attached to a programmable controller, it may be possible to continue gathering data without access to the central processing complex.

Communications between application and user

For each application, specify what type of terminal the user is to work with.

Decide if special procedures are to be provided to overcome communication problems; for example:

- Allow the users to continue work on an alternative terminal (but with appropriate security precautions, such as signing on again).
- In cases where the user's terminal is attached to a programmable controller, determine what recovery actions that controller (or the program in it) is capable of providing.
- If a user's printer becomes unusable (because of hardware or communication problems), consider the use of alternatives, such as the computer center's printer, as a standby.

This information is needed in internal design when considering the handling of communication breaks (see "Handling communication breaks" on page 98).

Security

Decide the security procedures for an emergency restart or a break in communications. For example, when confidential data is at risk, specify that the users should sign on again and have their passwords rechecked.

Bear in mind the security requirements when a user needs to use an alternative terminal if a failure is confined to one terminal (or to a few terminals).

Note: The signon state of a user is not retained after a VTAM persistent sessions restart.

Definitions for recovery functions

In the next few pages, you can find information about the definitions that form the basis of a system that uses recovery and restart functions. The information is a starting point, so that you know what to look for in the appropriate book in the CICS library.

Basic file definition

The file definitions needed for backout and forward recovery are described in "Implementing recoverability of files" on page 74.

System recovery table (SRT)

The basic DFHSRT entry (DFHSRT TYPE=INITIAL, SUFFIX=xx) causes CICS to intercept certain operating system abend codes and to attempt recovery. Use of an SRT also causes CICS to attempt recovery from program checks. If you want to intercept additional operating system abends, or abend codes, you must code DFHSRT TYPE=SYSTEM|USER macros.

For a brief overview of the system recovery program and table, see "Processing of operating system abends and program checks" on page 51. That chapter provides further references.

Definitions for transactions and programs

You use resource definition online (RDO) to define and install transactions, profiles, programs, and mapsets. Installing the following groups provides basic recovery functions:

DFHAKP
DFHBACK
DFHJRNL
DFHRSEND
DFHRSPLG
DFHSTAND
DFHVTAM

Note that backout occurs for all transactions.

For **file DL/I VSE recovery**, you should install the DFHAKP, DFHBACK, and DFHJRNL groups. You should also take the following options of an RDO TRANSACTION resource definition into account when defining user transactions that will update files and DL/I VSE databases:

RESTART
This option defines whether CICS will consider restarting a transaction. ("Editing the transaction restart program (DFHREST)" on page 89 tells you more about replacing the default DFHREST program.)

DTIMOUT
If the task remains suspended (inactive) for the specified interval, CICS initiates an abnormal termination of the task. CICS does not perform an abnormal termination if:

- DTIMOUT(NO) is specified.
- The task is currently not system-purgeable (SPURGE=NO).
- The task is not in a state suitable for an abnormal termination.

SPURGE
Indicates whether the transaction is initially system-purgeable. That is, can CICS purge the transaction as a result of the deadlock timeout facility (DTIMOUT), EXEC CICS TASK(id) PURGE command, or CEMT SET TASK(id) PURGE command? For more information about options on the RDO TRANSACTION definition, see the *CICS Resource Definition Guide*.

If you specify a transaction as system-purgeable, and backout is attempted, the backout might not complete successfully because of a lack of resources. For this reason, DFHDBP is defined in the DFHBACK group as being resident to avoid errors of not having enough storage to load the program.

For terminal error handling, you should install the DFHSTAND (needed by the terminal abnormal-condition handling program) and DFHVTAM (needed by the VTAM abnormal-condition program) groups. You should also consider the NEPCLASS option of the RDO PROFILE resource definition, described under "Your own NEP processors" on page 99. If you are interested in message protection for VTAM terminals, see "Specifying message-protection options for VTAM terminals" on page 81.

To define individual programs required for recovery and restart, you need an RDO PROGRAM resource definition for:

- Each user exit program
- Each replaceable program, for example DFHREST and DFHPEP
- Each program list table and each PLT program
- Any program that you want to override the automatically-generated version.

Note: User exit programs, replaceable programs, and PLT programs can be autoinstalled.

Definition of the system log and other journals

This is a basic definition of the system log using two disk data sets:

```
DFHJCT TYPE=ENTRY
       ,JFILEID=SYSTEM
       ,JOUROPT=(CRUCIAL,RETRY,AUTOARCH)
       ,ARCHJCL=DFH$ARCH
       ,JTYPE=DISK2
       ,BUFSIZE=nnnnn
3      ,DEVADDR=(SYSnnn,SYSnnn)
```

5 For a user journal with two disk data sets:

```
DFHJCT TYPE=ENTRY
       ,JFILEID={2-99}
       ,JOUROPT=(CRUCIAL,RETRY,AUTOARCH)
       ,ARCHJCL=DFH$ARCH
       ,JTYPE=DISK2
       ,BUFSIZE=nnnnn
3      ,DEVADDR=(SYSnnn,SYSnnn)
```

For further information, see Chapter 8, “Logging and journaling” on page 65.

System initialization parameters

The following list summarizes the system initialization parameters that you need to consider for recovery and restart. For more information about the options, see the *CICS System Definition Guide*.

```
AILDELAY={0-hhmmss}
AIRDELAY={700-hhmmss}
AKPFREQ={200-65535|0}
APPLID=({DBDCCICS|name1}[,name2])
CSDFRLOG={1-99}
CSDRECOV={NONE|ALL|BACKOUTONLY}
DBP={YES|xx}
DBUFSZ={500|number}
3 {DLI|DL1}=({NO|YES|xx}[,COLD])
FCT={YES|xx|NO}
JCT={YES|xx|NO}
JSTATUS=RESET
NEWSIT={YES|NO}
PGAICTLG={MODIFY|NONE|ALL}
PGAIEXIT={DFHPGADX|name}
PGAIPGM={INACTIVE|ACTIVE}
PLTPI={YES|xx|NO}
PSDINT={0-hhmmss}
SRT={YES|xx|NO}
START={AUTO|(AUTO,ALL)|COLD|(COLD,ALL)|LOGTERM|STANDBY}
SYSIDNT={CICS|name}
TBEXITS=([name1],[name2],[name3],[name4])
```

Activity keypoints must be taken to make emergency restart possible. Therefore, you should specify a nonzero value for AKPFREQ (the default is 200.)

If you code NEWSIT=YES at a warm start, the values in the SIT take effect, and there is no reference to the warm keypoint information that has previously been stored for the SIT.

Destination control table (DCT)

Use the DESTRCV={LG|PH} operand of the DFHDCT TYPE=INTRA macro for each intrapartition destination that you want to be recoverable. See the *CICS Resource Definition Guide* for information on which destinations must be recoverable.

Program list table (PLT)

You use the DFHPLT macro to name each program executed during initialization or controlled shutdown of CICS. See the *CICS Resource Definition Guide* for information on the names of each program during initialization or controlled shutdown.

Temporary storage table (TST)

When you define your temporary storage with DFHTST macros, note that TSAGE and DATAID operands influence the recovery characteristics of that temporary storage.

Transaction list table (XLT)

Use the DFHXLT macro to name the transactions that can be initiated from a terminal during the first quiesce stage of normal shutdown. See also the SHUTDOWN attribute on the RDO TRANSACTION resource definition.

Documentation and test plans

During internal design, consider how to document and test the defined recovery and restart programs, exits, and procedures.

Recovery and restart programs and procedures usually relate to exceptional conditions, and can therefore be more difficult to test than those that handle normal conditions. They should, nevertheless, be tested as far as possible, to ensure that they handle the functions they are designed for.

CICS facilities, such as the execution diagnostic facility (CEDF) and command interpreter (CECI), can assist in causing exception conditions and interpreting program and system reactions to those conditions.

The ability of the installed CICS system, application programs, operators, and terminal users to cope with exception conditions depends on the designer and the implementer being able to:

- Forecast the exceptional conditions that can be expected
- Document what operators and users should do in the process of recovery, and include escape procedures for problems or errors that persist.

Conditions that need documented procedures include:

- Power failure of the processor
- Failure of CICS
- Physical failure of data set(s)
- Transaction abends
- Communication failures—such as the loss of telephone lines or a printer being out of service.

Note: It is essential that recovery and restart procedures are tested and rehearsed in a controlled environment by all personnel who might have to cope with a failure. This is especially important in installations that have temporary operators.

Chapter 8. Logging and journaling

This chapter tells you how to implement the system log and journals on disk and tape. The use of journals for forward recovery, keypointing, the dynamic log, and the catalogs are discussed in the following sections:

System log

You define the system log using the DFHJCT TYPE=ENTRY, JFILEID=SYSTEM macro, which is described briefly on page 62 and more fully in the *CICS Resource Definition Guide*.

Implementing the system log on disk

The system log can be implemented on disk on one data set (JTYPE=DISK1 in the JCT, where the filename is DFHJ01A), or on two datasets (JTYPE=DISK2 in the JCT, where the file names are DFHJ01A and DFHJ01B).

One or two data sets?

You are recommended to use *two* disk data sets of equal size, and specify either automatic archiving, or the PAUSE option. In this way, online tasks need not be delayed, because one data set can be archived while the other is in use. Note that two disk data sets do not carry out dual logging. Information is logged to only one data set at a time.

If you use only *one* disk data set for the system log and it becomes full, online tasks may have to wait while the data set is archived to tape. You can avoid this problem by ensuring that the data set has enough space for the maximum amount of logging activity in one CICS session.

If you use two data sets, make both data sets large enough to contain the longest logical unit of work (LUW) (allow a safety margin). This is sufficient to enable backout of in-flight LUWs during emergency restart.

By using two data sets, you can also cater for errors such as an I/O error on the data set in use.

Preserving the system log (automatic archiving)

If you want to preserve the log (or any other journal) for forward recovery, batch backout utilities, audit trail, analysis, or other purposes, use the automatic archiving option (JOUROPT=AUTOARCH) in the DFHJCT TYPE=ENTRY macro. This simplifies the operation of logging, offers greater security, and reduces the delays caused by archiving just before you use a utility.

Automatic archiving is a more secure method of retaining log records than:

- Coding JOUROPT=PAUSE in the DFHJCT TYPE=ENTRY macro, to give the operator time to ensure that the other data set has been archived to tape by an offline procedure
- Using the user-replaceable DFHXJCO and DFHXJCC modules for controlled log archiving.

Whenever a journal data set is closed for output, a VSE archiving job is created. The job is submitted for execution to POWER. CICS cannot reuse the data set until the archive job has completed.

The journal archive control data set (DFHJACD) controls the submission of archive jobs and the reuse of journal data sets. The DFHJACD also contains the current status of journal data sets.

The *CICS Operations and Utilities Guide* describes the use of the DFHJACDU utility to determine the status of a log. You can also use CEMT or EXEC CICS commands to inquire about the status of the data sets. If the journals are switching when you use CEMT, an appropriate message is displayed.

The process of extracting and preserving forward recovery information from the system log needs tight controls. If emergency restart has backed out local DL/I database changes on DFHJ01A, that data set will be needed for forward recovery of those changes in addition to updates to VSAM files.

Implementing the system log on tape

The system log can be implemented on tape using one tape drive (where the file name is DFHJ01A), or two tape drives (where the file names are DFHJ01A and DFHJ01B).

One or two tape drives?

If you use only one tape drive for the system log and the tape becomes full, online tasks must wait while the tape rewinds and a new tape is mounted. You can avoid this problem by using two tape drives.

Note: CICS Transaction Server for VSE/ESA™ does not use any of the facilities provided by standard label tapes, and therefore does not control the use of the tapes for the system log, or any other journal. The operations staff must control the use of tape volumes manually.

Journals for forward recovery

For forward recovery, you can journal after-images to the system log (JOURNALID=1) or to any user journal (JOURNALID=2 through 99). For ease of administration, use the system log to reduce the number of online journals and archived copies. For speed of recovery, direct after-images for particular data sets to separate user journals; this enables a forward recovery utility to find the relevant information more quickly. For the definition of automatic journaling of after-images, see “Implementing recoverability of files” on page 74. (For DL/I VSE forward recovery, after-images are written only to the system log.)

If you choose to implement your own forward recovery strategy, you must provide procedures to extract and preserve forward recovery information either:

- From a completed journal or system log, before it is overwritten or preformatted for the next session; or
- From a copy of the journal or system log.

Note: As long as a disk journal is needed for a possible forward recovery, it should be archived before it is overwritten. Automatic archiving is the most efficient way to archive journals.

Defining journals

Use the DFHJCT TYPE=ENTRY macro to define user journals. This is similar to defining the system log.

Instead of specifying JFILEID=SYSTEM, you specify **JFILEID=nn** (where nn is in the range 2 through 99) to identify the journal. When you define a file as forward recoverable, you specify the number of the journal where after-images for forward recovery are recorded using the FWDRECOVLOG option of the RDO FILE resource definition. Likewise, an EXEC CICS WRITE JOURNALNUM command in an application program must specify the journal number.

You may also specify deferred opening of a journal (but not if you are using journal archiving), as described in “Deferred opening of journals” on page 69.

The positioning within a journal data set at startup, when two disk data sets are used, is explained in Table 4 on page 24.

Keypointing

The AKPFREQ system initialization parameter specifies the number of consecutive write operations that CICS makes to the system log between activity keypoints. Set the AKPFREQ value so that at least three activity keypoints are taken per disk log data set—more on tape log data sets.

Do not set AKPFREQ to zero— otherwise emergency restart will be impossible. The AKPFREQ value should not be greater than 2000—otherwise the time taken by an emergency restart might be excessive.

You can use the XAKUSER global user exit if you need to recover data not normally recovered by CICS itself (such as the common work area (CWA)). The exit would usually be associated with journaling, post-initialization program(s), and the XRCINPT transaction backout exit.

Using XAKUSER, you can record your own data as part of the periodic system activity keypoint data sent to the system log during normal CICS operation (see “System activity keypoints” on page 21). Whenever a system activity keypoint is written to the system log, the XAKUSER global user exit is invoked. The exit program can record application-dependent information on the system log, using the EXEC CICS WRITE JOURNALNUM(1) command.

At emergency restart, log records written by the exit program are presented to the XRCINPT global user exit. Only records written during the last complete activity keypoint of the current CICS execution are presented. Those written during uncompleted or earlier activity keypoints are not presented.

For programming information about these global user exits, see the *CICS Customization Guide*.

Dynamic log

The journal control program places dynamic log records in the dynamic buffer above the 16MB line. If that buffer becomes full, the overflow records are also placed above the 16MB line (see “Dynamic log (for dynamic transaction backout)” on page 19).

The DBUFSZ (dynamic buffer size) system initialization parameter influences the initial maximum size of the dynamic log buffer area by means of an algorithm. Choose the allocation for each transaction. If the value specified for DBUFSZ is too small, this may impair performance by forcing the overflow mechanism to be used too often. A value that is too large may allow excessive use of virtual storage by some transactions. For further information about the effects of DBUFSZ, see the *CICS Performance Guide*.

Explicit journaling

You can use using explicit journal commands (as opposed to system logging, or automatic journaling requested through file definition options). Explicit journaling is available to application programs to support requirements such as:

- Recording information for an audit trail
- Recording recovery-and restart-related information for resources not protected by CICS, such as:
  - Common work area (CWA) or tables in main storage
  - Extrapartition transient data
  - Messages from non-VTAM terminals.
- Support for your own recovery functions, such as forward recovery routines.

Explicit journal commands

Explicit journal commands (EXEC CICS WRITE JOURNALNUM and EXEC CICS WAIT JOURNALNUM) can be used to direct output to the system log (journal 1) or to any other journal. If you direct output to a journal other than the system log, note that:

- The records are not available during emergency restart except by using postinitialization (PLTPI) programs (see “Using initialization (PLTPI) programs” on page 84 for further information).
- If the transaction abends, you might need to use a user exit in the dynamic backout program (DFHDBP) to write journal records to reverse the effects of those written by the failed LUW.

Journal commands can cause immediate or deferred output to the journal; the identification of the journal must be specified, and a journal type identifier can be given to distinguish journal record types. If you write a journal record to the system log, the journal record type identifier (according to the setting of the high-order bit) also causes recovery control to copy the records to the restart data set during its backward scan of the log:

- For in-flight tasks only (high-order bit off)
- For all records encountered until the scan terminates (high-order bit on).

Programming information on the commands for explicit journaling (EXEC CICS WRITE JOURNALNUM and EXEC CICS WAIT JOURNALNUM) is in the *CICS Application Programming Reference* manual.

Note: You can use CEMT INQUIRE and SET JOURNALNUM or EXEC CICS INQUIRE and SET JOURNALNUM commands to display the status of the current data set and, if defined, the alternate (secondary) data sets. If the journal is switching when CEMT is used, an appropriate message is given. For information about CEMT commands, see the *CICS-Supplied Transactions* manual; for programming information about equivalent EXEC CICS commands, see the *CICS System Programming Reference* manual.

Defining journals

Define each journal in the JCT with a DFHJCT macro. You can use the OPEN option to specify when to open the journal:

- By CICS during system initialization
- Deferred until an explicit OPEN request is made.

The latter case is discussed in "Deferred opening of journals." For more information on the DFHJCT macro, see the *CICS Resource Definition Guide*.

Deferred opening of journals

You can specify deferred opening for any journal except the system log or journals specified with automatic archiving, by coding OPEN=DEFERRED in the DFHJCT TYPE=ENTRY macro.

Possible reasons for taking this option are security and resource use:

- For security reasons, you may not want to enable certain transactions outside specified hours. You do not, therefore, need to open an associated journal until the transactions are enabled.
- From a resource viewpoint, if a tape journal is not always needed, it makes sense not to mount it until necessary. This frees one or two tape drives for other uses.

Reading journal data sets offline

If you are designing your own recovery systems (for forward recovery, for example), you will need to write offline programs to read journal data sets. CICS can help you do this; for programming information about journaling, see the *CICS Customization Guide*.

Processing of journaled information at emergency restart

The journaled records and the activity keypoint records are presented at the XRCINPT exit of DFHUSBP during emergency restart.

Chapter 9. Recovering resources

This chapter describes in the following sections, data design considerations and the recoverability of resources:

Recovery of DL/I VSE resources is described in Chapter 19, "Recovery in a DL/I VSE environment" on page 139.

Protecting data files and databases

A CICS *file* is a logical view of a physical data set, defined to CICS in the file control table (FCT) with an 7-character file name. A CICS file is associated with a VSAM or DAM data set by one of the following:

- The DSNAME parameter in the RDO FILE resource definition
- The DSNAME parameter of an EXEC CICS CREATE FILE command
- A CEMT SET FILE DSNAME(name) command
- An EXEC CICS SET FILE DSNAME(name) command
- A DLBL statement specifying a DSNAME

More than one file can refer to the same data set.

A *data set* is defined to be the physical object residing on DASD. It has a 44-character DSNAME. A VSAM data set, for example, is defined using VSE/VSAM IDCAMS utility. For more information, see the *VSE/VSAM Commands* manual.

Data design

The main concern in data design is to ensure that, whatever the access method for the system's databases, they are protected from corruption and can recover from accidental damage.

Unless you use existing databases, you must select the access method for each database; what you select might well depend on the recovery and restart factors described below.

VSAM files

Recovery and restart factors, which vary according to the choice of access method, are discussed below in relation to:

VSAM Key-sequenced data sets (KSDS)
VSAM Relative record data sets (RRDS)
VSAM Entry-sequences data sets (ESDS)

Sharing data sets: Sharing data sets between online CICS update transactions and batch update programs using VSAM share options (where available) or job control sharing is **not** recommended. It introduces the risk that the data sets will be

logically damaged and that application programs will not function correctly. Such damage can occur, for example, if a CICS LUW updates a record that is later updated by a non-CICS job while the CICS LUW is still running. If the CICS LUW abends, dynamic transaction backout (DTB) backs out the record to the value it had at the start of the CICS LUW, destroying the update from the non-CICS job.

Forward recovery: For VSAM files, you can use a forward recovery and batch backout utility when online backout processing has failed. For forward recovery, you need to:

- Create backup copies of data sets
- Record after-images of file changes (see "Implementing recoverability of files" on page 74)
- Archive filled journal data sets, to preserve records that might be necessary for forward recovery
- Prepare the job to run a forward recovery utility, and keep control of backup data sets and journals that might be needed as input.

Backward recovery: To ensure that VSAM files can be backward recoverable, certain points should be considered:

- Key-sequenced data sets (VSAM-KSDS) and relative record data sets (VSAM-RRDS):
  - If the files referring to VSAM-KSDS or RRDS data sets are designated as recoverable, dynamic transaction backout and transaction backout during emergency restart can back out any updates, additions, and deletions made by an interrupted LUW.
  - For errors that can occur during backout, see Chapter 11, "User exits for transaction backout during emergency restart" on page 91 and "Global user exits in DFHDBP" on page 88.
- Entry-sequenced data sets (VSAM-ESDS):
  - New records are added to the end of a VSAM-ESDS. After they have been added, a record cannot be physically deleted. A logical deletion can be made only by modifying data in the record; for example, by flagging the record with a "logically deleted" flag.
  - As described on page 37, backout (performed during emergency restart or by DTB) operates on files referring to VSAM-ESDS data sets thus:
    - Each record that was updated (including a flagged deletion) is restored in place to its before-image; flagged deletions are reversed.
    - Records that were added to the file cannot be deleted by CICS. Such records must be either detected and ignored, or flag-deleted by code in exits available in DFHDBP and the transaction backout programs (see "Global user exits in DFHDBP" on page 88 and Chapter 11, "User exits for transaction backout during emergency restart" on page 91.)
- For all types of VSAM data set:
  - A backout utility enables you to run backout offline against files where normal backout procedures have failed. It uses:
    - The data set containing the uncommitted updates that could not be backed out

- Before-images from the archived system log(s)
- A user-supplied job to run the utility, with the failed data set and the archived log(s) as input.

Direct access method (DAM)

For DAM files, there is no support for forward recovery via the DFHFCT macro operands. You can implement your own forward recovery support using automatic journaling options.

Backout for DAM data sets is the same as for ESDS data sets in that you cannot delete records from the data set (see the previous section).

Presenting large quantities of data

Decide how to present and access large quantities of data. Possibilities include:

- Selection of particular elements of data
- Scrolling on a video display
- Displaying on a printer
- Paging to a video display.

This information is needed for internal design purposes (see “Implications of presenting large amounts of data to the user” on page 108).

Access to data by two or more users

Decide, for each data resource, whether it is possible for two or more users to access the data concurrently. If several users need frequent update access to the same data resource (such as a record that keeps a running total):

- Task deadlock is possible, and must be catered for by the internal design. (This is one of the factors to consider when choosing file access methods; see “Data design” on page 71.)
- Response times may be longer than desirable because all the tasks will be enqueuing on the one resource.
- Multiple path updating of VSAM files can cause forward recovery problems (see “Implementing forward recovery with existing utilities” on page 78).

If these characteristics are recognized in the external design, applications can be designed to avoid multiple tasks depending on access to one resource.

Protecting files against processing failure

Decide which files to protect—that is, which CICS files refer to data sets that need to be backed out if an updating task is interrupted. Generally, **all** files should be candidates for backward recovery. Making read-only files recoverable does not incur any overhead.

Protecting against data set failure

Decide the procedures for taking backup copies of data sets and for recording changed records so that forward recovery is possible in the event of a data set becoming unusable.

VSAM files may be taken offline for backup. Recovery is always performed offline.

Physical damage to disk or tape occurs infrequently, but it must be considered. Identify the data sets that need to be backed up, and the journals that need to be journaled and archived.

How often you take backup copies in readiness for forward recovery depends on the importance of restart speed (see “Question 8: How long can the business tolerate being unable to use the application in the event of a failure?” on page 58). Backup copies may be taken, for example:

- Before processing each set of batch updates. During batch updating of VSAM files CICS takes no record of the updates made, so you should consider taking a backup copy before and after the batch run. If the batch processing fails, the backup provides a clean base either for the batch updates to be run again, or for CICS processing.
- Before or after each CICS session.
- Once a day.
- Once a week.
- Once a month.

For successful forward recovery, it is necessary to have procedures that are **clearly documented** and **well tested,** and which the operations staff can use without having to consult the data management staff.

Decide which data sets are critical for the business and therefore require special recovery precautions so that they can be quickly recovered in the event of physical damage. To protect critical data sets, consider:

- Recording recovery information **in duplicate** on different journals. The amount of programming to do this should be balanced against the business risks involved.
- Taking duplicate backup copies of key data sets at intervals and storing them off-site. Note that the CICS catalogs and the CICS system definition file (which is treated like any other CICS file) are also vital to your CICS system, and you should consider how to safeguard against their failure.

Implementing recoverability of files

This section describes how to define the recovery characteristics of files using the CEDA transaction.

Defining files

With the CEDA DEFINE FILE command, you can specify support for both forward and backward recovery. The necessary parameters are RECOVERY and FWDRECOVLOG. A CEDA command to support a batch backout and forward recovery utility is:

```
CEDA DEFINE FILE(name) GROUP(groupname)
     DSNAME(data-set name)
       .
       .
     RECOVERY(ALL)
     FWDRECOVLOG(number)
       .
       .
```

Notes:

1. RECOVERY(ALL) means that before-images for updates made to this file are recorded on the system log (journal 01), and after-images are recorded on the journal specified by FWDRECOVLOG.
2. RECOVERY(ALL), plus FWDRECOVLOG, provides forward recovery support for VSAM files. Note that FWDRECOVLOG contains journal records incompatible with previous releases of CICS, as follows:
   - WRITE_ADD_COMPLETE, written when a record is added to the file. It is journaled **after** the I/O operation.
   - WRITE_DELETE, written to the FWDRECOVLOG when a record is deleted from the VSAM file.
   - WRITE_UPDATE, written to the FWDRECOVLOG when a record in the VSAM file is updated.

   Forward recovery support supplied by RECOVERY(ALL) and FWDRECOVLOG is totally independent of any automatic journaling options set. Existing forward recovery utilities can still use automatic journaling options instead of RECOVERY(ALL) and FWDRECOVLOG.

You may use the following options in CEDA to provide information for a utility of your own, perhaps for forward recovery. The following example provides support for backout, with after-images for forward recovery supplied by automatic journaling options:

```
CEDA DEFINE FILE(name) GROUP(groupname)
       .
       .
     RECOVERY(BACKOUTONLY)
     JOURNAL(number)
     JNLUPDATE(YES)
     JNLADD(BEFORE)
       .
       .
```

Notes:

1. RECOVERY(BACKOUTONLY) is equivalent to LOG=YES on the DFHFCT macro for DAM files. JNLUPDATE(YES) combined with JNLADD(BEFORE) is equivalent to JREQ=(WU,NU) on the DFHFCT macro, providing the necessary images for forward recovery to a journal specified by JOURNAL.
2. An automatic journaling option, JNLADD(AFTER), journals the addition of a record after the I/O is completed rather than before. Existing forward recovery utilities will, however, work only with JNLADD(BEFORE), because the JNLADD(AFTER) produces a record with a different JCRSTRID identifier.

For information about defining files, see the *CICS Resource Definition Guide*.

The CICS system definition (CSD) file is defined by means of system initialization parameters. Parameters equivalent to RECOVERY and FWDRECOVLOG are provided together with default automatic journaling options. See the *CICS System Definition Guide* for further information.

Backout of changes to files

To make files backward recoverable, use RECOVERY(ALL|BACKOUTONLY) on the RDO FILE resource definition or LOG=YES in the DFHFCT macro. For backing out changes to such files:

1. If there is a **transaction failure**, CICS uses information from the dynamic log. DFHDBP requires exit code to handle the special case of flag deletions to DAM and VSAM-ESDS data sets (see “Global user exits in DFHDBP” on page 88).
2. At **emergency restart**, CICS uses information from the system log. DFHFCBP requires exit code to handle the special case of DAM and VSAM-ESDS flag deletions (see Chapter 11, “User exits for transaction backout during emergency restart” on page 91).

RECOVERY(ALL|BACKOUTONLY) or LOG=YES specify that the file is to be backward recoverable, and control the recording of before-images on the system log (for emergency restart). Recoverability of files affects implicit enqueuing as described under “Enqueuing in application programs” on page 113. Note that CICS enqueues read-for-update, write, and delete requests for files designated with RECOVERY(ALL|BACKOUTONLY) or LOG=YES.

If you want only backout, and not forward recovery, use RECOVERY(BACKOUTONLY) rather than RECOVERY(ALL). This avoids the overhead of logging after-images that are not going to be used.

Trapping file and data set recovery inconsistencies

Always ensure consistency of recovery attributes between files referring to the same base data set cluster or its paths. File opens that detect an inconsistency in the settings for the file and those for the associated data set, will fail.

The first file open for the base data set determines the base data set recovery attributes.

To look at the recovery attributes, use the CEMT or EXEC CICS INQUIRE DSNAME command on the base cluster to which the file refers. If all files are consistent, the recovery attributes on the file will be the same as on the base cluster.

Using the XFCNREC global user exit

CICS provides a global user exit, XFCNREC, to enable you to continue processing regardless of any inconsistencies in the backout setting for files associated with the same data set. If XFCNREC is used to suppress open failures that are a result of inconsistencies in the backout settings, a warning message will be issued to alert the user that the integrity of the data set can no longer be guaranteed.

Any CEMT or EXEC CICS INQUIRE DSNAME RECOVSTATUS command from this point onward will return NOTRECOVABLE regardless of the recovery attribute that CICS has previously enforced on the base cluster. This condition will remain

until the next CEMT SET or EXEC CICS SET DSNAME REMOVE command, or COLD START the CICS system.

It may survive a cold start if the associated data set is in a backout-failed state, because backout failed is treated as a special case on cold start with some data set information recovered from the CICS global catalog.

The order in which files are opened for the same base data set will determine the content of the message received on suppression of an open failure using XFCNREC. If the base cluster block is set as unrecoverable and a mismatch has been allowed, access to the data set could be allowed via an unrecoverable file before the data set is fully recovered.

See the *CICS Customization Guide* for programming information about the XFCNREC global user exit.

CICS responses to file open requests

CICS file control uses the backout setting from the file definition to decide whether to do logging for a file request.

CICS takes the actions shown in the following list when opening a file for update processing (that is, ADD(YES), DELETE(YES), or UPDATE(YES) on the RDO FILE resource definition. If you set only READ(YES) and/or BROWSE(YES), CICS does not make these consistency checks). These checks are not made at resource definition or install time.

- If an FCT entry refers to an alternate index (AIX®) path and RECOVERY is ALL or BACKOUTONLY on the RDO FILE resource definition, or LOG=YES on the DFHFCT TYPE=FILE macro, the AIX must be in the upgrade set for the base. This means that any changes made to the base data set are also reflected in the AIX. If the AIX is not in the upgrade set, the attempt to open the FCT entry for this AIX path fails.
- If an FCT entry is the first to be opened against a base cluster after the last cold start, the recovery attributes of the FCT entry are copied into the base cluster block.
- If an FCT entry is not the first to be opened for update against a base cluster after the last cold start, the recovery attributes in the FCT entry are checked against those copied into the base cluster block at first open. There are the following possibilities:
  - Base cluster has RECOVERY(NONE) or LOG=NO:
    - FCT entry defined with RECOVERY(NONE) or LOG=NO: the open proceeds.
    - FCT entry defined with RECOVERY(BACKOUTONLY) or LOG=YES: the attempt to open the file fails unless the user is making use of the XFCNREC global user exit to allow inconsistencies in backout settings for files associated with the same base data set.
    - FCT entry defined with RECOVERY(ALL): the open fails.
  - Base cluster has RECOVERY(BACKOUTONLY) or LOG=YES:
    - FCT entry defined with RECOVERY(NONE) or LOG=NO: the attempt to open the file fails unless the user is making use of the XFCNREC

global user exit to allow inconsistencies in backout settings for files associated with the same base data set.

- FCT entry defined with RECOVERY(BACKOUTONLY) or LOG=YES: the open proceeds.
- FCT entry defined with RECOVERY(ALL): the open fails.

– Base cluster has RECOVERY(ALL):

- FCT entry defined with RECOVERY(NONE) or LOG=NO: the open fails.
- FCT entry defined with RECOVERY(BACKOUTONLY) or LOG=YES: the open fails.
- FCT entry defined with RECOVERY(ALL): the open proceeds unless the setting of FWDRECOVLOG is different from the base cluster setting, in which case the open fails.

Any failure to open a data set for an FCT entry results in a message to the operator. If necessary, the recovery options must be changed. To change the recovery attributes (held in the base cluster block) of a VSAM data set, you can use the CEMT or EXEC CICS SET DSNAME REMOVE commands. These delete the base cluster block, so CICS has no record of prior recovery settings for the this VSAM data set. The next file to open against this data set causes a new base cluster block to be built and, if the file is opened for update, the data set takes on the recovery attributes of this file.

The base cluster block, together with its recovery attributes, and the inconsistency condition that may be set if you are using XFCNREC, is preserved even when all the files relating to it are closed, and across warm and emergency restarts. It will also survive a cold start if the associated data set is in a backout-failed state because backout failed is treated as a special case on cold start with some information recovered from the catalog.

Implementing forward recovery with existing utilities

If you use your own forward recovery programs, make sure that all files referring to the same data set have the same settings for the following options on the RDO FILE resource definition, or the equivalent DFHFCT TYPE=FILE macro operands:

JOURNAL
JNLREAD
JNLSYNCREAD
JNLUPDATE
JNLADD
JNLSYNCWRITE

It is possible that two or more CICS files relate to a single VSAM base data set. Such files may refer directly to the base, or to an alternate index path defined over the base. If you are updating records in a single data set via multiple files, forward recovery of the data set must take account of all the journal records for the data set, which must be merged and reapplied in the correct chronological order.

After-images to be used by forward recovery are recorded on the journal with FCT file entry names. To enable journal records for a given base data set to be related, before any updates are made through a particular FCT entry name, the

44-character data set name associated with that FCT entry (which may be a VSAM path or the base itself) and the data set name of the corresponding base are written to the journal.

If you use dynamic allocation of data set names, the file name included in the journal to reflect changes to the file will not uniquely identify the data set being updated. To allow your forward recovery procedures to make the association between the FCT file name and the operating system data set name, a special record is written to the journal whenever the data set allocation changes. This record contains the FCT name and the data set name.

For programming information about the format of log and journal records, see the *CICS Customization Guide*.

Implementing recoverability of temporary storage

This section deals with both backward and forward recovery of temporary storage.

Backward recovery

Temporary storage queues that are to be recoverable by CICS must be on auxiliary temporary storage.

You must identify temporary storage queues as recoverable in the temporary storage table (TST), as shown in the following outline:

```
DFHTST TYPE=RECOVERY,
       DATAID=(DF,**,
               $$(,character-string)...)
```

The DATAID DF makes the temporary storage queues used by CICS recoverable.

The DATAIDs **, and $$ make those temporary storage queues used by BMS recoverable.

The DATAID character-string represents the leading characters of each temporary storage queue identifier that you want to be recoverable. For example, DATAID=(R,ZIP) makes recoverable all temporary storage queues that have identifiers starting with the character “R” or the characters “ZIP.”

For more information on allocation and space requirements, see the *CICS Operations and Utilities Guide*.

Forward recovery

If an unrecoverable input/output error or physical failure occurs on the temporary storage data set during emergency restart (indicated by message DFHTS1302), CICS abends, and you can do one of the following:

1. If you want forward recovery of temporary storage, you should record the changes made to temporary storage during the current CICS run; you must provide application programs to do this. At emergency restart time, you can then delay the emergency restart (by using PLTPI, for example) and, again using application programs, rebuild as much as possible of the temporary storage data using the records previously read.

2. Repeat the emergency restart but with the system initialization parameters amended to cold-start temporary storage (TS=(COLD)). Note, however, that this loses the contents of the entire temporary storage data set.

Implementing recoverability of intrapartition transient data

This section deals with both backward and forward recovery of intrapartition transient data.

Backward recovery

CICS can only recover **intrapartition** transient data. For extrapartition transient data considerations, see “Recovering extrapartition transient data” on page 83.

You need to specify the name of every intrapartition transient data destination that is to be recoverable. For each name that you specify as recoverable, the data, trigger level, transaction identifier, and terminal identifier are recovered. You specify each name in the destination control table (DCT) as follows:

```
DFHDCT TYPE=INTRA,
       DESTID=name,
       DESTRCV=LG|PH
```

DESTRCV=LG denotes **logical** recovery. This means that changes to transient data get/put pointers for an interrupted LUW are backed out. In general, you should use the LG option. If, for example, you make related changes to a set of resources, including transient data, and you want to commit or back out all the changes, you will require logical recovery.

DESTRCV=PH specifies **physical** recoverability; this is unique to transient data and is implemented only at emergency restart. If the interrupted LUW was reading from the transient data destination, the get pointer is reset to the last record read. The put pointer never changes.

After a CICS failure, you might choose to restart CICS as quickly as possible, and then look for the cause of the failure. By specifying destinations such as CSMT as intrapartition and physically recoverable, the messages produced just before the failure can be recovered and are therefore available to help you diagnose the problem.

The intrapartition data set is a VSAM-ESDS data set, with file name DFHNTRA. (For more information about allocation and space requirements, see the *CICS System Definition Guide*.)

Forward recovery

If you want forward recovery of your intrapartition transient data, you have to provide application programs to record in a journal the changes to the contents of your transient data while CICS is running. The information journaled must include:

- Each PUT, including the data that is written
- Each GET
- Each deletion of a queue
- For logically-recoverable queues, each backout, syncpoint, or syncpoint rollback.

When an unrecoverable input/output error or physical failure occurs on the intrapartition transient data (indicated by messages DFHTD0360I through DFHTD0363I), restart CICS with START=AUTO (which will resolve to an emergency restart). For the restart, you must amend the DCT system initialization parameter to DCT=(xx,COLD) to cold-start transient data, thus purging all the transient data queues.

You must provide the application program to rebuild the data by reading the journaled information and applying that information to the transient data. Your application program could run in the PLT phase or after emergency restart. Until the data set is fully recovered, you must not PUT to the queue, because that would probably result in wrongly-ordered data, and a GET might not provide valid data or any data at all. For these reasons, running the recovery program in the PLT phase is probably preferable to running it after the restart.

If you do not have such a recovery strategy and you cold start a corrupted intrapartition data set, you lose the contents of the intrapartition data set.

Specifying message-protection options for VTAM terminals

For VTAM terminals, the message protection options are part of the CEDA DEFINE PROFILE command:

```
CEDA DEFINE PROFILE MSGINTEG(YES)|PROTECT(YES)
```

Select the options by altering the specification of MSGINTEG or PROTECT.

For non-VTAM terminals, install the DFHSTAND group.

Message integrity (MSGINTEG) option

The results of specifying the MSGINTEG option are:

1. All output messages from the transaction come with a request for a definite response. (Note that this increases the traffic on the network compared with a request for a response only when there is an exception.)

   CICS transmits each output message when the transaction:

   - Issues a terminal wait request
   - Issues a SYNCPOINT command
   - Ends.

2. CICS preserves the contents of the terminal input/output area (TIOA) if it does not receive a definite response, so that it can retry the operation. The contents of the TIOA are lost if:

   - The session with the terminal terminates
   - A retry is successful
   - CICS terminates.

3. CICS does not write the messages to the system log.

The MSGINTEG option can be useful in the following situations:

- When the transaction sends data to a device such as a 3270 printer. Here, a temporary fault such as “out-of-paper” can be cleared in a short time and the output operation retried, using the message preserved in the TIOA.

- When you have your own NEP processors. The NEP processor has access, through the TIOA, to the message that did not transmit successfully.

  If exception response requested was used, any message that did not transmit successfully would not definitely be preserved in the TIOA because it might have been overwritten by a later message.

Protection (PROTECT) option

The results of specifying the PROTECT option are:

1. All output messages from the transaction come with a request for a definite response. (Note that this increases the traffic on the network compared with exception response requested, which is the default.)

   CICS defers the transmission of each output message until the transaction:

   - Issues a terminal wait request
   - Issues a SYNCPOINT command
   - Ends.

2. CICS preserves the contents of the terminal input/output area (TIOA) if it does not receive a definite response so that it can retry the operation. The contents of the TIOA are lost if:

   - The session with the terminal terminates
   - A retry is successful
   - CICS terminates.

3. All input and output messages (and their SNA sequence numbers) are logged.
4. The first input message for an LUW is recorded on the dynamic log, and is available to the user input exit in dynamic transaction backout (see “Global user exits in DFHDBP” on page 88).
5. During an emergency restart, logged messages from the system log are copied:

   a. To the restart data set, where they are available to the input exit in the transaction backout program (see Chapter 11, “User exits for transaction backout during emergency restart” on page 91).

   b. To message caches in temporary storage: one cache for each terminal (see Chapter 15, “Using message caches after emergency restart” on page 123).

6. If the controller for the VTAM terminal supports the SNA set-and-test-sequence-number (STSN) command, **and** if the resynchronization and resend programs are included:

   a. During an emergency restart, the most recently committed output message for that terminal is copied to a resend slot in temporary storage, to be saved for retransmission if necessary.

   b. After emergency restart, when the terminal network is initialized, CICS participates in an exchange of sequence numbers with the terminal controller. If the sequence numbers do not match, CICS retransmits the message in the resend slot (see “Resynchronization and re-presentation of VTAM messages” on page 41).

      For this to happen, the program(s) in the controller must be able to record the sequence numbers sent to and received from CICS. The

4

4 *CICS/DOS/VS IBM 3790/3730/8100 Guide* and other subsystem guides give further information on sequence numbers and resynchronization.

Using the PROTECT option causes a considerable increase in the amount of data written to the system log, which can increase response times. Use the PROTECT option, therefore, only for transactions that update recoverable resources. (With transactions that do not update recoverable resources, logical data integrity is not at risk if messages get lost or duplicated.)

Recovering extrapartition transient data

CICS does not recover extrapartition data sets. If you depend on extrapartition data, you must develop procedures to recover data for continued execution on restart following either a controlled or an uncontrolled shutdown of CICS.

There are two areas to consider in recovering extrapartition data sets:

- Input extrapartition data sets
- Output extrapartition data sets.

Input extrapartition data sets

The main information required on restart is the number of records processed up to the time the system ended. This can be recorded during processing using CICS journaling, as described in the following paragraphs.

Each application program that reads records from extrapartition input destinations should first enqueue exclusive access to those destinations. This will prevent interleaved access to the same destinations by other concurrently executing tasks.

The application programs then issue EXEC CICS READQ TD commands to read and process extrapartition input records. In this way, they accumulate the total of input records read and processed during execution for each destination. The total number of EXEC CICS READQ operations is written to a journal data set, together with the relevant destination identifications. This journaling should be done **immediately** before EXEC CICS RETURN or SYNCPOINT commands.

Following output of the journal record, each application program dequeues itself from the extrapartition input destinations to permit other application programs to access those extrapartition input destinations.

If uncontrolled shutdown occurs before this journaling, no records will appear on the journal data set for that logical unit of work. The effect of that in-flight task is, therefore, automatically backed out on emergency restart. However, if the journal record is written before uncontrolled shutdown, this completed input data set processing will be recognized on emergency restart.

An uncontrolled shutdown does not permit a tape journal data set to close normally. The tape journal can close using the CICS tape end-of-file utility program (DFHTEOF) before executing the recovery program.

On emergency restart following uncontrolled shutdown or on a warm start following a controlled shutdown, use the following procedure, which will reposition the extrapartition input data sets to reflect the input and processing of their records during previous CICS operation.

You can identify an extrapartition input recovery program in the PLT for execution during the initialization phase. This program reads the journal data set forward. Each journaled record indicates the number of EXEC CICS READQ TD operations performed on the relevant extrapartition input data set during previous execution of application programs. The same number of EXEC CICS READQ TD commands is issued again by the recovery program, to the same input destination that was referenced previously.

On reaching the end of the journal data set, the extrapartition input data sets are positioned at the same point they had reached before the initiation of tasks that were in-flight at uncontrolled shutdown. The result is the logical recovery of these input data sets with in-flight task activity backed out.

Output extrapartition data sets

The recovery of output extrapartition data sets is somewhat different from the recovery of input data sets.

For a tape output data set, use a new output tape on restart. You can then use the previous output tape if it is necessary to recover information recorded before termination.

To avoid losing data in tape output buffers on termination, it may be desirable to write unblocked records. Alternatively, write the data to an intrapartition disk destination (recovered by CICS on a warm start or emergency restart) and periodically copy it to the extrapartition tape destination by an automatically initiated task. In the event of termination, the data is still available to be recopied on restart.

If a controlled shutdown of CICS occurs, the previous output tape closes correctly and writes a tape mark. However, on an uncontrolled shutdown such as a power failure or machine check, a tape mark is not written to indicate the end of the tape.

For a line printer output data set, you can choose just to carry on from where printing stopped when the system stopped. However, if you want to continue output from a defined point such as at the beginning of a page, you may need to use a journal data set. As each page is completed during normal CICS operation, write a record to a journal data set.

On restart, the page that was being processed at the time of failure can be identified from the journal data set, and that page can be reprocessed to reproduce the same output. Alternatively, use an intermediate intrapartition destination (as previously described) for tape output buffers.

Using initialization (PLTPI) programs

You can use initialization (PLTPI) programs:

- As part of the processing required to recover extrapartition transient data.
- To ENABLE exits required during recovery.

There are *two* PLTPI phases. The first phase occurs before the system initialization task is attached, and should not use CICS resources because initialization is incomplete. The first phase is intended solely to enable exits that are needed during recovery processing. The second phase occurs after CICS

initialization is complete and, at this point, you may use PLT programs to customize the environment.

For information on how to code the PLT, see the *CICS Resource Definition Guide*. For programming information about the special conditions that apply to PLT programs, see the *CICS Customization Guide*.

Chapter 10. Dynamic transaction backout (DTB)

In transaction backout, CICS restores the resources specified as recoverable to the state they were in at the beginning of the task. This chapter discusses dynamic transaction backout in the following topics:

This chapter contains Product-sensitive Programming Interface information.

Specifying DTB

"Dynamic transaction backout (DTB)" on page 47 describes the way that DTB works for various resources. The specification of basic recovery and restart facilities is described on page 57. In addition, you should note the following:

- DTB is the default for all transactions. For other resources, you must decide whether to make a resource recoverable. For files, for example, you need to specify RECOVERY(ALL|BACKOUTONLY) if you are using RDO to define the file, or LOG=YES in the DFHFCT macro.
- For DTB, you must specify a journal control table in the system initialization parameters, because the journal control program writes records to the dynamic log. The dummy journal control program is not adequate.
- If DFHDBP is to back out changes to files referring to DAM or VSAM-ESDS data sets, you must prepare your own code for the file error exit from DFHDBP (see "Global user exits in DFHDBP" on page 88).
- To avoid the risk of CICS abending when it runs short-on-storage, you should make resident the version of DFHDBP (suffix 1$, 2$, or xx) that you choose.
3 You do this by changing the RDO PROGRAM resource definition of the
3 program to RESIDENT(YES). If DFHDBP is not resident, and CICS cannot load it when an abend occurs in a short-on-storage situation, another abend will occur which will terminate CICS.

Specifying automatic transaction restart

To specify automatic transaction restart:

1. Ensure that you define your resources for recovery.
2. Specify RESTART(YES) in the RDO TRANSACTION resource definition for the transactions that are to be candidates for automatic restart (see "Definitions for transactions and programs" on page 61).
3. Check the logic of those transactions for any additional resources that need to be made recoverable. For example:
   - Any temporary storage or transient data (intrapartition) queues used by a transaction that may be automatically restarted should be made recoverable (see "Implementing recoverability of temporary storage" on

page 79, and “Implementing recoverability of intrapartition transient data” on page 80).

- If an EXEC CICS START FROM command is used to create a restartable task, the initial data should be protected by, for example, a DFHTST TYPE=RECOVERY,DATAID=xx macro, where the DATAID parameter corresponds to the REQID parameter in the START FROM command.

4. Be aware of the conditions necessary for automatic transaction restart. The default transaction restart program does not request a restart if the transaction abends in the second or subsequent LUW or if terminal traffic has occurred for this task.

   If you want automatic restart to occur under different conditions, you can edit the CICS-supplied transaction restart program (DFHREST), as described in “Editing the transaction restart program (DFHREST)” on page 89.

Global user exits in DFHDBP

DFHDBP has four global user exit points:

1. XDBINIT
2. XDBIN
3. XDBDERR
4. XDBFERR.

You can write programs to be executed at any of these exits if the default action is not required or if you want to perform some processing in addition to the default action, such as:

- Examining log records (with the possibility of special action for certain types of record)
- Handling file and database error conditions
- Deciding whether backout is to continue or to be suppressed (either completely or for certain resources).

For programming information on the parameters passed to the exit programs, the XPI calls, and the return codes used by the exit programs, see the *CICS Customization Guide*.

The return codes that can be returned by the exit program are as follows:

UERCNORM The default return code. If UERCNORM is returned, the data set associated with the file is flagged as “backout failed”. The data set is no longer available to applications and, following a quiesce of activity against the base data set, you may run a batch backout utility. For more information about flagging backout errors, see Chapter 16, “Backout failure” on page 129. If you are not using a batch backout utility or some other means of coping with backout failures, and data integrity is at risk, abend CICS from your exit program, and perform an emergency restart to preserve data integrity.

UERCBYP Indicates that the error is ignored and backout continues. The file is not flagged as “backout failed”.

UERCRTRY Return code UERCRTRY has two meanings:

1. For the DBFEWA error type, the record that has been marked by the exit program as “logically deleted”, and which is held in the area pointed to by UEPFDATA, will be reapplied to the data set
2. For other error types, the file control request is retried.

UERCPURG Return code UERCPURG can also be issued by an exit program that has invoked the XPI (exit programming interface).

Coding DFHDBP global user exits

You may modify recoverable resources in DFHDBP global user exits, but note the following:

- Dynamic transaction backout exits must be quasi-reentrant. They may use the exit programming interface (XPI) and issue EXEC CICS commands. If EXEC CICS commands are included in the exit program, the program must be compiled with the NOEDF option to avoid the risk of an abend in the CEDF facility if dynamic backout of a transaction occurs while CEDF is active.
- In the XDBINIT exit, avoid changes to recoverable transient data and temporary storage because they will back out immediately.
- In the XDBIN exit, you can set a return code to ignore a file-related record if, for example, backout for a particular file is to be suppressed for some reason.
- A file control EXEC CICS READ UPDATE command should be properly unlocked, either implicitly or explicitly, or backout may be locked out. In fact, it is unwise to issue any file control requests when backing out file resources.
- The current DL/I PSB should be left scheduled; it should not be terminated.
- File control operations are performed by DFHDBP and changes made to files (including those performed in user exits) will be recorded in the system log by the file control program (DFHFCVS).

Editing the transaction restart program (DFHREST)

When planning to replace the default DFHREST, check to see if the logic of any of your transactions is inappropriate for restart.

- Transactions that execute as a single logical unit of work are safe. Those that execute a loop and, on each pass, read one record from a recoverable destination, update other recoverable resources, and close with a syncpoint, are also safe.
- Two types of transaction need to be modified to avoid erroneously repeating work done in the logical units of work that precede an abend:
  1. A transaction in which the first and subsequent logical units of work change different resources
  2. A transaction where the contents of the input data area are used in several logical units of work.

For programming information about DFHREST and guidance to help determine if transaction restart is to happen, see the *CICS Customization Guide*.

Chapter 11. User exits for transaction backout during emergency restart

This chapter describes the opportunities for including your own logic in global exit programs that run in the transaction backout programs—DFHFCBP, DFHUSBP, DFHTCBP, and DFHDLBP—at emergency restart time. The way these programs work is described in “Backout processing” on page 36. Transient data and temporary storage backout do not have any exits.

This chapter contains Product-sensitive Programming Interface information. For additional programming information on global user exits, see the *CICS Customization Guide*.

Where you can add your own code

At emergency restart, you can add your own code in postinitialization programs that you nominate in the program list table (described on page 84).

You can include functions in global exit programs that run during emergency restart to:

- Deal with flag deletions (in the XRCFCER exit of DFHFCBP)
- Handle file error conditions that arise during emergency restart
- Process journaled records (in the XRCINPT exit of DFHUSBP).

4 The transaction backout programs have five global user exit points:

1. XRCINIT—initialization and termination exit
4 2. XRCINPT—input exit (only for DFHFCBP, DFHUSBP or DFHTCBP)
3. XRCFCER—file error exit (only for DFHFCBP)
4. XRCOPER—open error exit (only for DFHFCBP).
5. XRCDBER—DL/I backout error exit (only for DFHDLBP)

You can use any of these exits to add your own processing if you do not want the default action. To use these exits, you must either enable them in PLT programs in the first stage of PLT processing, or specify them in system initialization parameters
4 with TBEXITS=(name1,name2,name3,name4,name5), where name1, name2,
4 name3, name4, and name5 are the names of your programs for XRCINIT,
4 XRCINPT, XRCFCER, XRCOPER, and XRCDBER.

Figure 3 on page 92 shows which programs the user exits are invoked in, and the order in which they are invoked.

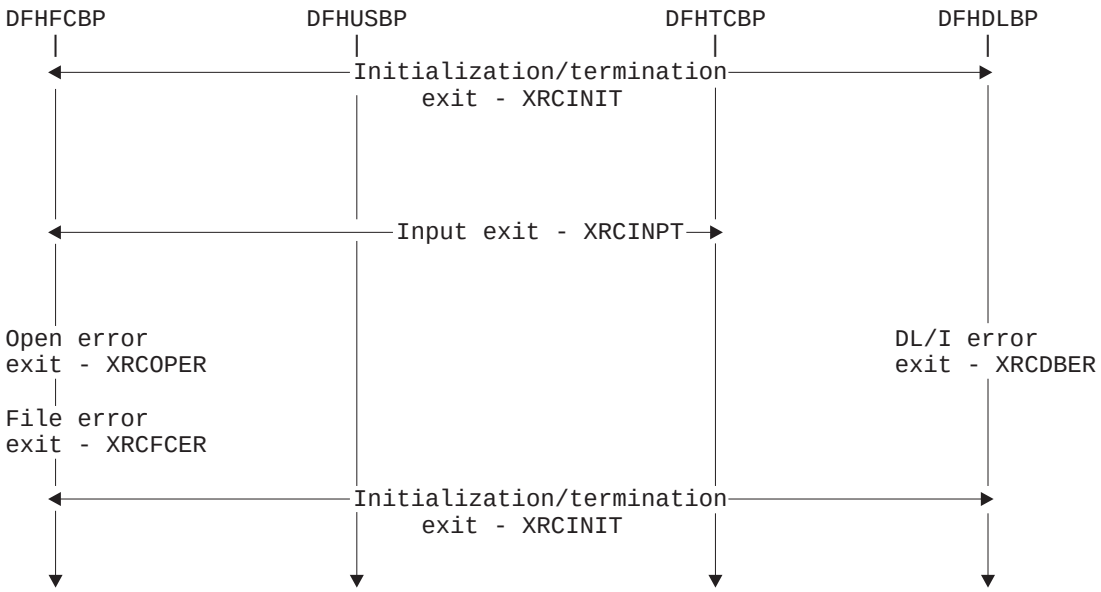

Figure 3. Global user exits for backout at recovery

Global user exit details

For programming information about the following, see the *CICS Customization Guide*.

- The identity of the invoking program
- The exit for initialization and termination
- The time of invocation, as indicated to the exit programs by parameters
- Writing exit programs
- Details of the input parameters
- Return codes for each exit

You must not set the UERCPURG return code for these exits, because the exit tasks cannot be purged.

XRCINIT exit

This is the initialization and termination exit. It gains control when:

1. Each of DFHUSBP, DFHFCBP, DFHTCBP, and DFHDLBP is first invoked
2. Each of these programs ends.

The XRCINIT exit code must always end with a return code of UERCNORM. No choice of processing options is available to this exit.

The XRCINIT exit can, however, set the no-action flags in the following:

- The file backout table (FBO)
- The DL/I backout table (DBO)
- The message backout table (MBO)

These tables are created on the restart data set during emergency restart. The XRCINIT exit is the only exit that can set these no-action flags.

For *file backout*, the FBO is described by the DFHFBODS copybook. The entries in the FBO are verified against the files that have been defined and marked as “absent” and “no action” if unmatched. Before giving control to the exit, DFHFCBP lists the absent file IDs to the console operator.

For *DL/I backout*, the DBO is described by the DFHDBODS copybook. The entries in the DBO are verified against the loaded DL/I DMB and PSB directories and marked as “absent” and “no action” if unmatched. Before giving control to the exit, DFHDLBP lists, to the console operator, the PSB and DMB names that either cannot be found or cannot be scheduled.

For any task in which DL/I VSE backout processing is stopped in this way, CICS safeguards DL/I VSE data integrity thus:

1. CICS identifies the PSB that was in use by the task, and then “stops” all those databases updated by in-flight tasks using that PSB. “Stopping” the databases means flagging them so that future tasks cannot schedule PSBs that refer to any of those databases.
2. CICS continues emergency restart processing.

For *message backout*, the MBO is described by the DFHMBODS copybook. The entries in the MBO are verified against the loaded terminal control table and marked as “absent” and “no action” if unmatched.

For *backout of user entries in the system log*, the transaction backout table (TBO), described in the DFHTBODS copybook, is relevant to distinguish between those records written by inflight-LUWs and those written by completed LUWs. Because of the absence of **no-action** flags in the TBO, the records of each type are presented at the XRCINPT user exit regardless of action taken at the XRCINIT exit. The processing made possible by this exit is described on page 38.

Note: Records for completed tasks are copied to the restart data set, even though backout processing ignores them. These records are presented to the XRCINIT exit. Completed tasks are those for which recovery control encounters records written with the high-order bit set on in the JTYPEID operand of the EXEC CICS WRITE JOURNALNUM command.

XRCINPT exit

This is the input exit. It is given control each time a record (other than a DL/I record) is read from the restart data set. (The record is copied to the restart data set from the system log.)

The default actions at this exit are:

User journaled records
No action.

Automatically journaled records
No action.

Logged records applying to files or terminals flagged for no action
No action.

Logged read-updates
Reapply the before-image of the record to the file.

Logged write-add
For DAM and VSAM-ESDS files, the XRCFCER file error exit (see below) is given control. For VSAM KSDS/RRDS files, the default action is to delete the record.

Logged temporary storage PUT(Q)-REPLACE
Reapply the before-image of the record to temporary storage.

Logged terminal messages
Save the records in the temporary storage resend slot or message cache, or both as appropriate.

If you want to ignore the log record, return with return code UERCBYP. This frees the record area immediately and reads a new record from the restart data set. Take care that this action does not put data integrity at risk.

XRCFCER exit

This is the file error exit. It is given control when an error condition is returned from the file control program during the backout processing, or if an error is detected by DFHFCBP itself. Error conditions include:

- Input/output errors
- Logical errors caused by attempting inconsistent file operations.

The return codes are:

UERCNORM If the default return code, UERCNORM is set, the data set associated with the file is flagged as "backout failed". Its backout status is set as failed in the base cluster block, the backout-failed record is logged, and all files open against the base are closed. The data set is no longer available to applications, and you may run a backout utility. For more information about flagging backout errors, see Chapter 16, "Backout failure" on page 129.

If you are not using a backout utility or some other means of coping with backout failures, and data integrity is at risk, you should abend CICS from your exit program, correct the source of the failure, and perform another emergency restart to preserve data integrity.

UERCBYP Indicates that the error is ignored and backout continues. The data set is not flagged as "backout failed".

UERCRTRY Return code UERCRTRY has two meanings:

1. For the TBFEWA error type, the updated record is reapplied to the data set
2. For other error types, the file control request is retried.

XRCOPER exit

This is the open error exit, for program DFHFCBP only. It assists in file control backout.

This exit gains control if an error occurs while opening a file. If the open error has been caused by a backout failure, the exit gains control without reference to the operator. If the open error is caused by anything else, a message is written to CSMT and to the console operator with a "GO" or "CANCEL" option. In that case, the exit only gains control if the "GO" option is selected, and backout failure control preserves data integrity. If CANCEL IS SELECTED, CICS abends. The default

action is to continue normally, and will include backout failure processing code. Upon return from the exit, the file backout table entry is marked “no action” by DFHFCBP.

Coding transaction backout exits

You have access to all CICS services, except terminal control services, during exit execution. However, the following restrictions should be considered:

- Transaction backout exits must be written in assembler code.
- Transaction backout exits must be quasi-reentrant. They may use the exit programming interface (XPI) and issue EXEC CICS commands.
- If an exit acquires an area as a result of a file control request, it is the responsibility of the exit to release that area.
- An exit must not attempt to make any file control requests to a file referring to a VSAM data set with a string number of 1, unless no action is specified for that file during the initialization exit.
- Task-chained storage acquired in an exit is released at the completion of emergency restart processing. However, the exit should attempt to release the storage as soon as its contents are no longer needed.
- No exit should reset either the absent or no-action indicators set by DFHFCBP.
- If an exit is not used, the default actions are taken.
- We strongly recommend that emergency restart exit code does not change any *recoverable* resource. If you do try to use temporary storage, transient data, file control, or DL/I, these resources may also be in a state of recovery and therefore “not open for business”. Access to these services will, therefore, at best cause serialization of the recovery tasks and, at worst, cause a deadlock.

Chapter 12. Handling communication errors

This chapter describes communication design and provides guidance on aspects of coding the following error programs:

- Node error program (NEP)
- Terminal error program (TEP).

The process is discussed in the following topics:

For information about how these programs work, and some design considerations for them, see Chapter 6, "Communication error processing" on page 53.

For programming information to complement the information in this book, see the *CICS Customization Guide*, which contains advice on writing these error programs.

Communication design

Communication design is discussed under the following headings:

Communications-related programming considerations

To tell a user that requested updates have been successfully applied, the application program usually sends a confirmation message after the updates are complete.

Assuming (1) that the transaction issues only one EXEC CICS SEND (or SEND MAP) command within an LUW, and (2) that the chosen command does not cause an immediate (not deferred) transmission (such as the EXEC CICS CONVERSE command), the output transmission is deferred until after syncpoint processing at the end of the LUW. That is, the confirmation message is not sent until the updates are committed. Using multiple SEND commands interleaved with file or database updates in the same LUW is not recommended because, if failure occurs, updates that the user believes to be complete may be backed out.

Notes:

1. A WAIT request associated with a SEND command destroys message integrity by forcing immediate transmission of the message. If the task then fails, updates to recoverable files are backed out, but the message cannot be recalled.
2. The DEFRESP option of an EXEC CICS SEND command to a VTAM terminal indicates that a definite response is required when the output operation has been completed. For programming information about EXEC CICS commands, see the *CICS Application Programming Reference* manual.
3. Specify maximum protection (for VTAM messages) by PROTECT(YES) on the RDO TRANSACTION resource definition. Output messages are preserved in

the TIOA (see 2 above). Input and output messages (with SNA sequence numbers) and the SNA responses are logged. This logging enables CICS to create message caches and resend slots during emergency restart (see “Specifying message-protection options for VTAM terminals” on page 81).

Journaling of messages

The application designer may wish to record input and output messages. Reasons for doing so include:

- Creating an audit trail of messages sent and received could assist in problem determination
- Logging messages for non-VTAM terminals to provide a similar function to that provided by CICS for VTAM terminals
- Gathering data for performance or stress tests, or for message reprocessing.

Handling communication breaks

The main reasons why you might want to tailor the supplied NEP or TEP are listed below. However, you are advised to use the default program for a while, getting experience of communication error handling before deciding what error handling best suits your needs.

- If CICS cannot deliver an output message that contains confidential information (and so cannot be rerouted) and if the communication error is not transitory, consider forcing the user off the system (so that a signon is required to continue).

  For VTAM terminals, code in the NEP could achieve this by setting flags that cause CICS to close destination and terminate the session with the terminal.
- If a message cannot be delivered and it relates to critical updates, it may be necessary to code the NEP or TEP to send a message to another terminal (for example, to the master terminal operator).
- If a message is sent to a 3270 printer and no printer is available, NEP or TEP code could reroute the message to another printer.
- If CICS attempts to send output (for example, an error message) to an input-only terminal that is to be used by the application, NEP or TEP code could reroute the message to another terminal.
- If too much error information is being printed, NEP or TEP code could reduce it to manageable proportions.

Node error program (DFHZNEP)—VTAM logical units

The VTAM node error program (NEP) is invoked by the node abnormal condition program (DFHZNAC) after it has prepared to issue error messages and has set flags appropriate to the type of error that has occurred. Chapter 6, “Communication error processing” on page 53 introduces the NEP, and “Handling communication breaks” offers some design ideas.

The NEP can be:

- The default NEP
- The CICS sample NEP
- Your own NEP or series of NEP processors.

The NEP can change the flag settings or perform other actions. When control returns to DFHZNAC, the flag settings control actions such as:

- Printing control blocks and areas associated with the error (for example, TIOA, VTAM RPL, TCTTE)
- Terminating VTAM send or receive requests, and abending the associated task
- Closing the session with the terminal.

You can handle some errors in your application program by using the TERMERR error condition. If you do handle errors in your own programs, you simplify recovery and restart design, because you will be able to determine a course of action (logging data or backing out, for example) in the application itself.

The default NEP

The default node error program is pregenerated. It performs no processing and leaves the flags set by DFHZNAC unchanged.

Because VTAM and the network control program (NCP) attempt to recover from error conditions, new CICS users are recommended to use the default NEP rather than generating the CICS sample NEP or writing special-purpose NEP processors. Until you understand the interactions of applications and network management, you can change the node status by using CEMT and VTAM commands.

The CICS sample NEP

The CICS sample NEP can provide extended error handling for 3270 logical units and interactive logical units. It can also provide a framework for your own NEPs.

You use the DFHSNEP macros to generate the sample NEP; there is programming information about this in the *CICS Customization Guide*.

Your own NEP processors

The implementation of terminal error processing for VTAM-supported terminals is such that any error is normally routed to the node abnormal condition program (DFHZNAC). Depending on the type of error, DFHZNAC sets error and action flags and hands over control to the appropriate node error program. This may be the CICS sample NEP or your own version(s) of that program.

Interactions between the applications and VTAM can depend upon the characteristics of the transaction and the installation. For this reason, CICS provides the framework for you to write NEP processors to handle different network error conditions.

CICS gives you the opportunity of providing, in table form, an interface module and a separate error routine for each of a number of transaction classes. The function of the interface module is to allow a particular transaction (or group of transactions):

- To have its own error processing procedure
- To determine which class of transaction is attached to the terminal
- To link from DFHZNAC to the appropriate node error program.

On completion of the action in the transaction class error routine, control returns to DFHZNAC from the NEP, using the EXEC CICS RETURN command.

Terminal error program (DFHTEP)—non-VTAM terminals

The terminal error program (TEP) is invoked by the terminal abnormal condition program (DFHTACP) when an abnormal condition associated with a non-VTAM terminal or line occurs. Chapter 6, “Communication error processing” on page 53 introduces the TEP, and “Handling communication breaks” on page 98 offers some design ideas.

The TEP can be:

- The CICS sample TEP
- Your own TEP.

The CICS sample TEP

4 The CICS sample TEP is supplied in the VSE/ESA™ sublibrary PRD1.BASE. The sample program and table supply default processing for terminal errors, with a maximum of 10 terminal error blocks (TEBs). If you use the sample, CICS can handle no more than 10 terminal errors concurrently. If you want to define your own error processing, use the DFHTEPM and DFHTEPT macros to generate an error program and a table that includes your error routines.

You obtain the required program definition by installing the DFHSTAND group from the CICS system definition (CSD) file.

Because the nature of communication errors is unpredictable, you are advised to use the sample TEP at first to gain experience of network operations in your environment. By studying CICS statistics about communication errors over a period of time, you can then decide how or if to change the sample TEP.

Your own TEP code

The implementation of terminal error processing for non-VTAM terminals is such that any error is normally routed to the terminal abnormal condition program (DFHTACP). Depending on the type of error, DFHTACP issues messages, sets error flags, places the terminal out of service, and hands over control to the terminal error program, DFHTEP, a sample version of which is supplied by CICS (DFHXTEP in source code form). After any necessary action by DFHTEP, control returns to DFHTACP.

There are some situations in which CICS may attempt to send a message to an input-only terminal; for example, an invalid transaction identification message, or a message erroneously sent by an application program. You can provide a terminal error program to reroute these messages to a system destination such as CSMT or CSTL or other destinations by means of transient data or interval control facilities.

Chapter 13. Recovery coding in application programs

This chapter describes how you can include recovery facilities in your application design. It covers the following topics:

Before you proceed, note the terms used in this section:

Application
In this context, application refers to a set of one or more *application units of work* designed to fulfill a particular need (or needs) of the user organization.

Application unit of work
This refers to a set of actions within an application which the designer chooses to regard as an entity. It is for the designer to decide how (if at all) to subdivide an application into application units of work, and whether any application unit of work should consist of just one or many CICS *logical* units of work (LUWs). (A logical unit of work (LUW) is a CICS term that refers to a sequence of processing where recoverable resources are protected against double updating, and changes to recoverable resources are backed out if the LUW is interrupted.)

Typically, but not exclusively, an application unit of work would correspond to a CICS LUW.

An order-entry application might comprise all the actions needed to process one order from a customer. It might be designed as a set of application units of work, as follows: (1) check customer's name and address and allocate an order number, (2) record details of ordered items and update inventory files, and (3) print invoices and shipping documents. According to the agreed recovery requirements statement, noting details of ordered items and updating files might be implemented as either one large application unit of work or many application units of work—one for each item within the order.

Application design

This section tells you how to design your applications so that they take advantage of the CICS recovery facilities.

Splitting the application into application units of work

Specify how to subdivide the application into application units of work. Name each application unit of work, and describe its function in terms that the user can understand.

Consider also the inclusion of supplementary application units of work to provide such functions as:

- *Progress transaction*, to check on progress through the application. Such a function could be used after a transaction failure or after emergency restart, as well as at any time during normal operation.

- *Catch-up function*, for entering data that the user may have been forced to accumulate by other means during a system failure.

Files accessed by each transaction

For each application unit of work, specify the files and databases that can be accessed.

Of the files and databases that can be accessed, specify those that are to be updated (as distinct from those that are only to be read).

Updates performed by each application unit of work

For those files and databases updated by an application unit of work, specify how to apply the updates; factors to consider here are the synchrony and the immediacy of updates.

Synchrony of updates: Specify which (if any) updates must happen in step with each other to ensure integrity of data. For example, in an order-entry application, it may be necessary to ensure that a quantity subtracted from the inventory file is, at the same time, added to the to-be-shipped file.

Immediacy of updates: Specify *when* newly entered data must or can be applied to the files or databases. Possibilities include:

- The application unit of work updates the files and databases as soon as the data is accepted from the user.
- The application unit of work accumulates updates for later processing, for example:
  - By a later application unit of work within the same application.
  - By a batch application that runs overnight. (If you choose this option, make sure that there is enough time for the batch work to complete the number of updates.)

Use the above information when deciding on the internal design of application units of work.

Relationships between application units of work

Specify what data needs to be passed from one application unit of work to another.

For example, in an order-entry application, one application unit of work may accumulate order items. Another, separate, application unit of work may update the inventory file. Clearly, there is a need here for the data accumulated by the first application unit of work to be passed to the other application unit of work.

This information is needed when deciding what resources are needed by each application unit of work (see “Mechanisms for passing data between transactions” on page 105).

SAA-compatible applications

The resource recovery element of the Systems Application Architecture (SAA) common programming interface (CPI) provides an alternative to the standard CICS application program interface (API) if you need to implement SAA-compatible applications. The resource recovery facilities provided by the CICS implementation of the SAA resource recovery interface are the same as those provided by CICS API. So, if you are an existing CICS/VSE™ user, you need to change from CICS API to SAA resource recovery commands only if your application needs to be SAA-compatible.

To use the SAA resource recovery interface, you need to include SAA resource recovery commands in your applications in place of EXEC CICS SYNCPOINT commands. This book refers only to CICS API resource recovery commands; for information about the SAA resource recovery interface, see the *CPI Resource Recovery Reference* manual.

Program design

This section tells you how to design your programs to use the CICS recovery facilities effectively.

Dividing transactions into logical units of work

When deciding how to implement application units of work in terms of transactions, logical units of work (LUWs), and programs, consider the following:

- In programs that support a dialog with the user, consider implementing each LUW to include only a single terminal read and a single terminal write. This can simplify the user restart procedures (see also "Processing dialogs with users" on page 104).

  Short LUWs are recommended for several reasons:

  - Data resources are enqueued for a shorter time. This reduces the chance of other tasks having to wait for the resource to be freed.
  - Backout processing time (in dynamic transaction backout or emergency restart) is shortened.
  - The user has less to reenter when a transaction restarts after a failure.

    In applications for which little or no rekeying is feasible (discussed under "Question 9: How is the user to continue or restart entering data after a failure?" on page 58), short LUWs are essential so that all entered data is **committed** as soon as possible.

- Consider the recovery/restart implications when deciding whether to divide a transaction into many LUWs. CICS functions such as dynamic transaction backout, message recovery, and transaction restart work most efficiently for transactions that have only one LUW. But there can be situations in which multiple-LUW transactions are necessary, for example if a set of file or database updates must be irrevocably committed in **one** LUW, but the transaction is to continue with one or more LUWs for further processing.

  The decision to have one LUW, or multiple LUWs, in a given transaction should be made only after carefully considering the recovery and restart implications.

- Where file or database updates must be kept in step, make sure that your application does them in the same LUW (see “Updates performed by each application unit of work” on page 102). This ensures that those updates will all be committed together or—in the event of the LUW being interrupted—will back out together to a consistent state.

Processing dialogs with users

An application may require several interactions (input and output) with the user. The following basic techniques for program design are available in CICS for use in such situations:

- Conversational processing
- Pseudoconversational processing.

Conversational processing

With conversational processing, the transaction continues to run as a task across all terminal interactions—including the time it takes for the user to read output and enter input. While it runs, the task retains resources that may be needed by other tasks. For example:

- The task occupies storage and enqueues database records for a considerable period of time. Also, in the event of a failure and subsequent backout, all the updates to files and databases made up to the moment of failure have to be backed out (unless the transaction has been subdivided into LUWs).
- If the transaction uses DL/I VSE, and the number of scheduled PSBs reaches the maximum allowed, tasks needing to schedule further PSBs have to wait.

Conversational processing is not generally favored, but may be required where multiple file or database updates made by multiple interactions with the user must be related to each other—that is, they must all be committed together, or all backed out together, in order to maintain data integrity.

Pseudoconversational processing

With pseudoconversational processing, successive terminal interactions with the user are processed as separate tasks—usually consisting of one LUW each. (This approach can result in a need to communicate between tasks or transactions (see “Mechanisms for passing data between transactions” on page 105) and the application programming can be a little more complex than for conversational processing.)

However, at the end of each task, the updates are committed, and the resources associated with the task are released for use by other tasks. For this reason, the pseudoconversational technique is generally preferred to the conversational technique.

When multiple terminal interactions with the user are related to each other, data for updates should accumulate on a recoverable resource (see “CICS recoverable resources for communication between transactions” on page 105), and then be applied to the database in a single task (for example, in the last interaction of a conversation). In the event of a failure, emergency restart or dynamic transaction backout would need to back out only the updates made during that individual step; the application would be responsible for restarting at the appropriate point in the conversation. This may involve re-creating a screen format.

Bear in mind, however, that other tasks may try to update the database between the time when update information is accepted, and the time when it is applied to the database. Design your application to ensure that no other application can update the database at a time when it would corrupt your updating.

Mechanisms for passing data between transactions

In those applications where one transaction needs to access working data created by a previous transaction, consider what mechanism should carry that data over. The possible mechanisms are discussed under two broad headings:

- Main storage areas for communication between transactions
- CICS recoverable resources for communication between transactions.

See also “Implications of interval control START requests” on page 107.

Main storage areas for communication between transactions

Main storage areas that can be used to pass data between transactions include:

- The communication area (COMMAREA)
- The common work area (CWA)
- Temporary storage (main)
- The terminal control table user area (TCTUA).

CICS does not log changes to these areas (except as noted later in this section). Therefore, in the event of an uncontrolled shutdown, data stored in any of these areas is lost, which makes them unsuitable for applications needing to retain data between transactions across an emergency restart.

The advantages of main storage areas are realized only where recovery is not important, or when passing data between programs servicing the same task.

Note: Programs should be designed so that they do **not** rely on the presence or absence of data in the COMMAREA to indicate whether or not control has been passed to the program for the first time (for example, by testing for a data length of zero). Consider the abend of a transaction where dynamic transaction backout and automatic restart are specified. After the abend, a COMMAREA could be passed to the next transaction from the terminal, even though the new transaction is unrelated. Similar considerations apply to the terminal control table user area (TCTUA).

CICS recoverable resources for communication between transactions

Resources recoverable by backout for communication between transactions include:

- Temporary storage (auxiliary) queues
- Transient data queues
- User files and DL/I databases.

CICS can return all these to their status at the beginning of an in-flight LUW in the event of an abnormal task termination.

Temporary storage (auxiliary) queues: A temporary storage item can be used for communication between transactions. (For this purpose, the temporary storage item needs to be unique to the terminal ID. If the terminal becomes unavailable,

the transaction sequence is interrupted until the terminal is again available.) The temporary storage queue-name (QUEUE option on EXEC CICS TS commands) can be read and reread, but the application program must delete it when it is no longer needed for communication between a sequence of transactions.

Transient data queues: Transient data (intrapartition) is similar to temporary storage (auxiliary) for communicating between transactions, the main difference being that each record in the queue can be read only once. Transient data must be specified as **logically** recoverable (in the destination control table) to achieve backout to the start of any in-flight LUW.

User files and DL/I databases: You can dedicate files or database segments to communicating data between transactions.

Transactions can record the completion of certain functions on the dedicated file or database segment. A progress transaction (whose purpose is to tell the user what updates have and have not been performed) can examine the dedicated file or segment.

In the event of physical damage, user VSAM files, and DL/I databases can be forward recovered.

Designing to avoid transaction deadlock

To avoid transaction deadlock (see “Possibility of transaction deadlock” on page 119), consider the following techniques:

- Arrange for all transactions to access files in a sequence agreed in advance. This could be a suitable subject for installation standards. Be extra careful if you allow updates through multiple paths. More information is at the end of this section.
- Enforce explicit installation enqueueing standards so that all applications:
  - Enqueue by the same character string
  - Use those strings in the same sequence.
- Always access records within a file in the same sequence. For example, where multiple file or database records are updated, ensure that you access them in ascending sequence.

  Ways of doing this include the following:

  - The terminal operator always enters data in the existing data set sequence.

    This method requires special terminal operator action, which may not be practical within the constraints of the application. (For example, orders may be taken by telephone in random product number sequence.)

  - The application program first sorts the input transaction contents so that the sequence of data items matches the sequence on the data set.

    This method requires additional application programming, but imposes no external constraints on the terminal operator or the application.

  - The application program issues an EXEC CICS SYNCPOINT command after processing each data item entered in the transaction.

    This method requires less additional programming than the second method. However, issuing a synchronization point implies that previously processed data items in the transaction are not to be backed out if a system or

transaction failure occurs before the entire transaction ends. This may not be valid for the application, and raises the question as to which data items in the transaction were processed and which were backed out by CICS. If the entire transaction must be backed out, synchronization points should not be issued, or only one data item should be entered per transaction.

Of the three methods, the second (sorting data items into an ascending sequence by programming) is most widely accepted.

Note that, if you allow updates on a data set through the base and one or more AIX paths, or through multiple AIX paths, sequencing multiple record updates may not provide protection against transaction deadlock. You are not protected because the different base key sequences will probably not all be in ascending (or descending) order. If you do allow updates through multiple paths, and if you need to perform multiple record updates, always use a single path or the base. Such a procedure should be defined by installation standards.

Implications of interval control START requests

Interval control EXEC CICS START requests initiate another task—for example, to perform updates accumulated by the START-issuing task; this allows the user to continue accumulating data without waiting for the updates to be applied.

The PROTECT option on a EXEC CICS START request ensures that, if the task issuing the START fails during the LUW, the new task will not be initiated, even though its start time may have passed.

Consider also the possibility of a started task that fails. Unless you include abend processing in the program, only the master terminal will know about the failure. The abend processing should analyze the cause of failure as far as possible, and restart the task if appropriate. Ensure that either the user or master terminal operator can take appropriate action to repeat the updates. You can, for example, allow the user to reinitiate the task.

An alternative solution is for the started transaction to issue an EXEC CICS START command specifying its *own* TRANSID. Immediately before issuing the EXEC CICS RETURN command, the transaction should cancel the START command. The effect of this will be that, if a started task fails, it will automatically restart. (If the interval specified in the START command is too short, the transaction could be invoked again while the first invocation is still running. Ensure that the interval is long enough to prevent this.)

Implications of automatic task initiation (transient data trigger level)

Specifying the TRANSID operand in the DCT for an intrapartition transient data destination starts the named transaction when the trigger level is reached. Designate such a destination as **logically** recoverable. This ensures that the transient data records are committed before the task executes and uses those records.

Implications of presenting large amounts of data to the user

Ideally, a transaction that updates files or databases should defer confirmation (to the user) until such updates are committed (by user syncpoint or end of task).

In cases where the application requires the reply to consist of a large amount of data that cannot all be viewed at one time (such as data required for browsing), several techniques are available, including:

- Terminal paging through BMS
- Using transient data queues.

Terminal paging through BMS

The application program (using the EXEC CICS SEND PAGE BMS commands) builds pages of output data on a temporary storage queue for subsequent display using operator page commands. (Such queues should, of course, be specified as recoverable, as described in “Implementing recoverability of temporary storage” on page 79.)

The application program should then send a committed output message to the user to say that the task is complete, and that the output data is available in the form of terminal pages.

If an uncontrolled termination occurs while the user is viewing the pages of data, those pages are not lost (assuming that temporary storage for BMS is designated as recoverable). After emergency restart, the user can resume terminal paging by using the CSPG CICS-supplied transaction and terminal paging commands. (For more information about CSPG, see the *CICS-Supplied Transactions* manual.)

Using transient data queues

When a number of tasks direct large amounts of data to a single terminal (for example, a printer receiving multipage reports initiated by the users), it may be necessary to queue the data (on disk) until the terminal is ready to receive it.

Such queuing can be done on a transient data queue associated with a terminal. A special transaction, triggered when the terminal is available, can then format and present the data.

For recovery and restart purposes:

- The transient data queue should be specified as logically recoverable by the DESTRCV=LG operand of the DFHDCT TYPE=INTRA macro.
- If the transaction that presents the data fails, dynamic transaction backout is called.

  If the terminal that the transaction runs at is a printer, however, dynamic transaction backout (and a restart of the transaction by whatever means) may cause a partial duplication of output—a situation that might require special user procedures. The best solution is to ensure that each LUW corresponds to a printer page or form.

Coping with transaction and system failures

To cope with transaction failures and uncontrolled shutdown of the system, a number of facilities are available to help ensure that:

1. Files and databases remain in a coordinated and consistent state
2. Diagnostic and warning information is produced if a program fails
3. Communication between transactions is not affected by the failure

These facilities are discussed under the following headings:

- Transaction failures
- System failures

The actions taken by CICS are described under Chapter 5, “Abend processing” on page 45 and “Processing of operating system abends and program checks” on page 51.

Transaction failures

When a transaction fails, the following CICS facilities can be invoked during and after the abend process:

- CICS condition handling
- EXEC CICS HANDLE ABEND commands, and user exit code
- The EXEC CICS SYNCPOINT ROLLBACK command
- Dynamic transaction backout (DTB)
- Transaction restart after DTB
- The program error program (DFHPEP)

These facilities can be used individually or together. During the internal design phase, specify which facilities to use and determine what additional (application or systems) programming may be involved.

The RESP option on a command returns a condition ID that can then be tested. Alternatively, an EXEC CICS HANDLE CONDITION command is used in the local context of a transaction program to name a label where control is passed if certain conditions occur.

For example, if file input and output errors occur (where the default action is merely to abend the task), you may wish to inform the master terminal operator who may decide to terminate CICS, especially if the file(s) are critical to the application.

Your installation may have standards relating to the use of RESP options or EXEC CICS HANDLE CONDITION commands. Review these for each new application.

HANDLE ABEND commands

As described in “How CICS handles transaction abends” on page 45, a HANDLE ABEND command can pass control to a routine within a transaction or a separately compiled program when the task abends.

The kind of things you might do in abend-handling code include:

- Capturing diagnostic information (in addition to that provided by CICS) before the task abends, and sending messages to the master terminal and end user.

- Executing cleanup actions, such as canceling start requests (if the PROTECT option has not been used).
- Writing journal records to reverse the effects of explicit journaling performed before the abend.

  See “Explicit journaling” on page 68.

Your installation may have standards relating to the use of EXEC CICS HANDLE ABEND commands; review these for each new application.

EXEC CICS SYNCPOINT ROLLBACK command

Before using ROLLBACK, you should understand its potential effects on your application.

ROLLBACK might be useful within your transaction if, for instance, the transaction discovers logically inconsistent input after some database updates have been initiated, but before they are committed by the syncpoint.

Before deciding to use it, however, consider the following:

- Rollback backs out updates to recoverable resources performed **in the current LUW only**—not the task as a whole.
- The EXEC CICS SYNCPOINT command (with or without the ROLLBACK option) causes a new LUW to start.
- If you have a transaction abend, and you do not want the transaction to continue processing, issue an EXEC CICS ABEND and allow dynamic transaction backout to recover the updates and ensure data integrity. Use rollback only if you want the application to regain control after nullifying the effects of a unit of work.

  For programming information about the SYNCPOINT command, see the *CICS Application Programming Reference* manual.

Dynamic transaction backout (DTB)

DTB occurs for all transactions and cannot be overridden by CEDA. (The actions of DTB are described under “Dynamic transaction backout (DTB)” on page 47.)

Remember that:

- For transactions that access a recoverable resource, DTB helps to preserve logical data integrity.
- Resources that are to be updated should be made recoverable.
- DTB takes place only after program level abend exits (if any) have attempted cleanup or logical recovery.

If you want to obtain DTB support, see Chapter 10, “Dynamic transaction backout (DTB)” on page 87.

Transaction restart after DTB

For each transaction where DTB is specified, consider also specifying automatic transaction restart. For example, for transactions that access DL/I databases (and are subject to program isolation deadlock), automatic transaction restart is usually specified. If you want to obtain support for automatic transaction restart, see “Specifying automatic transaction restart” on page 87.

Even if transaction restart is specified, a task will restart automatically only under certain default conditions (listed under “Abnormal termination of a task” on page 47). These conditions can be changed, if absolutely necessary, by editing the restart program DFHREST. Such editing must be done with care, as described in “Editing the transaction restart program (DFHREST)” on page 89.

Use of the program error program (DFHPEP)

Decide whether or not to include your own functions, examples of which are given in “Program error program (DFHPEP)” on page 121. (DFHPEP is invoked during abnormal task termination as described at “Abnormal termination of a task” on page 47.)

System failures

Specify how an application is to be restarted after an emergency restart.

Depending on how far you want to automate the restart process, application and system programming could achieve the following functions:

- User exits for transaction backout processing to handle:
  - The logical deletion of records added to DAM or VSAM-ESDS files. (See Chapter 11, “User exits for transaction backout during emergency restart” on page 91 for further information).
  - File errors during transaction backout.
  - Journal records transferred from the system log to the restart data set (DFHRSD) during emergency restart.
- A progress transaction to help the user discover what updates have and have not been performed. For this purpose, application code can be written to search existing files or databases for the latest record or segment of a particular type.

Handling abends and program level abend exits

Chapter 5, “Abend processing” on page 45 describes how CICS processes abend requests and executes program level abend exit code.

Information that is available to a program-level exit routine or program includes the following:

EXEC CICS command	Information provided
ADDRESS TWA	The address of the TWA
ASSIGN ABCODE	The current CICS abend code
ASSIGN ABPROGRAM	The name of the failing program for the latest abend
ASSIGN ASRAINTRPT	The PSW interrupt data for the latest ASRA or ASRB abend
ASSIGN ASRAKEY	The execution key at the time of the last ASRA, ASRB, AICA, or AEYD abend, if any
ASSIGN ASRAPSW	The PSW for the latest ASRA or ASRB abend
ASSIGN ASRAREGS	The general-purpose registers for the latest ASRA or ASRB abend
ASSIGN ASRASTG	The type of storage being addressed at the time of the last ASRA or AEYD abend, if any
ASSIGN ORGABCODE	Original abend code in cases of repeated abends

Notes:

1. If an abend occurs during the invocation of a CICS service, issuing a further request for the same service may cause unpredictable results because the reinitialization of pointers and work areas and the freeing of storage areas in the exit routine may not have been completed.
2. Some, but not all, ASPx abends, which are task abends while in syncpoint processing, do not cause entry to a user specified routine that handles abends.

In program-level abend exit code, you may wish to perform actions such as the following (it is recommended, however, that you keep abend exit code to a minimum):

- Record application-dependent information relating to that task in case it terminates abnormally.

  If you want to initiate a dump, do so in the exit code at the same program level as the abend. If you initiate the dump at a program level higher than where the abend occurred, you may lose valuable diagnostic information.
- Attempt local recovery, and then continue running the program.
- Send a message to the terminal operator if, for example, you believe that the abend is due to bad input data.

For transactions that are to be dynamically backed out if an abend occurs, beware of writing exit code that ends with an EXEC CICS RETURN command. This would indicate to CICS that the transaction had ended normally and would therefore prevent dynamic transaction backout (and automatic transaction restart where applicable). (See the description of program level abend processing in “How CICS handles transaction abends” on page 45.)

Exit programs can be coded in any supported language, but exit routines must be in the same language as the program of which they are a part.

See the *VSE/ESA Messages and Codes Volume 3* for the transaction abend codes for abnormal terminations that CICS initiates, their meanings, and the recommended actions.

Programming information relating to the coding of program-level exit code (such as addressability and use of registers) is in the *CICS Application Programming Reference* manual. For background information, see the *CICS Application Programming Guide*.

Processing the IOERR condition

Any program that attempts to process an IOERR condition for a recoverable resource must not issue an EXEC CICS RETURN or SYNCPOINT command, but must be terminated by issuing an EXEC CICS ABEND command. A RETURN or SYNCPOINT command would delete the dynamic log records, and commit changes to recoverable resources.

START TRANSID commands

In a transaction that uses the START TRANSID command to start other transactions, observe the following points to maintain logical data integrity:

1. Always use the PROTECT option of the START TRANSID command. This ensures that if the start-issuing task is backed out, the new task does not start.
2. Designate the temporary storage DATAID used for passing data to the started transaction as recoverable (see “Implementing recoverability of temporary storage” on page 79).

   This ensures that data passing to another task does not inadvertently stay on the temporary storage queue in the event of the start-issuing task being backed out.

   - If REQID is not used, the default DATAID is ‘DFRxxx’.
   - If REQID is used, that REQID is the DATAID designated as recoverable in the TST.

Use of a recoverable DATAID also ensures that, if a system failure occurs after the start-issuing task has completed its syncpoint, the transaction starts as soon as CICS has emergency started when the expiry time is reached and the terminal requested by TERMID (if specified) is available. Note that a DATAID is relevant only if data is being passed to the started transaction. Data is passed if FROM or FMH or RTRANSID or RTERMID or QUEUE is specified on the START command.

Enqueuing in application programs

This section describes enqueuing functions implicitly performed by CICS when transactions change:

- Recoverable files
- Recoverable transient data destinations
- Recoverable temporary storage destinations on auxiliary storage
- DL/I databases.

(The **explicit** enqueuing functions are described in “Explicit enqueuing (by the application programmer)” on page 118.)

Note: Enqueuing (implicit or explicit) on data resources protects data integrity in the event of a failure, but can affect performance if several tasks attempt to operate on the same data resource at the same time. The effect of enqueuing on performance, however, is minimized by implementing

applications with short LUWs, as discussed under “Dividing transactions into logical units of work” on page 103.

Implicit enqueuing on files

This section first describes the implicit enqueuing (exclusive control) provided while *nonrecoverable* files are being updated. It then describes the *extended* enqueuing actions when *recoverable* files are being updated.

Nonrecoverable files

For DAM files that are nonrecoverable (that is, LOG=NO is not specified on the DFHFCT macro entry), CICS itself provides no exclusive control over records that are being updated. You may specify the use of DAM exclusive control, in which case CICS will specify exclusive control on an EXEC CICS READ UPDATE request, and release control either on the associated EXEC CICS REWRITE or UNLOCK command, or at syncpoint.

For nonrecoverable VSAM files, VSAM locks the control interval during an update.

Figure 4 illustrates the extent of exclusive control for nonrecoverable files. Two tasks are shown updating the same record or control interval. Task A is given exclusive control of the record or control interval between the READ UPDATE and REWRITE commands. During this period, task B waits.

Figure 5 illustrates two tasks updating the same record or control interval. Task A is given exclusive control of the record until the update is committed (at the end of the LUW). During this period, task B waits.

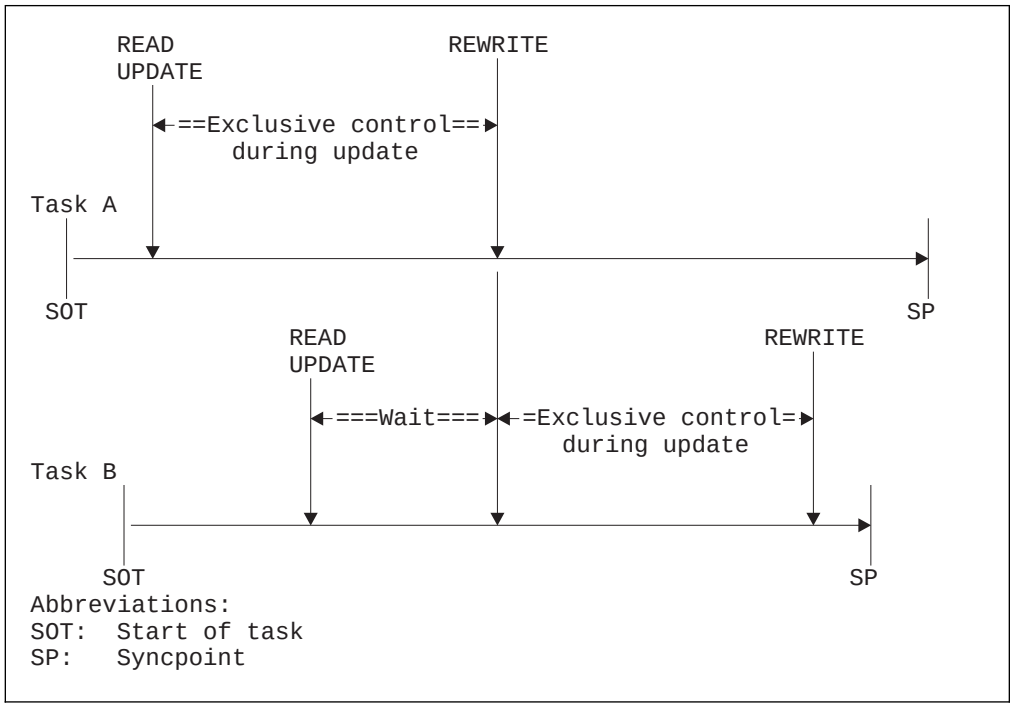

Figure 4. Exclusive control during updates to nonrecoverable files

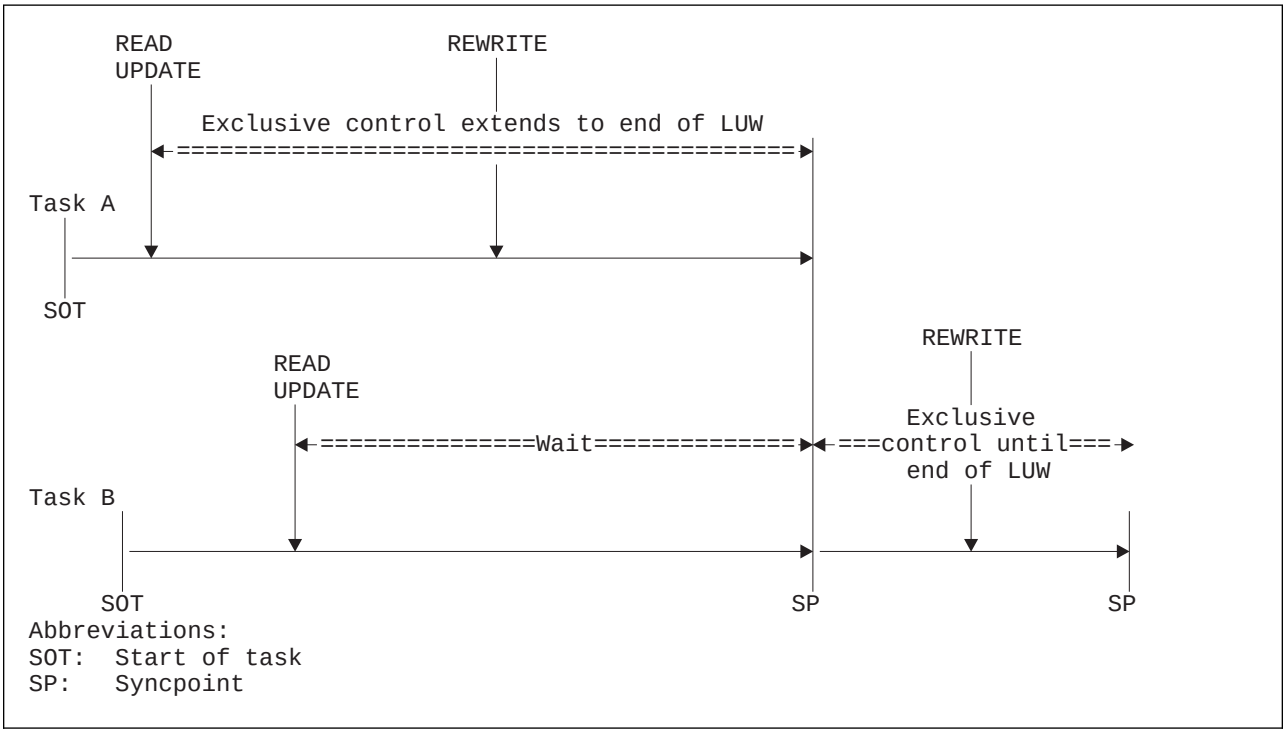

Figure 5. Enqueuing (exclusive control) during updates to recoverable files

Recoverable files

For VSAM or DAM files designated as recoverable, CICS extends the duration of its enqueuing action as shown in Figure 5. For VSAM files, the extended enqueuing is on the updated record only, not the whole control interval.

The extended period of exclusive control is needed to avoid an update committed by one task being backed out by another task. Consider what could happen if the nonextended exclusive control shown in Figure 4 on page 114 was used when updating a *recoverable* file. If task A abends just after task B has reached syncpoint and has thus committed its changes, the subsequent backout of task A returns the file to the state it was in at the beginning of task A, and task B's committed update is lost.

To avoid this problem, whenever a transaction issues a command that changes a recoverable file (or reads from a recoverable file prior to update), CICS automatically enqueues the task to the updated record until the change is committed (that is, until the end of the LUW). Thus in the above example, Task B would not be able to access the record until Task A had committed its change at the end of the LUW. Hence, it becomes impossible for Task B's update to be lost by a backout of Task A.

The file control EXEC CICS commands that invoke automatic enqueuing in this way are:

- READ (for UPDATE)
- WRITE
- DELETE

Notes:

1. Enqueuing as described above can lead to *transaction deadlock* (see “Possibility of transaction deadlock” on page 119).
2. The spheres of CICS exclusive control are the physical block for DAM data sets and the VSAM record for VSAM data sets.

   If a transaction requests a record for update that is within the sphere of control of another record being updated, the second task is queued until the first update is complete.
3. *VSAM exclusive control*. The CICS enqueuing action on recoverable files, which always lasts until the end of the LUW, does not, of course, affect VSAM's exclusive control actions. When a transaction issues an EXEC CICS READ UPDATE command (for any file, recoverable or not), VSAM maintains its exclusive control of the control interval containing the record until an EXEC CICS REWRITE (or UNLOCK or DELETE or SYNCPOINT) command is issued. Two READ UPDATE commands for records in the same control interval without an intervening REWRITE command will raise the INVREQ condition.
4. For recoverable files, do not use unique key alternate indexes (AIXs) to allocate unique resources (represented by the alternate key). If you do, backout may fail in the following set of circumstances:

   a. A task deletes or updates a record (through the base or another AIX) and the AIX key is changed.

   b. Before the end of the first task's LUW, a second task inserts a new record with the original AIX key, or changes an existing AIX key to that of the original one.

   c. The first task fails and backout is attempted.

   The backout fails because a duplicate key is detected in the AIX. There is no locking on the AIX key to prevent the second task taking it before the end of the first task's LUW. If there is an application requirement for this sort of operation, you should use the CICS enqueue mechanism to reserve the key until the end of the LUW.
5. To ensure that the data being read is up to date, the application program should issue a READ UPDATE command (rather than a simple READ), thus enqueuing on the data until the end of the LUW.

Implicit enqueuing on logically recoverable transient data destinations

CICS provides an enqueuing protection facility for logically recoverable (as distinct from physically recoverable) transient data destinations in a similar way to that for recoverable files. There is one minor difference, however — CICS regards each recoverable destination as two separate recoverable resources—one for writing and one for reading.

Transient data control commands that invoke implicit enqueuing are:

- EXEC CICS WRITEQ TD
- EXEC CICS READQ TD
- EXEC CICS DELETEQ TD

Thus, for example:

- If a task issues an EXEC CICS WRITEQ TD command to a particular destination, the task is enqueued upon that write destination until the end of the task (or LUW). While the task is thus enqueued:
  - Another task attempting to write to the same destination is suspended.
  - Another task attempting to read from the same destination is allowed to read only *committed* data (not data being written in a currently incomplete LUW).
- If a task issues an EXEC CICS READQ TD command to a particular destination, the task is enqueued upon that read destination until the end of task (or LUW). While the task is thus enqueued:
  - Another task attempting to read from the same destination is suspended.
  - Another task attempting to write to the same destination is allowed to do so and will itself enqueue on that write destination until end of task (or LUW).

Implicit enqueuing on recoverable temporary storage queues

CICS provides the enqueuing protection facility for recoverable temporary storage queues in a similar way to that for recoverable files on VSAM data sets. There is one minor difference, however: CICS enqueuing is not invoked for EXEC CICS READQ TS commands, thereby making it possible for one task to read a temporary storage queue record while another is updating the same record. To avoid this, use explicit enqueuing on temporary storage queues where concurrently executing tasks can read and change queues with the same temporary storage identifier. (See “Explicit enqueuing (by the application programmer)” on page 118.)

Temporary storage control commands that invoke implicit enqueuing are:

- EXEC CICS WRITEQ TS
- EXEC CICS DELETEQ TS

Implicit enqueuing on DL/I VSE databases

There are two distinct cases—program isolation scheduling and intent scheduling. Each is discussed separately in the sections that follow.

Program isolation scheduling

When a task accesses a segment by a DL/I VSE database call, it implicitly enqueues on all segments in the same database record as the accessed segment. The duration of enqueuing depends on the access method being used:

- **Direct methods (HDAM, HIDAM)**—If an ISRT, DLET, or REPL call is issued against a segment, that segment, with all its child segments (and, for a DLET call, its parent segments as well), remains enqueued upon until a DL/I TERM call is issued. The task dequeues from all other segments in the database record by accessing a segment in a different database record.
- **Sequential methods (HSAM, HISAM, SHISAM)**—If the task issues an ISRT, DLET, or REPL call against any segment, the entire database record remains enqueued upon until a DL/I TERM call is issued. If no ISRT, DLET, or REPL call is issued, the task dequeues from the database record by accessing a segment in a different database record.

The foregoing rules for program isolation scheduling can be overridden using the ‘Q’ command code in a segment search argument (this command extends

enqueuing to the issue of a DL/I TERM call), or by using PROCOPT=EXCLUSIVE in the PCB (this macro gives exclusive control of specified segment types throughout the period that the task has scheduled the PSB).

Intent scheduling

When a task issues a DL/I VSE scheduling call, it is interpreted as intending to update all the segments it is possible to update under the specified PSB. Therefore, until a DL/I VSE TERM call is issued, no other task is allowed to schedule a PSB that would permit updating of any of the segments scheduled for update by the first task.

Application programming note

This section describes the DL/I VSE TERM call, but you are advised to use EXEC CICS SYNCPOINT or EXEC CICS RETURN commands instead of DL/I VSE TERM calls. These make the program logic clearer.

A DL/I VSE TERM call commits DL/I VSE updates and ends implicit enqueuing as described above. It also causes an implicit SYNCPOINT command to be issued. This terminates the LUW, and thus commits all non-DL/I VSE updates as well. An explicit EXEC CICS SYNCPOINT command (or EXEC CICS RETURN command in the last or only LUW of a task) would have exactly the same effect for both DL/I VSE and non-DL/I VSE resources.

The application programmer must be aware of the implications of issuing a DL/I VSE TERM call. It signals end-of-LUW to CICS. This means that not only all DL/I VSE updates but all related updates to non-DL/I VSE resources are regarded as logically complete. Therefore, they are not eligible for backout if CICS or the transaction should subsequently abend.

Even if the programmer recognizes this and writes a correct program, the possibility remains that the logic may not be understood by a different programmer maintaining the code.

Explicit enqueuing (by the application programmer)

CICS provides the following explicit enqueuing commands:

- EXEC CICS ENQ RESOURCE
- EXEC CICS DEQ RESOURCE

These commands can be useful in certain applications when, for example, you want to:

- Protect data written into the common work area (CWA), which is not automatically protected by CICS
- Prevent transaction deadlock by enqueuing on records that might be updated by more than one task concurrently
- Protect a temporary storage queue from being read and updated concurrently.

To be effective, however, all transactions must adhere to the same convention. A transaction that accesses the CWA without using the agreed ENQ and DEQ commands is not suspended, and protection is violated.

After a task has issued an EXEC CICS ENQ RESOURCE(data-area) command, any other task that issues an ENQ RESOURCE command with the same data-area

parameter is suspended until the task issues a matching EXEC CICS DEQ RESOURCE(data-area) command, or until the LUW ends.

Note: The concurrent use of enqueues against more than one resource introduces the possibility of transaction deadlock.

Possibility of transaction deadlock

The enqueuing and program isolation scheduling mechanisms, which protect resources against double updating, can cause a situation known as **transaction deadlock.**[5]

As shown in Figure 6, transaction deadlock means that two (or more) tasks cannot proceed because each task is waiting for the release of a resource that is enqueued upon by the other. (The enqueuing or DL/I program isolation scheduling action protects resources until the next synchronization point is reached.)

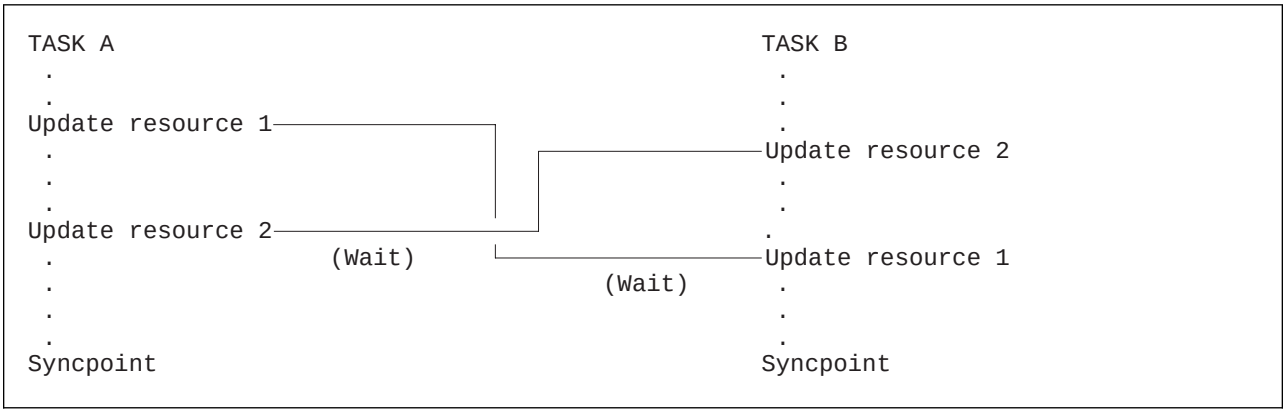

Figure 6. Transaction deadlock (generalized)

If transaction deadlock occurs, one task abends and the other proceeds. Which deadlocked task abends depends primarily on the resource types involved in the deadlock:

- If both resources are CICS resources (that is, non-DL/I), the task whose DTIMOUT period elapses first is abended. (It is possible for both tasks to time out simultaneously.) If neither task has a DTIMOUT period specified, they both remain suspended indefinitely unless the master terminal operator cancels one of them.
- If one resource is a DL/I database and the other is a CICS resource, the task using the CICS resource abends after its DTIMOUT period has elapsed. If DTIMOUT is not specified for the task using the CICS resource, both tasks remain suspended indefinitely unless one is canceled by the master terminal operator.
- If the resources are both DL/I databases (and program isolation scheduling is being used), DL/I itself detects the potential deadlock as a result of the tasks issuing their scheduling calls, and abends the task that has less update activity.

The abended task may then be backed out by dynamic transaction backout, as described in “Dynamic transaction backout (DTB)” on page 47. (Under certain

[5] Transaction deadlock is sometimes known as **enqueue deadlock**, **enqueue interlock**, or **deadly embrace**.

conditions, the transaction can be automatically restarted, as described under “Abnormal termination of a task” on page 47. Alternatively, the terminal operator may restart the abended transaction.)

For more information, see “Designing to avoid transaction deadlock” on page 106.

Chapter 14. Using a program error program (DFHPEP)

This chapter describes aspects of coding the program error program (DFHPEP). The way this program works, and some design considerations for it, are described in “Actions taken at abnormal task termination” on page 50.

This chapter contains Product-sensitive Programming Interface information. For programming information to complement the information in this book, see the *CICS Customization Guide*, which contains detailed advice on writing these error programs.

Program error program (DFHPEP)

As described on page 50, the program error program (PEP) gains control after all program-level ABEND exit code has executed and after dynamic transaction backout has been performed. The PEP can be:

- Omitted entirely
- The CICS-supplied PEP
- Your own PEP created by editing the CICS-supplied version.

Omitting the PEP

The CICS-supplied PEP is included in the pregenerated system. The CICS abnormal condition program, however, will not link to it if no program resource definition for DFHPEP is installed. If CICS cannot link to DFHPEP (for this or any other reason), it sends a DFHAC2259 message to CSMT.

The CICS-supplied PEP

If the PEP is included in your system, use the CEDA INSTALL command to install the CICS-supplied group, DFHMISC, which contains the PEP.

The CICS-supplied PEP performs no processing. The only effect of including DFHPEP is to suppress the DFHAC2259 message when you link to the PEP.

Your own PEP

During the early phases of operation with CICS, the master terminal commands can put abending transactions into disabled status while the cause of the abend is being investigated and corrected.

Where a program needs to handle this process, or where associated programs or transactions should also be disabled, you may decide to incorporate your own PEP. This will depend on the importance of the applications being served.

The program error program is a command-level program that can be written in any of the languages that CICS supports. The CICS abnormal condition program passes, to the PEP, a COMMAREA containing information about the abend. Add code to take actions appropriate to your installation.

Functions you might consider including in a program error program include:

- Disabling a particular transaction identifier (to prevent other users using it) pending diagnostic and corrective actions. This would avoid the need for a master terminal operator command and the risk of several more abends in quick succession.
- Disabling other transactions or programs that depend on the satisfactory operation of a particular program.
- Keeping a count of errors by facility type (transaction or file).
- Abending CICS after a transaction abend. Conditions for this might be:
  - If the abended transaction was working with a critical file.
  - If the abended transaction was critical to system operation.
  - If the abend was such that continued use of the application would be pointless, or could endanger data integrity.
  - If the error count for a particular facility type (transaction or file) reached a predetermined level. (An alternative to abending CICS in this context would be to disable the facility, which would keep the system running longer.)

Note: CEMT SET TRDUMPCODE or EXEC CICS SET TRANDUMPCODE is a simpler way of doing this.

If a task terminates abnormally (perhaps because of a program check or an ABEND command), code in a program-level exit or the PEP can flag the appropriate transaction code entry in the installed transaction definition as disabled. CICS will reject any further attempt by terminals or programs to use that transaction code until it is enabled again. Consequently, the effect of program checks can be minimized, so that every use of the offending transaction code does not result in a program check. Only the first program check is processed. If the PEP indicates that the installed transaction definition is to be disabled, CICS will not accept subsequent uses of that transaction code.

Following correction of the error, the master terminal operator can enable the relevant installed transaction definition for the transaction code to allow terminals to use it. The master terminal operator can also disable transaction codes when transactions are not to be accepted for application-dependent reasons, and can enable them again later. The *CICS-Supplied Transactions* manual tells you more about the master terminal operator functions.

If logic within DFHPEP determines that it is unsafe to continue CICS execution, you can force a CICS abend by issuing an operating system ABEND macro. If DFHPEP abends (transaction abend), CICS produces message DFHAC2263.

Chapter 15. Using message caches after emergency restart

This chapter describes how an inquiry program that is run after an emergency restart can use the contents of message caches. A message cache is a temporary storage queue with a DATAID of DFHMXXXX, where XXXX is the identification of the logical unit. The inquiry program should be able to help a terminal operator determine whether the last piece of work before system failure completed, or if it backed out during emergency restart.

Note: The information in this chapter relates only to transactions that work with VTAM terminals *and* have the PROTECT option specified. See "Specifying message-protection options for VTAM terminals" on page 81 for details of this option.

Using message caches after emergency restart is discussed under the following headings:

Logic of inquiry program

The inquiry program should inspect the message cache[6] for the inquiring terminal by issuing a READQ TS command, using the queue name DFHMXXXX, where XXXX is the 4-character identifier of the inquiring terminal. When an INQUIRY program is run:

- If the terminal had *no* in-flight task at the time of uncontrolled shutdown, a QIDERR error condition is returned to the program. (For programming information, see the *CICS Application Programming Reference* manual.)
- If the terminal does have an in-flight task, one or more temporary storage records will be returned to the program from the message cache.

The contents of the temporary storage records from the message cache will depend on *when* the uncontrolled shutdown occurred in relation to message logging (see "Interpreting the contents of a message cache" on page 124).

If a record contains an input message, the inquiry program should present that input message and associated information to the terminal operator. The terminal operator can then decide whether to request CICS to reprocess the transaction.

The inquiry program should allow a request for reprocessing to proceed only if the
2 terminal operator has the necessary authority (based on CICS transaction attach
2 security or operator class of the signed-on user). Processing could then take place as if the transaction request had just been entered.

6 During emergency restart, logged messages are copied from the restart data set into message caches, as described on page 38.

Notes:

1. To identify the type of message in a message cache, see “Message cache records” on page 127.
2. Assuming that the message cache temporary storage queues are recoverable, there may be messages for more than one task in the cache. It is recommended that you delete a message cache immediately after use — you can do this with an EXEC CICS DELETEQ TS command in the inquiry program.

   If there are records for more than one task in the message cache, the inquiry program should check the JCSPTASK field (DSECT DFHJCRDS), which contains the task number. (For programming information about journal fields, see the *CICS Customization Guide*.)
3. If the input message is associated with a VTAM programmable controller, the inquiry program can be automatically initiated by the controller after message resynchronization and recovery have completed. The in-flight input message (transmitted back to the controller by the inquiry program) may be presented automatically to the relevant terminal operator for a decision whether or not to reprocess. Alternatively, if application and security considerations permit, the controller may automatically make the decision whether or not to reprocess, and notify the inquiry program.
4. For further information about operator classes and CICS transaction attach security, see the *CICS Security Guide*.

Interpreting the contents of a message cache

This section describes the CICS message protection mechanism to help you interpret the contents of a message cache after emergency restart. For example:

- Table 7 on page 125 shows the main actions performed by CICS during the execution of a single-LUW message-protected transaction. After an output message has been logged in the syncpoint records, the output message is said to be committed—that is, CICS preserves the message in case the system fails. A committed output message is said to be in doubt until a positive response to the message has been received and logged.
- Table 8 on page 126 shows the result of emergency restart processing (in terms of what can appear in a message cache) following an uncontrolled shutdown at different points during the task’s execution. The step numbers in the first column of this figure refer to the step numbers in the previous figure.

These figures show that a message cache (if there is one) can contain either an input message or an in-doubt output message. These, and other combinations of records that can appear in a message cache after emergency restart are listed below—together with a possible interpretation of each one. (These interpretations, plus the message texts, should enable you to design programs that resume processing and communication.)

Case 1: A single initial input message: This indicates that:

- The task that received the input message was in-flight at the time of the uncontrolled shutdown. Therefore, the interrupted LUW backed out during the emergency restart processing.

- The task that received the input message:
  - Was executing its first LUW; or
  - Had completed a prior LUW from which there was no committed output message; this is typical of input-only tasks with multiple LUWs.

Resynchronization (see “Resynchronization and re-presentation of VTAM messages” on page 41) uses the sequence numbers established **before** the input message was received. These are the sequence numbers pertaining after the successful completion of a prior LUW in this task or of an earlier message-protected task working with the same logical unit.

Table 7. CICS actions during execution of a single-LUW message-protected task

Application Action	**CICS Action**
	Step 1: Receive first input message
	Step 2: Initiate task
Start	
.	
.	Step 3: Log first input message (on system log)
Read input message	
.	
.	
.	
Write output message	Step 4: Defer transmission of output message until syncpoint records are written on system log
.	
.	
.	
.	
End	Step 5: Process CICS-supplied syncpoint
	Step 6: Put syncpoint records (including text on output message) on system log. (Output message is now committed but in doubt.)
	Step 7: Transmit output message with definite response requested
	Step 8: Receive definite response to output message
	Step 9: Record definite response on system log. (Committed output message is now not in doubt.)

Table 8. Contents of message cache after emergency restart for a single-LUW message-protected task	
Time of uncontrolled shutdown	**CICS emergency restart action on message cache**
Before first input message is logged (before step 2 is complete).	No action
After first input message is logged and before syncpoint records are logged (before step 6 is complete).	CICS puts first input message into the message cache. See case 1 on page 124.
After syncpoint records are logged and before definite response to output message is logged (before step 9 is complete).	CICS puts output message into the message cache with in-doubt indicator on in the system prefix. See case 2.
After definite response to output message is logged (after step 9 is complete).	No action

Case 2: A single committed, but in-doubt, output message: This indicates that:

1. A positive response to the output message has not been logged. This means that, *at the time of the uncontrolled termination*, the output message may or may not have been delivered.
2. The LUW that issued the message has completed, and is therefore not subject to backout.
   - If the LUW was the last (or only) LUW of the task, the task is known to be complete.
   - If the LUW was not the last LUW of the task, the task will not have started a new LUW. (It will have been waiting for the positive response to the output message before proceeding.)

Resynchronization uses the sequence numbers established when the output message was originally sent. This message is also copied to the resend slot, and CICS uses it if, after resynchronization, the VTAM terminal has not received the message.

Case 3: An initial input message followed by a committed not-in-doubt output message: This indicates that:

1. The task that received the input message was in-flight at the time of the uncontrolled shutdown. Therefore, the interrupted LUW backed out during the emergency restart processing.
2. The task had completed a prior LUW that issued an output message whose delivery had been confirmed.

Resynchronization uses the sequence numbers established at the time when the response to the committed output message was logged.

Case 4: A single committed not-in-doubt output message: This indicates that:

1. A positive response to the output message has been logged. Delivery of the output message is thus confirmed.
2. The LUW that issued the message has completed, and is therefore not subject to backout.

3. The task that issued the message has not completed and might have started a new LUW. Further:
   - The work (if any) of this new LUW will have been backed out.
   - The new LUW has not requested any terminal input; this is typical of output-only tasks with multiple LUWs.

Resynchronization uses the sequence numbers established when the positive response to the committed output message was logged.

Message cache records

Records copied to the message cache have the same layout as journal records.

Input and output messages in a message cache have different values in the 2-byte JCRSTRID field:

- For **input messages**, the value of JCRSTRID is X'C110' or X'C510'.
- For **output messages**, the value of JCRSTRID is X'F110' or X'F210'.

  If an output message in the message cache is in doubt, the JCSPMIDT flag is set on.

The names JCRSTRID and JCSPMIDT refer to the DSECT called DFHJCRDS.

For programming information about the layout of journal records, see the *CICS Customization Guide*.

Chapter 16. Backout failure

This chapter describes the actions that occur after a backout failure.

CICS handles backout failure in the same way, whether the failure occurs during DTB or during backout processing in emergency restart.

When the backing out of uncommitted changes to a data set fails, CICS:

- Sets the backout status field in the CICS base cluster block (one for each base cluster) to “failed”.
- Stores a backout-failed log record on the system log, to enable a backout utility to start and stop its scan of the log in the correct places, and to locate the relevant before-images.
- Sets a backout-failed status record in the global catalog.

In these ways, CICS can maintain the backout-failed status across all types of start, including a cold start. CICS issues a backout-failing log record (BOFLGREC) the first time a backout failure is detected. This BOFLGREC indicates that this is the first combination of file and task to detect a backout failure. CICS issues subsequent BOFLGRECs if the same task suffers a backout failure via a different file or if a different task suffers a backout failure. There is therefore a BOFLGREC for each combination of file and task that has failed backout. A BOFLGREC is also issued when all files relating to the failure have been closed.

In addition, to preserve data integrity, CICS closes all files that are open against the base cluster, and protects files in the following ways:

- If a transaction using a file referring to the data set attempts an update after the base cluster has been flagged as backout failed, CICS abends the transaction.
- For transactions trying to become new users of a file referring to the data set, CICS returns a NOTOPEN response code.
- If an attempt is made using CEMT to explicitly open a file referring to the data set, CICS returns a backout-failed response. For EXEC CICS SET FILE OPEN requests, it returns an INVREQ response with a RESP2 value of 15.

When CICS informs the operator of a backout failure, the operator should check, using CEMT INQUIRE DSNAME FAILED, that no other backout-failure processing is in progress. When all current backout failure processing is complete, the operator must switch the system log and archive the now-inactive log data set, so that it may be used by a backout utility. (Automatic archiving makes archiving easier and less prone to error—see “Preserving the system log (automatic archiving)” on page 65.)

A backout utility may now be run, using the archived log (or logs), the failed data set and user-provided JCL. The operator can find out the data set names to insert in the JCL by using the CEMT INQUIRE DSNAME FAILED command. It is essential to keep good records of the archived log data sets, so that there is no delay in creating the JCL and running a backout utility.

After the backout utility has been run, the operator must reset the status of the data set by using the CEMT SET DSNAME NORMAL transaction (see the *CICS-Supplied Transactions* manual).

Chapter 17. Operations

This chapter describes operations activities related to recovery and restart.

Time required for forward recovery and emergency restart

Estimate the time likely to be needed for forward recovery of the largest data set and an emergency restart. Compare this period with the allowed amount of downtime discussed under “Question 8: How long can the business tolerate being unable to use the application in the event of a failure?” on page 58.

By ensuring that the user has standby procedures (see “Question 9: How is the user to continue or restart entering data after a failure?” on page 58), it may be possible to negotiate a longer downtime for exceptional conditions.

Daily and weekly schedules

Specify the planned timetable of systems use (online and offline operations).

If the system is active for almost 24 hours a day, allow sufficient time for offline housekeeping operations needed for recovery, such as taking backup copies, checking their usability, extracting forward recovery information from CICS journals and logs, and merging such information with similar information from other sources.

If the system is active for 24 hours per day, consider the need to take data sets offline to make backup copies, or else schedule housekeeping operations for a day when the system is not in use.

Check the above timetable again when more detailed design work has been done.

Offline recovery

VSAM data sets and DL/I VSE databases may be taken offline for recovery activities while CICS continues to run. In this way, unaffected CICS users can continue to work normally. For information about the VSAM recovery utilities, see Chapter 16, “Backout failure” on page 129 and, for DL/I dynamic allocation support, see Chapter 19, “Recovery in a DL/I VSE environment” on page 139. Operators should be well-practiced in offline procedures so that recovery is not delayed.

Chapter 18. Report controller recovery

The report controller is a separately orderable feature of *CICS Transaction Server for VSE/ESA*. Before you read this chapter about recovery, you should have read the information about the report controller in the *CICS Report Controller Planning Guide*.

The recovery processing described in this chapter is provided by CICS. You make choices concerning recovery in the EXEC CICS spool commands, and by operator actions, for example, switching from a failed printer.

If you use the interface to POWER, without the report controller, you have logical recovery only (as explained below). However, with the report controller, you have the ability to specify further recovery options, as described in this chapter.

A terminal error causes a link to the user-replaceable node error program (NEP) or terminal error program (TEP) running for that terminal. If the transaction CEPW is running on that terminal, a link is then made to the report controller NEP or TEP. This report controller code attempts to prevent CEPW from being abended, and the terminal being put out of service. The report controller code is not replaceable.

Failures may occur during report creation or during report printing. Most of these failures are detected, and you may be able to recover from them. In this chapter, first the types of failure are considered, and then, on page 134, a description is given of how recovery from those failures is achieved.

Types of report controller failure

The main areas of concern are:

- CICS printer failures
- CICS transaction abends and CICS abends
- POWER abends
- VSE system abends.

CICS printer failures

CICS printers may be initiated from POWER or CICS, and in both cases printer errors are handled by CICS.

Printing failures may be due to such things as:

- Printer faults
- Access method failures
- Report data stream errors
- Operator intervention requesting that printing be stopped while a report is still printing.

The recovery restart action depends upon the type of failure and the recovery options specified for the report.

CICS transaction abends and CICS abends

If CICS abends, the XPCC link is broken, and POWER sets the status of any open report on its queues to indicate an error situation. CICS may modify this status during emergency restart. If CICS is cold started, the status of any report is not modified. For both CICS abends and transaction abends, the recovered report status depends on the operation in progress at the time of failure, and on the report options used:

- During report creation, the type of recovery specified (LOGICAL or PHYSICAL) determines the state of the unfinished report.
- During report processing, the PRINTFAIL option specifies that further action is required by an operator before processing can continue.

For information about XPCC and POWER dispositions, see the *VSE/POWER Administration and Operation* manual.

POWER abend

In the case of a POWER abend, two situations must be considered:

- CICS running under POWER
- CICS not running under POWER.

In the first case, the failure of POWER causes CICS to abend at the same time.

In the second case, CICS remains operational, but the report controller is disabled.

Note: POWER must be warm started for reports to be maintained on the POWER queue.

VSE system abend

A VSE system abend brings down CICS and POWER. The actions taken at the restart of VSE, POWER, and CICS depend on system parameters and operator action.

To initiate recovery, both POWER warm start and CICS emergency restart should be utilized.

Recovering from failures

The report controller uses the EXEC CICS SYNCPOINT command, in conjunction with POWER checkpointing and report options, to effect recovery during report creation and printing.

To provide full protection against a CICS or VSE/ESA abend while printing a report, you must ensure that any disk journal is large enough to cover the time taken to print the largest report.

Recovery from failures during report creation

On the EXEC CICS SPOOLOPEN REPORT command, you can specify either LOGICAL or PHYSICAL recovery, with or without the PRINTFAIL option:

With LOGICAL recovery, the report is created in LUWs.

- You must issue an EXEC CICS SPOOLCLOSE REPORT command to commit a report to the POWER queue, or issue an EXEC CICS SYNCPOINT command which implies an EXEC CICS SPOOLCLOSE REPORT command. If you issue a EXEC CICS SPOOLCLOSE RESUME REPORT command before the EXEC CICS SYNCPOINT command, you can issue an EXEC CICS SPOOLOPEN RESUME REPORT command after the EXEC CICS SYNCPOINT command, to continue writing the report.
- If the transaction writing the report fails, the report lines are backed out to the last EXEC CICS SPOOLCLOSE REPORT command.

For the different types of report, the logical recovery characteristics are shown in Table 9.

Table 9. LOGICAL recovery from failures during report creation

Time of failure	Standard	Resumable	Log
Before report is closed or before syncpoint.	Report is deleted.	Records added to report since last SPOOLCLOSE RESUME REPORT are deleted and report is closed with DISP=A. If no previous SPOOLCLOSE RESUME REPORT, the report is deleted.	N/A
After report syncpoint.	All records written are committed to report and report closed with DISP=D.	All records written are committed to report and report closed with DISP=D.	N/A

With PHYSICAL recovery, every line written is committed to the report. The more frequent checkpointing is an overhead to be weighed against the enhanced recovery provided.

For the different types of report, the physical recovery characteristics are shown in Table 10 on page 136.

Table 10. PHYSICAL recovery from failures during report creation			
Time of failure	**Standard**	**Resumable**	**Log**
Before report is closed or before syncpoint.	All records written are committed to report. If failure due to user transaction abend, report closed with DISP=D. If failure due to CICS abend, report closed with DISP=X.	All records written are committed to report. If failure due to user transaction abend, report closed with DISP=D. If failure due to CICS abend, report closed with DISP=X.	All records written are committed to report. If failure due to user transaction abend, report stays open. If failure due to CICS abend, report closed with DISP=X.
After report syncpoint.	All records written are committed to report and report closed with DISP=D.	All records written are committed to report and report closed with DISP=D.	All records written are committed to report. If failure due to user transaction abend, report stays open. If failure due to CICS abend, report closed with DISP=X.

Recovery from failures during printing

Printing failures are categorized as either:

- Severe failures - which cause the CEPW transaction (also called the report writer task) to terminate and thus affect all reports
- Less severe failures - which affect only one report or which can be corrected by simple operator intervention.

Severe failures

On detection of a severe printing failure, if the CEPW transaction is not forced to abend immediately, the report controller sends a message to the transient data queue CSPW, and, because further processing by the CEPW transaction is pointless, abends the CEPW task. Processing of other reports by this printer are suspended until CEPW is restarted.

When specified on a report, the PRINTFAIL option specifies that CICS cannot attempt recovery during emergency restart or dynamic transaction backout, but is to leave the report in an ERRPRT status (DISP=Y). After CEPW is restarted, operator intervention is required before printing of the report can continue. If PRINTFAIL is not specified, the report is reset to its original disposition.

You can use the PRINTFAIL option to avoid the risk of printing a document (such as an invoice) twice.

Note: When you use the EXEC CICS SPOOLOPEN INPUT command to read a report, recovery depends on the options specified at report creation. If PRINTFAIL is specified, the report is set to DISP=Y. If PRINTFAIL is not specified, the report is reset to its original disposition.

The end user can usually recover from such a failure by redirecting the inflight report to an alternative printer.

Less severe failures

For nonsevere printing failures:

- Printing of the report may continue after the error is cleared.

  For errors such as “out-of-paper”, when printing may eventually continue, the condition is detected and CEPW is suspended with a message sent to the transient data queue CSPW. Operator intervention is required to clear the error, after which the operator resumes CEPW. Recovery in this instance means that the print run restarts at the correct page, and that the correct number of copies is printed.

- Printing of the report may not continue.

  For errors which prevent printing from continuing, CEPW logs the failure to CSPW. An example of such an error is the sending of a report to a CICS terminal printer that fails a CICS security check. The report is held with an ERRPRT status (DISP=Y) on the POWER report list. Processing of other reports continues.

  For more information about security for RCF, see the *CICS Report Controller Planning Guide*.

Note: If you have multiple printers for a destination, and one printer produces faulty reports, or fails to produce reports at all, you can look at the audit trail on the transient data queue CSPA. CSPA tells you which reports were printed (or should have been printed) by each printer.

ESCAPE format report processing failure

The processing of ESCAPE reports requires an escape routine to receive control at print time.

CEPW reads the report into a temporary storage queue, places the temporary storage queue name into the communication area, and links to the escape routine. The communication area is 80 bytes in length, with the queue name in the first 8 bytes. CEPW expects a return code in byte 0. The remaining 79 bytes may be used to pass a message to transient data queue CSPW.

With a return code of 0, the report is held or deleted according to the report status. If the return code is not 0, or if the escape routine is not available at processing time, an error message is sent to transient data queue CSPW. The report is held with ERRPRT status (DISP=Y) on the POWER report list. In either event, processing of other reports continues.

An abend in the escape routine abends CEPW.

MAP format report processing failure

The processing of MAP reports requires that BMS maps, specified in the reports, are available.

If the map is not available at processing time, an error message is sent to the transient data queue CSPW. The report is held with ERRPRT status (DISP=Y) on the POWER report list. Processing of other reports continues.

Other failures

Some other types of report failure are not handled by CICS. Two examples are described below.

JCL format processing failure

JCL type reports are held and processed on the POWER reader queue as jobs to VSE. Failure of this type of report results in normal VSE job error handling.

Failure with non-CICS printers

Reports printed on VSE/POWER system printers are not controlled or monitored by CICS. System print failures are handled by VSE/POWER.

Chapter 19. Recovery in a DL/I VSE environment

This chapter describes how to implement DL/I VSE recovery and some of the processes that handle the relationship between CICS and DL/I VSE.

The information is divided as follows:

- "Use of DL/I VSE"
- "Design factors"
- "Implementing recoverability of DL/I VSE databases" on page 140
- "DL/I VSE error processing" on page 141.

You should consult your IBM® representative on the availability in your area of relevant IBM System Center documents. Like all System Center documents, they are written by people with experience of the situations you are likely to encounter.

For information about DL/I VSE database I/O error handling within a CICS environment, see the *DL/I DOS/VS Version 1 Release 11 Release Guide*. The DLIOER system initialization parameter is described in the *CICS System Definition Guide*.

Use of DL/I VSE

DL/I VSE offers the following advantages:

- DL/I VSE has benefits when databases are to be accessed by more than one application; data does not need to appear several times in the database even though the data might need to be retrieved in several different ways for various applications.
- DL/I VSE enables online database information to be shared by batch programs.
- When data resources are all on DL/I VSE, and assuming that program isolation scheduling is used, CICS and DL/I VSE combine to handle transaction deadlocks automatically.
- CICS and DL/I VSE automatically record both before- and after-images without the need for user journaling. DL/I VSE provides forward recovery utilities.

Design factors

A design factor relating to recovery and restart is the choice of scheduling method: program isolation scheduling or intent scheduling.

With program isolation scheduling, protection against multiple updating applies to specific occurrences of a segment type; with intent scheduling, protection applies to all segments of a given segment type. With both types of scheduling, protection against multiple updating lasts until the end of the LUW that issued the scheduling call.

Program isolation scheduling is the method usually chosen because it can lead to faster scheduling, better throughput, and faster response times.

DL/I VSE provides the facility to detect impending deadlock between DL/I VSE requests. A request that would result in a deadlock causes one of the tasks to abend.

Assuming that program isolation scheduling is to be used:

- Any transaction that accesses DL/I VSE resources might result in a program isolation deadlock. For this reason, you are advised to make all such transactions capable of dynamic transaction backout (DTB). You can also make them start automatically after DTB, in which case, they should:
  1. Contain only one LUW.
  2. Not perform any terminal activity until all database accesses and updates within the LUW have completed. This ensures that the default conditions required for automatic transaction restart are not violated.
- A request from a program to terminate a DL/I VSE PSB implies an EXEC CICS SYNCPOINT, which commits both DL/I and non-DL/I VSE changes.
- Transactions that update both DL/I VSE and non-DL/I VSE resources should always access the resources in the same sequence. In this way you avoid the possibility of a transaction deadlock between DL/I VSE and non-DL/I VSE resources.

For application programming considerations, see “Implicit enqueuing on DL/I VSE databases” on page 117.

3 If batch programs and online programs are to use a DL/I VSE database concurrently (by means of Multiple Partition Support (MPS), make checkpoint requests at appropriate intervals (seconds, rather than minutes). Frequent checkpoints:

- Minimize the risk that the DL/I VSE enqueue pool will fill and cause failure
- Help avoid unnecessary delays in response to users.

Batch programs must be able to restart from a checkpoint. (See the *DL/I DOS/VS Recovery/Restart Guide*, for information about writing restartable programs.)

Implementing recoverability of DL/I VSE databases

DL/I VSE writes both before- and after-images of changed segments to the CICS system log, thus providing records to support either forward or backward recovery. For this reason, good operational control of the system log files is vital. Loss or destruction of a system log file could jeopardize database integrity.

The logging for DL/I VSE is handled by CICS, and not directed to a separate DL/I VSE log.

To achieve this, the last 2 bits (bits 6 and 7) of the UPSI byte in the JCL must be set to 0. When the UPSI byte is not supplied, or is not needed for other reasons, bits 6 and 7 default to 0.

Backward recovery of DL/I VSE databases

Transaction backout (by DTB during a task abend, or by emergency restart after a system failure) causes backout of DL/I VSE database changes.

If a failure occurs while backing out DL/I VSE database changes, the XRCDBER exit of DFHDLBP, or the XDBDERR exit of DFHDBP, should be used to cancel CICS so that data integrity is maintained. (See Chapter 11, “User exits for transaction backout during emergency restart” on page 91 and “Global user exits in DFHDBP” on page 88.)

Forward recovery of DL/I databases

You may use forward recovery to recover lost or damaged files. DL/I VSE provides utilities for forward recovery of DL/I VSE databases. During implementation, it is necessary to establish procedures for the integration of the system log files produced during execution of CICS with those produced by execution of a non-MPS batch job. These procedures are needed to ensure a coherent and complete set of forward recovery information.

If a CICS abnormal termination has occurred, you must perform emergency restart after completion of forward recovery. Emergency restart then backs out the effects of tasks that were in flight at the time of failure.

Notes:

1. When the CICS system log is implemented on disk, ensure that each system log file is copied to tape before it is overwritten; otherwise, forward recovery information collected on the system log will be lost. Consider the use of the PAUSE option to prevent loss of information.
2. Similarly, when the CICS system log is implemented on tape, ensure that tapes are not reused until their forward recovery information is no longer needed.

Program isolation or intent scheduling

You specify program isolation or intent scheduling by the PI=YES|NO operand in the DL/I VSE application control table (ACT).

DL/I VSE error processing

When using CICS with DL/I VSE, error conditions can arise which may indicate that the integrity of the databases is at risk.

DL/I VSE pseudoabends causing transaction failure

Error conditions within DL/I VSE may be of a type that can be transformed into a transaction abend. Errors of this type do not damage the databases and do not prevent the continued execution of CICS. Examples of this type of error are program isolation deadlock, or no space in the database for an insertion. See “Transaction abend processing” on page 45.

DL/I VSE abends causing CICS failure

In situations where the DL/I VSE-detected error is sufficiently serious, the CICS system is abended to allow diagnosis of the error and database recovery actions to be taken.

This type of error causes a user abend of CICS. It is possible to detect and attempt recovery of this type of error using the system recovery table and system recovery program (although this is not recommended). See “Processing of operating system abends and program checks” on page 51.

DL/I VSE backout failure during DTB or emergency restart

If a failure occurs during the backout of DL/I VSE databases, a message is sent to the operator to show that an error has occurred.

- If this happens during an emergency restart, the default action is to give the system console operator the option to cancel CICS, or to allow emergency restart to continue. (For more information, see Figure 7 on page 143.)

  There is also a user exit, XRCDBER, which gives you the choice of ignoring the error, or of asking the operator whether CICS should continue or be canceled. You may also cancel CICS from the user exit.

  Note: If you are concerned about data integrity, you are recommended to cancel CICS by user code in the exit.

  With CICS terminated, the DL/I VSE database utilities can be run.

- If the DL/I VSE backout error occurs during dynamic transaction backout (DTB), the message does not give the operator the option to cancel CICS.

  Therefore, for DL/I VSE data integrity, CICS should be canceled by user code in the XDBDERR exit of DFHDBP; see “Global user exits in DFHDBP” on page 88.

Figure 7 on page 143 shows that if DL/I VSE data integrity is to be maintained after DL/I VSE backout failure, CICS must be canceled by the operator.

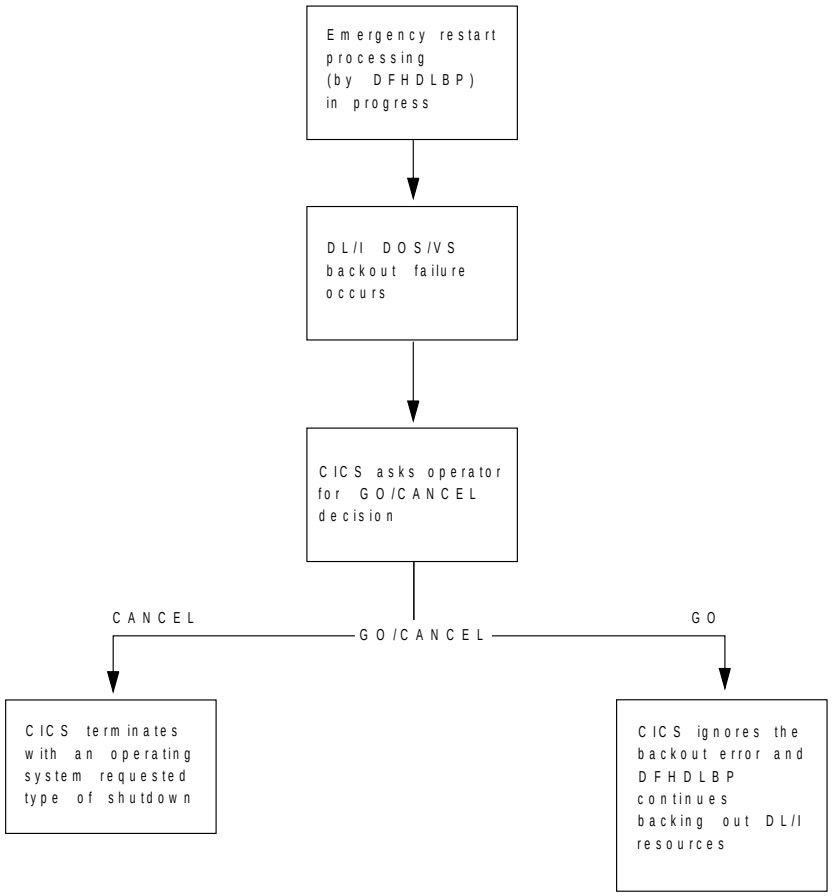

Figure 7. CICS/VSE processing of a DL/I VSE backout failure during emergency restart

Bibliography

CICS Transaction Server for VSE/ESA Release 1 library

Evaluation and planning	
Release Guide	GC33-1645
Migration Guide	GC33-1646
Report Controller Planning Guide	GC33-1941
General	
Master Index	SC33-1648
Trace Entries	SC34-5556
User's Handbook	SC34-5555
Glossary (softcopy only)	GC33-1649
Administration	
System Definition Guide	SC33-1651
Customization Guide	SC33-1652
Resource Definition Guide	SC33-1653
Operations and Utilities Guide	SC33-1654
CICS-Supplied Transactions	SC33-1655
Programming	
Application Programming Guide	SC33-1657
Application Programming Reference	SC33-1658
Sample Applications Guide	SC33-1713
Application Migration Aid Guide	SC33-1943
System Programming Reference	SC33-1659
Distributed Transaction Programming Guide	SC33-1661
Front End Programming Interface User's Guide	SC33-1662
Diagnosis	
Problem Determination Guide	GC33-1663
Messages and Codes Vol 3 (softcopy only)	SC33-6799
Diagnosis Reference	LY33-6085
Data Areas	LY33-6086
Supplementary Data Areas	LY33-6087
Communication	
Intercommunication Guide	SC33-1665
CICS Family: Interproduct Communication	SC33-0824
CICS Family: Communicating from CICS on System/390	SC33-1697
Special topics	
Recovery and Restart Guide	SC33-1666
Performance Guide	SC33-1667
Shared Data Tables Guide	SC33-1668
Security Guide	SC33-1942
External CICS Interface	SC33-1669
XRF Guide	SC33-1671
Report Controller User's Guide	GC33-1940
CICS Clients	
CICS Clients: Administration	SC33-1792
CICS Universal Clients Version 3 for OS/2: Administration	SC34-5450
CICS Universal Clients Version 3 for Windows: Administration	SC34-5449
CICS Universal Clients Version 3 for AIX: Administration	SC34-5348
CICS Universal Clients Version 3 for Solaris: Administration	SC34-5451
CICS Family: OO programming in C++ for CICS Clients	SC33-1923
CICS Family: OO programming in BASIC for CICS Clients	SC33-1671
CICS Family: Client/Server Programming	SC33-1435
CICS Transaction Gateway Version 3: Administration	SC34-5448

Books from VSE/ESA 2.4 base program libraries

VSE/ESA Version 2 Release 4

Book title	Order number
Administration	SC33-6705
Diagnosis Tools	SC33-6614
Extended Addressability	SC33-6621
Guide for Solving Problems	SC33-6710
Guide to System Functions	SC33-6711
Installation	SC33-6704
Licensed Program Specification	GC33-6700
Messages and Codes Volume 1	SC33-6796
Messages and Codes Volume 2	SC33-6798
Messages and Codes Volume 3	SC33-6799
Networking Support	SC33-6708
Operation	SC33-6706
Planning	SC33-6703
Programming and Workstation Guide	SC33-6709
System Control Statements	SC33-6713
System Macro Reference	SC33-6716
System Macro User's Guide	SC33-6715
System Upgrade and Service	SC33-6702
System Utilities	SC33-6717
TCP/IP User's Guide	SC33-6601
Turbo Dispatcher Guide and Reference	SC33-6797
Unattended Node Support	SC33-6712

High-Level Assembler Language (HLASM)

Book title	Order number
General Information	GC26-8261
Installation and Customization Guide	SC26-8263
Language Reference	SC26-8265
Programmer's Guide	SC26-8264

Language Environment for VSE/ESA (LE/VSE)

Book title	Order number
C Run-Time Library Reference	SC33-6689
C Run-Time Programming Guide	SC33-6688
Concepts Guide	GC33-6680
Debug Tool for VSE/ESA Fact Sheet	GC26-8925
Debug Tool for VSE/ESA Installation and Customization Guide	SC26-8798
Debug Tool for VSE/ESA User's Guide and Reference	SC26-8797
Debugging Guide and Run-Time Messages	SC33-6681
Diagnosis Guide	SC26-8060
Fact Sheet	GC33-6679
Installation and Customization Guide	SC33-6682
LE/VSE Enhancements	SC33-6778
Licensed Program Specification	GC33-6683
Programming Guide	SC33-6684
Programming Reference	SC33-6685
Run-Time Migration Guide	SC33-6687
Writing Interlanguage Communication Applications	SC33-6686

VSE/ICCF

Book title	Order number
Adminstration and Operations	SC33-6738
User's Guide	SC33-6739

VSE/POWER

Book title	Order number
Administration and Operation	SC33-6733
Application Programming	SC33-6736
Networking Guide	SC33-6735
Remote Job Entry User's Guide	SC33-6734

VSE/VSAM

Book title	Order number
Commands	SC33-6731
User's Guide and Application Programming	SC33-6732

VTAM for VSE/ESA

Book title	Order number
Customization	LY43-0063
Diagnosis	LY43-0065
Data Areas	LY43-0104
Messages and Codes	SC31-6493
Migration Guide	GC31-8072
Network Implementation Guide	SC31-6494
Operation	SC31-6495
Overview	GC31-8114
Programming	SC31-6496
Programming for LU6.2	SC31-6497
Release Guide	GC31-8090
Resource Definition Reference	SC31-6498

Books from VSE/ESA 2.4 optional program libraries

C for VSE/ESA (C/VSE)

Book title	Order number
C Run-Time Library Reference	SC33-6689
C Run-Time Programming Guide	SC33-6688
Diagnosis Guide	GC09-2426
Installation and Customization Guide	GC09-2422
Language Reference	SC09-2425
Licensed Program Specification	GC09-2421
Migration Guide	SC09-2423
User’s Guide	SC09-2424

COBOL for VSE/ESA (COBOL/VSE)

Book title	Order number
Debug Tool for VSE/ESA Fact Sheet	GC26-8925
Debug Tool for VSE/ESA Installation and Customization Guide	SC26-8798
Debug Tool for VSE/ESA User’s Guide and Reference	SC26-8797
Diagnosis Guide	SC26-8528
General Information	GC26-8068
Installation and Customization Guide	SC26-8071
Language Reference	SC26-8073
Licensed Program Specifications	GC26-8069
Migration Guide	GC26-8070
Migrating VSE Applications To Advanced COBOL	GC26-8349
Programming Guide	SC26-8072

DB2 Server for VSE

Book title	Order number
Application Programming	SC09-2393
Database Administration	GC09-2389
Installation	GC09-2391
Interactive SQL Guide and Reference	SC09-2410
Operation	SC09-2401
Overview	GC08-2386
System Administration	GC09-2406

DL/I VSE

Book title	Order number
Application and Database Design	SH24-5022
Application Programming: CALL and RQDLI Interface	SH12-5411
Application Programming: High-Level Programming Interface	SH24-5009
Database Administration	SH24-5011
Diagnostic Guide	SH24-5002
General Information	GH20-1246
Guide for New Users	SH24-5001
Interactive Resource Definition and Utilities	SH24-5029
Library Guide and Master Index	GH24-5008
Licensed Program Specifications	GH24-5031
Low-level Code and Continuity Check Feature	SH20-9046
Library Guide and Master Index	GH24-5008
Messages and Codes	SH12-5414
Recovery and Restart Guide	SH24-5030
Reference Summary: CALL Program Interface	SX24-5103
Reference Summary: System Programming	SX24-5104
Reference Summary: HLPI Interface	SX24-5120
Release Guide	SC33-6211

PL/I for VSE/ESA (PL/I VSE)

Book title	Order number
Compile Time Messages and Codes	SC26-8059
Debug Tool For VSE/ESA User's Guide and Reference	SC26-8797
Diagnosis Guide	SC26-8058
Installation and Customization Guide	SC26-8057
Language Reference	SC26-8054
Licensed Program Specifications	GC26-8055
Migration Guide	SC26-8056
Programming Guide	SC26-8053
Reference Summary	SX26-3836

Screen Definition Facility II (SDF II)

Book title	Order number
VSE Administrator's Guide	SH12-6311
VSE General Introduction	SH12-6315
VSE Primer for CICS/BMS Programs	SH12-6313
VSE Run-Time Services	SH12-6312

Notices

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

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

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

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

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply in 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 publication 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.

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 United Kingdom Laboratories, MP151, Hursley Park, Winchester, Hampshire, England, SO21 2JN. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Trademarks and service marks

The following terms, used in this publication, are trademarks or service marks of IBM Corporation in the United States or other countries:

CICS, CICS/VSE, DB2 for VSE/ESA, &DL1,
IBM, VSE/ESA, VSE/VSAM, VSE/POWER

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Sending your comments to IBM

CICS® Transaction Server for VSE/ESA™

Recovery and Restart Guide

SC33-1666-00

If you want to send to IBM any comments you have about this book, please use one of the methods listed below. Feel free to comment on anything you regard as a specific error or omission in the subject matter, and on the clarity, organization or completeness of the book itself.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail:

  IBM UK Laboratories
  Information Development
  Mail Point 095
  Hursley Park
  Winchester, SO21 2JN
  England

- By fax:
  - From outside the U.K., after your international access code use 44 1962 870229
  - From within the U.K., use 01962 870229
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: HURSLEY(IDRCF)
  - Email: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

- The publication number and title
- The page number or topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Program Number: 5648-054

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-1666-00

Spine information:

CICS TS for VSE/ESA Recovery and Restart Guide *Release 1*