CICS/ESA

Recovery and Restart Guide

Version 4 Release 1

SC33-1182-01

CICS/ESA

Recovery and Restart Guide

Version 4 Release 1

SC33-1182-01

Note!

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

First edition (October 1994)

This edition applies to Version 4 Release 1 of the IBM licensed program Customer Information Control System/Enterprise Systems Architecture (CICS/ESA), program number 5655-018, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Consult the latest edition of the applicable IBM system bibliography for current information on this product.

This book is based on the Recovery and Restart Guide for CICS/ESA 3.3, SC33-0658-02. Changes from that edition are marked by vertical lines to the left of the changes.

The CICS/ESA 3.3 edition remains applicable and current for users of CICS/ESA 3.3.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

At the back of this publication is a page entitled "Sending your comments to IBM". If you want to make comments, but the methods described are not available to you, please address them to:

IBM United Kingdom Laboratories Limited, Information Development,
Mail Point 095, Hursley Park, Winchester, Hampshire, England, SO21 2JN.

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

Contents

Notices

The following paragraph does not apply in any 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 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.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. 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 of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, U.S.A.

Programming interface information

This book is intended to help you design and implement a recovery and restart plan for your CICS system.

This book also documents Product-sensitive Programming Interface and Associated Guidance Information and Diagnosis, Modification or Tuning Information provided by CICS.

Product-sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of CICS. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive Programming Interface and Associated Guidance information is identified where it occurs, by an introductory statement to a chapter or section.

Diagnosis, Modification or Tuning Information is provided to help you diagnose problems with your CICS system.

Warning: Do not use this Diagnosis, Modification or Tuning Information as a programming interface.

Diagnosis, Modification or Tuning Information is identified where it occurs, by an introductory statement to a chapter or section.

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:

ACF/VTAM, CICS/ESA, DB2, IBM, IMS/ESA,
MVS/DFP, MVS/ESA, SAA, System/370, VTAM

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.

If you need to know where programming interface information is described, or about the definitions of the different types of information in the CICS library, you should read the *CICS Family: Library Guide*.

The information in this book is generally restricted to a single CICS system. For guidance on ISC and MRO, see the *CICS/ESA Intercommunication Guide*. For information about XRF systems, see the *CICS/ESA 3.3 XRF Guide*. However, the XRF takeover is based on emergency restart processing, so the information in this book is relevant to XRF.

This book discusses CICS with local DL/I as a data-sharing subsystem. It also contains a brief DB2 section, and a select DB2 bibliography. For information on the CICS-IMS interface using Database Control (DBCTL), see the *CICS/ESA CICS-IMS Database Control Guide*.

This book does not describe recovery and restart for the CICS/ESA front end programming interface. For information on this topic, see the *CICS/ESA 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 topic.

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 of writing 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's unlikely that any one reader would have to implement all the possible techniques discussed in this book. By using the table of contents, you should be able to find the sections relevant to your work. Readers new to recovery and restart should find Part 1 helpful, because it introduces the concepts of recovery and restart.

Notes on terminology

In this book, the term **CICS** refers to CICS/ESA, the term **VTAM** refers to ACF/VTAM, the term **TCAM** refers to the DCB Interface of ACF/TCAM, and the term **IMS** refers to IMS/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 17
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 57
Describes how to implement the functions of recovery and restart. Each chapter deals in detail with a particular topic, referring back to information about design or processes when necessary.

Part 4, "Recovery in a DL/I environment" on page 193
Describes local DL/I recovery processing, and the DB2 and IMS data base recovery control (DBRC) and internal resource lock manager (IRLM) facilities that you can use with CICS. It describes how to implement DL/I recovery with and without data sharing.

Determining if a publication is current

IBM regularly updates its publications with new and changed information. When first published, both hardcopy and BookManager softcopy versions of a publication are in step, but subsequent updates will probably be available in softcopy before they are available in hardcopy.

For CICS books, these softcopy updates appear regularly on the *Transaction Processing and Data Collection Kit* CD-ROM, SK2T-0730-xx. Each reissue of the collection kit is indicated by an updated order number suffix (the -xx part). For example, collection kit SK2T-0730-06 is more up-to-date than SK2T-0730-05. The collection kit is also clearly dated on the cover.

Here's how to determine if you are looking at the most current copy of a publication:

- A publication with a higher suffix number is more recent than one with a lower suffix number. For example, the publication with order number SC33-0667-02 is more recent than the publication with order number SC33-0667-01. (Note that suffix numbers are updated as a product moves from release to release, as well as for hardcopy updates within a given release.)

- When the softcopy version of a publication is updated for a new collection kit the order number it shares with the hardcopy version does not change. Also, the date in the edition notice remains that of the original publication. To compare softcopy with hardcopy, and softcopy with softcopy (on two editions of the collection kit, for example), check the last two characters of the publication’s filename. The higher the number, the more recent the publication. For example, DFHPF104 is more recent than DFHPF103. Next to the publication titles in the CD-ROM booklet and the readme files, asterisks indicate publications that are new or changed.
- Updates to the softcopy are clearly marked by revision codes (usually a “#” character) to the left of the changes.

Bibliography

CICS/ESA 4.1 library

Evaluation and planning		
Release Guide	GC33-1161	April 1997
Migration Guide	GC33-1162	April 1997
General		
CICS Family: Library Guide	GC33-1226	April 1995
Master Index	SC33-1187	October 1994
User's Handbook	SX33-1188	April 1997
Glossary (softcopy only)	GC33-1189	n/a
Administration		
Installation Guide	GC33-1163	April 1997
System Definition Guide	SC33-1164	April 1997
Customization Guide	SC33-1165	April 1997
Resource Definition Guide	SC33-1166	April 1997
Operations and Utilities Guide	SC33-1167	April 1997
CICS-Supplied Transactions	SC33-1168	April 1997
Programming		
Application Programming Guide	SC33-1169	October 1994
Application Programming Reference	SC33-1170	April 1997
System Programming Reference	SC33-1171	April 1997
Sample Applications Guide	SC33-1173	October 1994
Distributed Transaction Programming Guide	SC33-1174	October 1994
Front End Programming Interface User's Guide	SC33-1175	October 1994
Diagnosis		
Problem Determination Guide	SC33-1176	October 1994
Messages and Codes	GC33-1177	April 1997
Diagnosis Handbook	LX33-6093	October 1994
Diagnosis Reference	LY33-6082	April 1997
Data Areas	LY33-6083	April 1997
Supplementary Data Areas	LY33-6081	October 1994
Closely-Connected Program Interface	LY33-6084	November 1996
Communication		
Intercommunication Guide	SC33-1181	April 1997
Server Support for CICS Clients	SC33-1591	February 1996
CICS Family: Inter-product Communication	SC33-0824	October 1996
CICS Family: Communicating from CICS on System/390	SC33-1697	October 1996
Special topics		
Recovery and Restart Guide	SC33-1182	October 1994
Performance Guide	SC33-1183	October 1994
CICS-IMS Database Control Guide	SC33-1184	October 1994
CICS-RACF Security Guide	SC33-1185	October 1994
Shared Data Tables Guide	SC33-1186	October 1994
External CICS Interface	SC33-1390	April 1997
CICS ONC RPC Feature for MVS/ESA Guide	SC33-1119	February 1996
CICS Web Interface Guide	SC33-1892	November 1996

The book that you are reading was published in hardcopy format on the date shown in the right-hand column in the above table. Some of the CICS/ESA 4.1 books were republished in new hardcopy editions in April 1997 to incorporate updated information previously available only in softcopy. Note that it is possible that other books in the library will be updated after April 1997.

When a new order is placed for the CICS/ESA 4.1 product, the books shipped with that order will be the latest hardcopy editions.

The style of IBM covers changes periodically. Books in this library have more than one style of cover.

For information about the softcopy books, see “Determining if a publication is current” on page x. The softcopy books are regularly updated to include the latest information.

Other CICS books

- *CICS Application Migration Aid Guide*, SC33-0768
- *CICS Application Programming Primer (VS COBOL II)*, SC33-0674
- *CICS/ESA Facilities and Planning Guide* for CICS/ESA Version 3 Release 3, SC33-0654
- *CICS/ESA XRF Guide* for CICS/ESA Version 3 Release 3, SC33-0661
- *CICS Family: API Structure*, SC33-1007
- *CICS Family: General Information*, GC33-0155
- *IBM CICS Transaction Affinities Utility MVS/ESA*, SC33-1159

CICS Clients

- *CICS Clients: Administration*, SC33-1436
- *CICS Family: Client/Server Programming*, SC33-1435

Books from related libraries

Hardware:

IBM System/370 and 4300 Processors Bibliography, GC20-0001

Operating system:

OS PL/I Optimizing Compiler: Programmer’s Guide , SC33-0006
OS/VS TCAM System Programmer’s Guide, GC30-2051
OS/VS TCAM Application Programmer’s Guide, GC30-3036
OS/VS TCAM Concepts and Applications, GC30-2049

IMS/ESA Version 3.1 and 4.1:

IMS/ESA Utilities Reference, SC26-4284 (IMS/ESA 3.1 only)
IMS/ESA Utilities Reference: Data Communication, SC26-4628 (IMS/ESA 4.1 only)
IMS/ESA Utilities Reference: Systems, SC26-4629 (IMS/ESA 4.1 only)
IMS/ESA Operations Guide, SC26-4287
IMS/ESA Messages and Codes, SC26-4290
IMS/ESA Operator’s Reference Manual, SC26-4288
IMS/ESA Sample Operating Procedures, SC26-4324
IMS/ESA Installation Guide, SC26-4276
IMS/ESA Application Programming: Design Guide, SC26-4279
IMS/ESA Application Programming: DL/I Calls, SC26-4274

IMS/ESA Application Programming: EXEC DLI Commands, SC26-4280
IMS/ESA Diagnosis Guide and Reference, LY27-9539

Database 2:

IBM Database 2 General Information, GC26-4073
IBM Database 2 Reference, SC26-4078
IBM Database 2 Operation and Recovery Guide, SC26-4083
IBM Database 2 Operations and Recovery Sample Procedures, GG24-3180
IBM Database 2 Application Programming for CICS/OS/VS Users, SC26-4080
IBM Database 2 Utilities Guide, GG24-3130

CICS VSAM Recovery (CICSVR) MVS/ESA:

CICS VSAM Recovery MVS/ESA Version 2 General Information, GH19-6708
CICS VSAM Recovery MVS/ESA Version 2 User’s Guide and Reference, SH19-6970
CICS VSAM Recovery MVS/ESA Version 2 Implementation Guide, SH19-6971
CICS VSAM Recovery MVS/ESA Version 2 Messages and Problem Determination, SH19-4006

Storage management subsystem:

MVS/DFP General Information, GC26-4553
MVS/ESA Storage Management Library: Storage Management Subsystem Migration and Planning Guide, SC26-4659
MVS/ESA Catalog Administrator Guide, SC26-4502
MVS/DFP Storage Administration Reference, SC26-4566
Data Facility Hierarchical Storage Manager General Information, GH35-0092
Data Facility Data Set Services General Information, GC26-4123.

Systems Application Architecture CPI Resource Recovery Interface:

CPI Resource Recovery Reference, SC31-6821.

Summary of changes

Changes for this CICS/ESA 4.1 edition

New and significantly changed information for this CICS/ESA 4.1 edition includes:

- A description of VTAM persistent sessions and their purpose
- Revised discussions of autoinstall to distinguish between terminals and programs
- A description of backup while open (BWO)
- Restoring the count of the number of shipped definitions
- Removal of IMS/VS 2.2 as a supported release for local DL/I
- A description of CICSVR 2.2.

Changes are indicated by vertical lines to the left of the changes.

Changes for the CICS/ESA 3.3 edition

New and changed information for the CICS/ESA 3.3 edition included:

- Minor improvements and corrections to the text. No new function was described.

Changes for the CICS/ESA 3.2.1 edition

New and changed information for the CICS/ESA 3.2.1 edition included:

- A description of the backup-while-open (BWO) facility has been added. BWO provides forward recovery protection for files that need to be open for update for extensive periods of time.
- The group-commit function of CEDA installs now applies only to VTAM TCT resources.
- Information about recovery of main storage data has been removed, because DFHUAKP is no longer supplied.
- Changes to the way CICS handles automatic initiate descriptors (AIDs) are described.
- Information has been added about the MRO/ISC in-doubt window.
- The chapter on forward recovery has been changed to describe the MVS/ESA version of the CICS VSAM Recovery utility.
- The use of the CONSNAME attribute of CEDA DEFINE TERMINAL for defining console terminals is discussed.

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
- Recovery requirements
- Logical units of work and synchronization points.

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.

Backing up your data while it is in use

In some environments, a VSAM file managed by CICS file control might need to remain online and open for update for extended periods. You can use a backup manager, such as the Data Facility Storage Management Subsystem/MVS (DFSMS/MVS), in a separate job under MVS/ESA, to back up this VSAM file at regular intervals while it is open for update by CICS applications. This operation is known as **backup while open (BWO)**. Even changes made to the VSAM file while the backup is in progress will be recorded.

Then, if a data set failure occurs, you can use a forward recovery utility, such as CICS VSAM Recovery (CICSVR) MVS/ESA, to recover the VSAM file.

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/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, such as CICSVR, 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.

Chapter 7, “Starting to specify recovery and restart facilities” on page 59 gives an outline of how to define a system with basic recovery and restart facilities.

CICS logs the information required for the forward recovery of DL/I databases (after-images). If your system uses DBRC (Database Recovery Control component of IMS), on shared or unshared DL/I databases, read Part 4, “Recovery in a DL/I environment” on page 193.

VTAM persistent sessions considerations

Persistent session support improves the availability of CICS. It benefits from VTAM 3.4.1 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.

You can change the interval using the CEMT SET VTAM command, or the 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 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 by the system programmer.

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 or XRF are in use:

- If CICS is running without VTAM persistent sessions or XRF, and fails, the user sees the VTAM logon panel followed by the “good morning” message (if AUTOCONNECT(YES) is specified for the 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 RECOVOPTION and RECOVNOTIFY keywords allows the system administrator 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 end user to take the appropriate recovery actions.

  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/ESA when the persistent recovery is complete. After a successful emergency restart, the options defined in PSRECOVERY of the CONNECTION definition and RECOVOPTION of the session definition take effect. If the appropriate recovery options have been selected (see the *CICS/ESA Resource Definition Guide*), and the APPC sessions are in the correct state, CICS/ESA will perform an ISSUE ABEND (see the *CICS/ESA Distributed Transaction Programming Guide*) to inform the partner that the current conversation has been abnormally terminated.

Multi-node persistent sessions

If you are running CICS with VTAM 4.4 or later and are using multi-node persistent sessions (MNPS) support, sessions can be retained across a VTAM failure. After VTAM has been restarted, sessions are restored when the ACB is reopened (either automatically by the COVR transaction or by a CEMT, or EXEC CICS, SET VTAM OPEN command). To ensure that CICS retains its sessions and restores them when the ACB is reopened, code PSTYPE=MNPS as a system initialization parameter.

When the VTAM failure occurs and the TPEND failure exit is driven, the autoinstalled terminals that are normally deleted at this point are retained by CICS. If the session is not restored and the terminal is not reused within the AIRDELAY interval, CICS deletes the TCTTE when the AIRDELAY interval expires after the ACB is re-opened successfully.

Table 1 indicates whether sessions are restored or unbound in a number of recovery scenarios: (1) a CICS emergency restart with persisting sessions; (2) a VTAM restart after a VTAM failure, where CICS continues running and the ACB is reopened; (3) CICS cold and warm starts; and (4) the dynamic open of the VTAM ACB by a CICS operator command.

Table 1. Effect of VTAM persistent sessions support on CICS terminals and sessions

Recovery scenario	**Terminal**	**LU62 (synclevel 1)**	**LU62 (synclevel 2)**	**LU61**
CICS emergency restart	Restored	Restored	Restored	Unbound
VTAM_RESTART	Restored	Restored	Restored	Unbound
CICS COLD start	Unbound	Unbound	Unbound	Unbound
CICS WARM start	Unbound	Unbound	Unbound	Unbound
Dynamic open of VTAM ACB	Unbound	Unbound	Unbound	Unbound

Single-node persistent sessions

If you are running with VTAM single-node persistent sessions (SNPS) support (that is, PSTYPE=SNPS is specified as a system initialization parameter or is allowed to default) sessions are not retained or recovered after a VTAM abend and subsequent opening of the VTAM ACB (CEMT SET VTAM OPEN).

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 you perform a COLD start 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 a terminal or session is defined with the recovery option (RECOVOPT) set to UNCONDREL or NONE
- If CICS determines that it cannot recover the session without unbinding and rebinding it
- If a connection 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, MVS, or processor (CPC) failure
- VTAM does not retain CICS sessions if you close VTAM with any of the following 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/ESA 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 a special type of CICS startup called **emergency restart**.

In readiness for backout during CICS emergency restart, CICS writes recovery information to a journal called 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
  - BDAM 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, signalled 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 program specification block (PSB) termination (TERM) call or command. This means that only one DL/I 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 program issues a DL/I checkpoint call. This can occur when the batch DL/I program is sharing a database with CICS applications through the CICS shared database facility.

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 on page 11, 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.

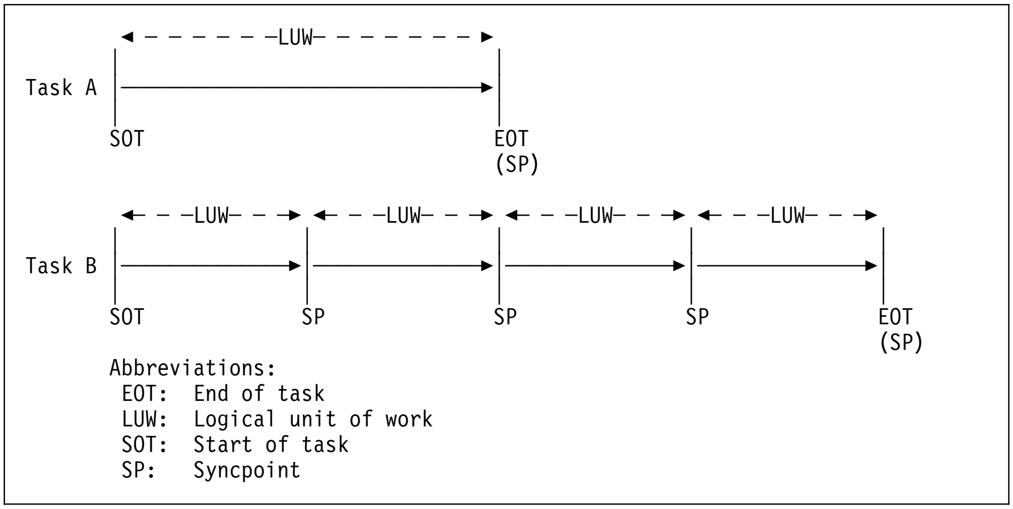


Figure 1. Logical units of work (LUWs) and syncpoints

Figure 2 on page 12 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.

If there is a system failure at time X:

- The change(s) 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:

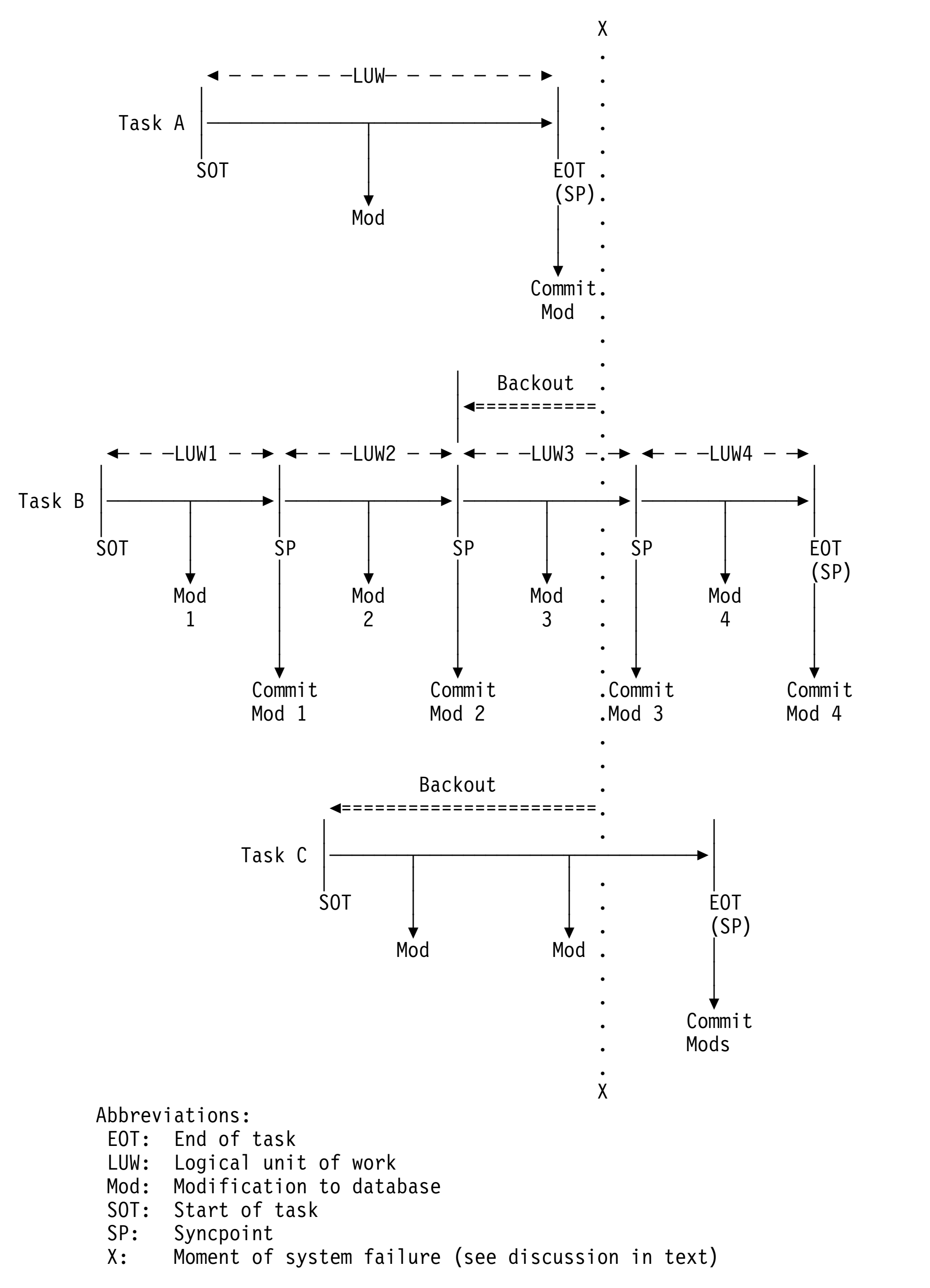
1. Take a backup copy of the data set at regular intervals.
2. Record an after-image of every change to the data set on the system log or any other journal.
3. After the failure, use the information recorded on the system log or other journal to bring the backup copy to the most up-to-date condition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

These operations are known as **forward recovery**.

Forward recovery of local DL/I databases

CICS writes after-images of local DL/I database changes to the system log. These records are available for forward recovery operations, possibly using IMS utility programs.

The use of IMS database recovery control (DBRC) makes forward recovery of DL/I databases easier. See Part 4, “Recovery in a DL/I environment” on page 193 for further information.



Abbreviations:
EOT: End of task
LUW: Logical unit of work
Mod: Modification to database
SOT: Start of task
SP: Syncpoint
X: Moment of system failure (see discussion in text)

Figure 2. Backout of logical units of work

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 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 81. See Chapter 2, "Recording of recovery information" on page 19 for more information about automatic archiving.

If the data set must remain open for update for long periods, you can use DFSMS, in a separate job, to take a backup copy of the VSAM data set at regular intervals (see Chapter 11, "Backup while open (BWO)" on page 97).

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 87). 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 15).

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.
- # Loss of an MVS image that is a member of a sysplex

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 more information on link failure, see the *CICS/ESA Intercommunication Guide*. For information about a link failure involving XCF/MRO, see “XCF/MRO partner failures” on page 14.

When communication fails, the communication system access method either retries the transmission or notifies CICS. 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 15).
- Reducing the amount of error information printed.

“Your own NEP processors” on page 159, and “Your own TEP code” on page 160, have more information about the sample NEPs and TEPs. For programming information about coding your own NEPs and TEPs, see the *CICS/ESA Customization Guide*. More general information is in Chapter 6, “Communication error processing” on page 55.

XCF/MRO partner failures

Loss of communication between CICS regions can be caused by the loss of an MVS image in which CICS regions are running. If the regions are communicating over XCF/MRO links, the loss of connectivity may not be immediately apparent because XCF waits for a reply to a message it issues.

The loss of an MVS image in a sysplex is detected by XCF in another MVS, and XCF issues message IXC402D. If the failed MVS is running CICS regions connected through XCF/MRO to CICS regions in another MVS, tasks running in the active regions are initially suspended in an IRLINK WAIT state.

XCF/MRO-connected regions do not detect the loss of an MVS image and its resident CICS regions until an operator replies to the XCF IXC402D message. When the operator replies to IXC402D, the CICS interregion communication program, DFHIRP, is notified and the suspended tasks are abended, and MRO connections closed. Until the reply is issued to IXC402D, an INQUIRE

# CONNECTION command continues to show connections to regions in the failed
# MVS as in service and normal.

# When the failed MVS image and its CICS regions are restarted, the interregion
# communication links are reopened automatically.

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 49.

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 such as CICSVR can then recover the data sets offline while CICS remains running.

Chapter 5, “Abend processing” on page 47 gives more details about CICS processing of 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 CEDA.

More details of CICS processing following a system failure are in “Emergency restart” on page 37. 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 information, see Part 4, “Recovery in a DL/I environment” on page 193.

Chapter 2. Recording of recovery information

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

- “Global catalog”
- “Restart data set” on page 21
- “Dynamic log (for dynamic transaction backout)” on page 21
- “System log (journal 1)” on page 22
- “Journals 2 through 99” on page 27
- “Journal archive control data set” on page 28

When local DL/I runs under IMS with CICS, all logging for DL/I recovery is directed to the CICS dynamic and system logs. Do not use the batch log that is normally created in DL/I batch processing when running DL/I under CICS. However, if you are using DBRC with local DL/I, see Chapter 21, “Using IMS DBRC for recovery control” on page 203 and Chapter 22, “Recovery and restart in an IMS data-sharing environment” on page 209.

Recording on the catalogs

CICS uses two catalogs:

- The global catalog
- The local catalog.

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

You should consider taking backups of the catalog(s) 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. Also, while CICS is running, you should not compress a library that contains any program load modules that are used by the CICS system, because this could invalidate information on the catalogs.

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 30).
- Tape-volume identifier and descriptor lists (for standard-labeled tapes only). For journals (including the system log) on standard-labeled tapes, CICS records on the global catalog the identifier of the tape volume opened for output. The identifier of the previous tape volume opened for output is overwritten.

  This information is held across all types of start, and is used by emergency restart to determine the volume identifier of the system log volume to be used. (For more information about standard-labeled tapes, see page 69.)
- 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 28.
- Resource information. The following information is recorded on the global catalog during CICS execution (see “Recovering dynamically added resource definitions” on page 42), 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 global catalog)
  - DL/I status information
  - DL/I error information, including EEQE 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
  - APPC connection information so that relevant 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 DUMPDS=AUTO 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 38).

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.
- 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 CEDA DEFINE TRANSACTION command.

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 50).

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 67 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 70).

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 152).

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 37.

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.

The following additional actions are taken for files that use backup while open (BWO):

- Tie-up records (TURs) are recorded on the forward recovery journal. A TUR associates a CICS file name with a VSAM data set name.
- Recovery points are recorded in the integrated catalog facility (ICF) catalog. These define the starting time for forward recovery. Data recorded on the forward recovery log before that time will not be used.

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 about this parameter, see the *CICS/ESA System Definition Guide*.
- Every time logging starts on a new disk data set or tape volume (unless an activity keypoint is already being written).
- By opening a labeled journal tape.

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.

DFHJ01X data set

During emergency restart only, a third data set (DFHJ01X) can be used as the system log to keep backout changes during emergency restart separate from the changes that were logged for the previous run. DFHJ01X must be defined if local DL/I is used in the CICS system; otherwise it is optional.

If the system log is used for forward recovery, CICS also logs forward recovery information to DFHJ01X during emergency restart. Therefore, this data set becomes part of the input stream for the forward recovery utility.

The presence of a DD statement for DFHJ01X in the startup JCL causes CICS to use that data set during emergency restart. If DFHJ01X is not defined, CICS uses either DFHJ01A or DFHJ01B, whichever was used during the previous run.

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 68.

An alternative is to use the DFHJCT JOUROPT PAUSE and LRU|EXTA options, which request 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 local DL/I is in use, whether automatic archiving is specified, and whether the system log data sets have been formatted for this CICS run.

Table 2 on page 25 illustrates where logging starts on two-disk system log data sets that have not been formatted for this CICS run.

Table 3 on page 26 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/ESA System Definition Guide* tells you more about DFHJCJFP and formatting journals. You should not format the system log before an emergency restart because you would 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
DL/I with DFHJ01X data set	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.
	emergency	At start of DFHJ01X, and then whichever data set is READY, or DFHJ01A, as above.	At start of DFHJ01X, then continuing at start of DFHJ01A, or at start of less recently used of DFHJ01A/B depending on JOUROPT=LRU|EXTA option.
No DL/I, but with the DFHJ01X data set	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 1.)
	emergency	At start of DFHJ01X, and then whichever data set is READY, or DFHJ01A, as above.	At start of DFHJ01X, then continuing at start of DFHJ01A, or at start of less recently used of DFHJ01A/B depending on JOUROPT=LRU|EXTA option.
No DL/I, and no DFHJ01X data set	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. If the last data set used is near the end of volume, or 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. If the global catalog indicates DFHJ01X as the current data set, journaling is repositioned to the start of DFHJ01A. 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.

2. If the last data set used is near the end of volume, or the data set chosen conflicts with 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 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 closed 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 67.

6. You cannot specify warm or emergency starts. They depend on the START=AUTO system initialization parameter.

7. Records written to DFHJ01X represent backed-out changes used for DL/I or VSAM forward recovery, so DFHJ01X must be copied for forward recovery purposes before the next emergency restart.

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**
DL/I	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.
	emergency	Not possible.	Not possible.
No DL/I	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.)
	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 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. DD names are associated with the tape drive(s) as follows:

- For one tape drive, the DD name is DFHJ01A. When one tape volume has been filled, another tape volume is mounted and recording continues using the **same** DD name.
- For two tape drives, the CICS DD 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 DD names, alternately.

Unlabeled or standard-labeled tapes

CICS supports the use of unlabeled or standard-labeled tapes, specified by the LABEL operand in the journal control table (JCT). For more information, see "Implementing the system log on tape" on page 68.

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 such as CICSVR:

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

  You can create user journal records by executing 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. This is requested by options on CEDA DEFINE TRANSACTION commands. These messages could be used to create audit trails. Remember that syncpoint records are written only to the system log.

  Automatic journaling is requested by options in the CEDA DEFINE FILE command or by DFHFCT operands. Automatic journaling is used for user-defined purposes; for example, for an audit trail, or for a forward recovery program. Automatic journaling is not used for CICS recovery purposes.

- CICS records after-images of updates made to CICS files for use with a forward recovery utility. Also, for files that use backup while open (BWO), CICS records tie-up records (TURs). A TUR associates a CICS file name with a VSAM data set name.

  You specify which journal is to receive this data by the FWDRECOVLOG option of the CEDA DEFINE FILE command. Any journal, including the system log, may be used for this purpose.

Like the system log, user journals may be defined for one or two disks or tape. Table 4 on page 28 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. If the last data set used is near the end of volume, or 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 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 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, 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 will be as for a cold or warm start.

Chapter 8, “Logging and journaling” on page 67 provides information about implementing journals.

Journal archive control data set

For journals defined with automatic journal archiving (JOUROPT=AUTOARCH), details of their status are kept on the journal archive control data set. This is a VSAM relative record data set (RRDS).

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.

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 will complete.
- Tasks that are automatically initiated will be 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) will 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.

  Files that are eligible for BWO support have the BWO attributes in the ICF catalog set to indicate that BWO is not supported. This prevents BWO backups being taken in the subsequent batch window.
- CICS writes statistics to the system management facility (SMF).
- CICS writes the following information to the global catalog:
  - A warm keypoint (see “Warm keypoints”).
  - The tape volume descriptor lists for all journals (if any) implemented on standard-labeled tapes—including the system log.
  - 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 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 35 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 global catalog.
- CICS does not close files defined to CICS file control.
- Sessions will 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 global catalog whenever CICS is initialized. This indicator remains until the next startup, provided you do not reinitialize the 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 53.)

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 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 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 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, because of the emergency-restart-needed indicator written to the 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.
  - # A cold start if CICS is running with JCT=NO specified as a system
    # initialization parameter.

  When START=AUTO is specified, CICS inspects the control record on the 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 that indicator (when the 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 (see note 4 on page 196).

The use of the catalogs, the system log, and user journals at restart, is described in Chapter 2, "Recording of recovery information" on page 19.

The CICS initialization process for cold and warm starts 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 (JACD). If, for some reason, you want to override the status information, you need to redefine the JACD. The *CICS/ESA 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. If you wish to ignore this status information at startup, use the JSTATUS=RESET system initialization parameter.
- For journals defined to use standard-labelled tape volumes, CICS recovers the status from the CICS global catalog. To ignore this status information at startup, use the SERIES=PURGE system initialization parameter.

Cold start

In a cold start, CICS initializes with limited reference to any system activity recorded in the CICS catalogs. A fully 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 (CSD) 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, you should **not** attempt to do an emergency restart. This is because the information needed for emergency restart may not have been written to the global catalog. When the cause of the failure has been corrected, you should 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 91).

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 33).
- 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 global catalog.
- Data on intrapartition transient-data and on auxiliary temporary storage is lost.
- Extended error queue element (EEQE) information for local DL/I databases is held on the global catalog, and is used at a cold start. **If you need the EEQEs for data integrity, do not delete your catalog before the cold start.** The EEQE information in “DL/I warm start” on page 198 for a warm start, is also valid for a cold start.
- Dump table entries are lost.
- The value of the LPA 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 30).

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 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, you should be aware that there might be a conflict between the information in the warm keypoint and in the control table. This might cause problems later.

- Tape-volume descriptor lists are reconstructed from the tape-volume descriptors recorded in the global catalog during the most recent normal shutdown.
- CICS recovers the status of the system log and user journals (see "Recovery of system log and user journals" on page 33).

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 91).

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 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.

- Local DL/I. You may add databases at restart, but for existing databases you should be aware of the relationships between the elements (see "DL/I warm start" on page 198).
- 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.

If you code PGAICTLG=NONE, autoinstalled program definitions are not cataloged. If you code PGAICTLG=MODIFY 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. If you code 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.

  Notes:

  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 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.
- LPA=YES|NO status.

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 database, you do not normally need to run the DFSBBO00 batch backout utility before the emergency restart. But, if backout of the DL/I 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 *IMS/ESA Operations Guide* tells you when to do this, and the *IMS/ESA Utilities Reference: Database* manual tells you how 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 the CEDA INSTALL command, are preserved (see “Recovering dynamically added resource definitions” on page 42).

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

You should **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 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 22). To make emergency restart processing possible, you must set a nonzero value in the system initialization AKPFREQ 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 33).

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. The tape-volume descriptor lists are recreated. 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 91
- Chapter 13, "User exits for transaction backout during emergency restart" on page 151.

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 30). Any terminals installed after CICS wrote the last warm keypoint will not be recovered. If CICS cannot find a warm keypoint, it will install resources that were active at the last cold start. To make changes to these terminals, always use RDO.
- The global catalog. The 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 LPA status are reapplied from the local catalog.

  | The 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 40).
- 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 13, "User exits for transaction backout during emergency restart" on page 151).

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 (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 BDAM) 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 that was 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 backout exit program (XRCFCER) (see “XRCFCER exit” on page 154).

    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, though, 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.
- Local DL/I databases. If DL/I backout processing fails, all potentially affected databases are stopped to preserve data integrity, but emergency restart processing continues (see “DL/I backout failure” on page 198).
- 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 with OPENTIME(STARTUP), 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 CEDA DEFINE FILE command.

- 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 184. 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 153.

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 SET FILE commands or by 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 FILSTAT specification in the file definition, 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) which 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.

Recovering local DL/I database statuses

At emergency restart, CICS maintains the integrity of local DL/I databases if databases have had:

- Write I/O errors. For information about I/O errors, see Chapter 20, “Implementing local DL/I recovery and error processing” on page 195.
- Status changes because of master terminal operations. The status of the following parameters of the CEMT SET DLIDATABASE command is maintained from the previous run of CICS:
  - STARTED
  - STOPPED
  - RECOVERDB
  - DUMPDB
  - ACCESS value (RO, RD, UP, or EX).

Note: The information about extended error queue elements (EEQEs) under warm start information in “DL/I warm start” on page 198, also applies to emergency restart. But at emergency restart, you should not change the relationship between the items listed in “DL/I warm start” on page 198 for a database unless that database is forward-recovered before the emergency restart.

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 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 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 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 global catalog as soon as each resource is installed. If CICS does not succeed in installing the entire group, CICS does not back out the individual installed resources. They are, in effect, committed individually.

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

If CICS fails before commit processing has started for the group, CICS will, on a subsequent emergency restart, recover any resources (except VTAM TCT resources) from the 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 global catalog. Other resources in the group that were installed before CICS failed will be on the global catalog.

During the subsequent emergency restart, the resource manager creates its set of resource definitions from the 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 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 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 35 for information about when autoinstalled programs are cataloged in the global catalog.

Recovering console terminals: In an MVS/ESA SP 4.1 sysplex environment, you should assign a unique name to each console terminal, by means of the CONSNAME attribute of CEDA DEFINE TERMINAL. If you do not use the CONSNAME attribute, the console will be identified by a numeric identifier that MVS assigns in the order in which the MVS image joins the sysplex.

When the sysplex is restarted, these numeric identifiers are lost and MVS assigns new identifiers. So, if a CICS restart occurs across such a sysplex restart, console terminal entries are reinstalled, but might refer to different physical console devices.

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 transaction. The *CICS/ESA 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.[1] 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.[2]

If the previous session abended, you must not use COLDACQ because this overrides CICS integrity control. This could lead to data integrity problems. Also, you should check the CSMT log for an activity keypoint after the restart of a session following a CICS failure. If there is no activity keypoint, you should 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.

Table 5 (Page 1 of 2). Comparison of types of CICS restart

	Cold Start	**Warm Start**	**Emergency Restart**
Information from system log of previous run?	Not used	Not used	Used
Tape volume descriptor lists obtained from?	Global catalog (see Note 1 on page 45)	Global catalog	Last activity keypoint on system log. System log tape ID is obtained from the global catalog.
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

[1] For a discussion of message resynchronization with TCAM, see the *OS/VS TCAM System Programmer's Guide*, the *OS/VS TCAM Application Programmer's Guide*, and *OS/VS TCAM Concepts and Applications*.

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

Table 5 (Page 2 of 2). Comparison of types of CICS restart

	Cold Start	**Warm Start**	**Emergency Restart**
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 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 2.	Not used	Not used
RD information in the groups in the list named by the GRPLIST system initialization parameter for the PREVIOUS initialization	Not applicable	Obtained from global catalog	Obtained from global catalog
RD information in groups that have been INSTALLed since the last cold start	Not applicable	Obtained from global catalog	Obtained from global catalog (and system log for VTAM TCT resources)
Autoinstalled terminals	Not applicable	Global catalog if AID outstanding	Global catalog
Autoinstalled programs	Not applicable	Obtained from global catalog. See Note 3.	Obtained from global catalog. See Note 3.

Notes:

1. Tape-volume descriptor lists are not used in a cold start if SERIES=PURGE is specified as a system initialization parameter.
2. For more information about the CSD, see the *CICS/ESA Resource Definition Guide*.
3. In the case of autoinstalled programs, these may or may not have been recorded on the global catalog depending on the PGAICTLG option 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 91.

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 53.

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 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
- A pseudoabend in IMS.

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 49 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 HANDLE ABEND commands; for programming information on these, see the *CICS/ESA 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 program LINKs to another program, the LINKed-from program **and** the LINKed-to program can each have one active exit.
2. When an exit is activated (at a particular program level), any other exit that may already be active at the same level automatically becomes deactivated.

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 171.

When an abend request is issued for a task, CICS immediately passes control to the exit that is active at the current logical level[3]:

- 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 49).

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 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 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 a RETURN command, CICS returns control to the next higher logical level at the point following the LINK command (not to any exit code that may be active) just as if the 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 a 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.

[3] A LINKed-to program 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/ESA 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 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)
- Whether 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. The default program requests restart under certain conditions; for example, in the event of a program isolation deadlock (that is, when two tasks each wait for the other to release a particular DL/I database segment), one of the tasks is backed out and automatically restarted, and the other is allowed to complete its update.

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

For migration information about DFHREST, see the *CICS/ESA Migration 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 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/ESA 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 “User exits in DFHDBP” on page 94).

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 local DL/I 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 21). 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 BDAM), records to be deleted should be marked for deletion in an XDBFERR exit program (see “User exits in DFHDBP” on page 94). (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 system log
- Sets a backout-failed status in the 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 such as CICSVR 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 the batch backout utility, see Chapter 18, “Backout failure” on page 189. See Chapter 12, “Forward recovery and backout with CICSVR” on page 125 for information about CICSVR.

Local DL/I databases

If DL/I backout processing fails, all potentially affected databases are stopped to preserve data integrity, but CICS continues to run. See “DL/I backout failure” on page 198.

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 sync point)
- 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 will start at its specified time (and may already be executing when the START-issuing task is backed 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 will still start at its specified time, but the data will not be 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 immediate restart of the new task. (It may qualify for restart like any other task, if RESTART(YES) is coded in the CEDA DEFINE TRANSACTION command.) On emergency restart, a started task is restarted only if it was started with data written to a **recoverable** temporary storage queue.

EXEC CICS CANCEL requests

Recovery from CANCEL requests during DTB depends on whether:

- Data is being passed to the started task (specified by the FROM data option on the original START command)
- The temporary storage queue used to pass the data is defined as recoverable.

During DTB of a failed task that issued a CANCEL command, CICS recovers both the temporary storage queue and the interval control element (ICE). Thus the effect of the recovery is as if the CANCEL command had never been issued.

If there is no data associated with the START command, or if the temporary storage queue is not recoverable, then the cancelled started task (the ICE) is not recovered and it stays cancelled.

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/ESA Resource Definition Guide*.

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

- REQID operand in the SEND MAP command
- REQID operand in the SEND TEXT command
- REQID operand in the 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.
4. DFHPEP is not given control when a task has abended and CICS is short on storage.

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 181). 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 31).

- 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/ESA 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/ESA Resource Definition Guide*.

Note: Because it is possible to introduce recursions between program checks and abends, you should 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) and TCAM. 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 14, “Handling communication errors” on page 157 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 159.)

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/ESA 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/ESA 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. There is additional information in other parts of this book, and there are other CICS books that contain the full reference information.

For information about resource definition, see the *CICS/ESA Resource Definition Guide*. For further information about system initialization parameters, see the *CICS/ESA 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[4] 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.

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

[4] In the context of these questions, the term “data sets” includes databases.

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 must be preserved while recovery of the data set takes place, you will need to 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[5]). 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.

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 191.

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

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 59).
- 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 161).
- 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 60).

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 might 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 158).

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 persistent sessions restart.

Definitions for recovery functions

In the next few pages, you will 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 81.

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 53. 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
DFHDLI
DFHJRNL

DFHRSEND
DFHRSPLG
DFHSTAND
DFHVTAM

Note that backout occurs for all transactions.

For **file and local DL/I recovery**, you should install the DFHAKP, DFHBACK, DFHDLI, and DFHJRNL groups. You should also take the following options on the CEDA DEFINE TRANSACTION command into account when defining user transactions that will update files and local DL/I databases:

RESTART

This option defines whether CICS will consider restarting a transaction. (“Editing the transaction restart program (DFHREST)” on page 95 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 TRANSACTION definition, see the *CICS/ESA 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 **DBRC and IMS data sharing** for local DL/I, you should install the DFHDLI group.

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 CEDA DEFINE PROFILE command, described under “Your own NEP processors” on page 159. If you are interested in message protection for VTAM terminals, see “Specifying message-protection options for VTAM terminals” on page 87.

To define individual **programs required for recovery and restart**, you need a CEDA DEFINE PROGRAM command 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
```

For a journal with two disk data sets:

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

For further information, see Chapter 8, “Logging and journaling” on page 67.

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/ESA System Definition Guide*.

```
AILDELAY={0-hhmmss}
AIRDELAY={700-hhmmss}
AKPFREQ={200-65535|0}
APPLID=({DBDCCICS|name1}[,name2])    (for data sharing)
CSDBKUP={STATIC|DYNAMIC}
CSDFRLOG={1-99}
CSDRECOV={NONE|ALL|BACKOUTONLY}
DBP={YES|xx}
DBUFSZ={500|number}
DLDBRC={YES|NO}
{DLI|DL1}=({NO|YES|REMOTE}[,COLD])
DLIOLIM={100|number}
DLIRLM={YES|name|NO}                (for data sharing)
FCT={YES|xx|NO}
JCT={YES|xx|NO}
JSTATUS=RESET
NEWSIT={YES|NO}
PGAICTLG={MODIFY|NONE|ALL}
PGAIEXIT={DFHPGADX|name}
PGAIPGM={INACTIVE|ACTIVE}
PISCHD={NO|YES}
PLTPI={YES|xx|NO}
PSDINT={0-hhmmss}
SERIES=PURGE
SRT={YES|xx|NO}
START={AUTO|(AUTO,ALL)|COLD|(COLD,ALL)|LOGTERM|STANDBY}
SYSIDNT={CICS|name}                (for data sharing)
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/ESA 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/ESA 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 parameters 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 CEDA DEFINE TRANSACTION command.

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.

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. It discusses briefly the use of journals for forward recovery, keypointing, the dynamic log, and the catalogs. Starting on page 73, there is a discussion of more advanced functions in journaling: explicit journaling and the activity keypoint program.

System log

You define the system log using the DFHJCT TYPE=ENTRY, JFILEID=SYSTEM macro, which is described briefly on page 64 and more fully in the *CICS/ESA 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 DD name is DFHJ01A), or on **two** datasets (JTYPE=DISK2 in the JCT, where the DD 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 JOUROPT=LRU (if DFHJ01X is available—see below) and PAUSE options. 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 are using local DL/I, you must create an additional disk data set, called DFHJ01X, for use during emergency restart. If you do not use local DL/I, you can still create DFHJ01X for use during emergency restart. See “Characteristics of the system log on disk” on page 23.

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. Another use would be with DL/I and the CEMT SET DLIDATABASE(fileid) RECOVERDB command. This command stops CICS reading and updating DL/I databases, to allow processing by a recovery utility in another region. Part of this processing involves switching journal extents.

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 JCT. 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 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, an MVS archiving job is created. The job is submitted for execution via the JES internal reader. CICS cannot reuse the data set until the archive job has completed.

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

Sample JCL (named DFH$ARCH) for submitting archive jobs is supplied on a library (CICS410.ADFHSAMP). The *CICS/ESA 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.

If the system fails, certain data sets may not have been archived. In particular, if you are using log data set DFHJ01X, after an emergency restart you must archive and reformat it so that it is ready for the next emergency restart. Use DFHJCJFP for formatting (see the *CICS/ESA System Definition Guide* for further information). If you are using the journal archiving facilities supplied by CICS, you need not reformat DFHJ01X because it will be closed correctly.

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 DFHJ01X, 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 DD name is DFHJ01A), or two tape drives (where the DD 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.

Unlabeled or standard-labeled tapes?

CICS supports the use of unlabeled or standard-labeled tapes, specified by the LABEL operand in the journal control table (JCT). If you are using tape logging and DBRC, you must have standard-labeled tapes.

Unlabeled tapes: Unlabeled tapes have no magnetic labels that can be recognized or processed by the operating system. CICS does not control the use of unlabeled tapes. Thus, the operations staff need to control the use of tape volumes manually.

Standard-labeled tapes: When standard-labeled tapes are used, CICS can control which tapes are to be used. It does this by means of a volume manager and **tape-volume descriptor lists** (one list per journal, including the system log) held on the global catalog and retained across all starts. Each list contains the volume identifiers of the series of tapes that are to constitute the journal or system log.

When it is time to mount a tape, the volume manager tells the operator which volume to put up. For a single-drive journal, the mount must, of course, follow the unloading of the previous tape. For a two-drive journal, CICS overlaps its operations and issues an open-ahead command against the drive that is not receiving output. So the mount will normally be completed in advance of use, and should not delay output to that journal.

Whenever a new tape volume is mounted, CICS records the identifier of the **preceding** volume in the header label of the new volume. CICS uses this information during an emergency restart when reading the system log backward (see “Emergency restart” on page 37).

At all times, CICS keeps track of which volumes are occupied (have been written to), which volumes are open, which volume is in use, and which volumes are waiting to be used.

The series of volumes in the tape-volume descriptor list can be **linear** or **cyclic** (as specified by the LAYOUT parameter in the JCT):

- For a **linear** series of tapes, the tape-volume descriptor list shows which tapes have been used and which tapes are available for use.

  With CEMT and EXEC CICS commands, you can edit and query the list, remove volumes no longer required, and add new ones.

  CICS does not allow reuse of any tape volume in a linear series while its identifier is still in the descriptor list.

- For a **cyclic** series of tapes, the tape-volume descriptor list shows which tapes are available for use **or reuse**. The list effectively describes a closed pool of tapes that are used and reused in rotation. In the same way as for a linear series, you can edit and query the list with CEMT and EXEC CICS commands.

You should have operational controls to avoid the possibility of accidentally overwriting a system log tape that might still carry information that would be necessary for forward recovery. Data on a tape is lost as soon as a volume is reopened for output, but not when new data is written.

A cyclic series of tapes has the advantage of operational simplicity, but also means that a tape is automatically released at the end of the cycle when it might still be required for input to a forward recovery run. A linear sequence requires explicit operator action to release a tape for reuse.

Whichever method you choose, ensure that the number of tapes allocated is sufficient to cover all possible forward recovery requirements.

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 81. (For local DL/I 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.

Notes:

1. If you are using DBRC, the RECON data set is used to select those items necessary for DL/I forward recovery. For further information on this, see "IMS database recovery control" on page 203.
2. 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.
3. When you need to recover a data set, you must manually switch the current forward recovery log (using a SET JOU(...) ADVANCE command) in order to close it for use during the recovery operation. However, if the current buffer is not full at the time of the switch, it is not flushed and the contents are written to the new data set. This means you must manually switch again to ensure that all data is available for the forward recovery.

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. 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 74.

The positioning within a journal data set at startup, when two disk data sets are used, is explained in Table 4 on page 28.

Keypointing

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

AKPFREQ must not be set 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 may wish to make use of 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 23). 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 WRITE JOURNALNUM(1) command.

At emergency restart, log records written by the exit program are presented to the XRCINPT 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 user exits, see the *CICS/ESA 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 21).

The DBUFSZ (dynamic buffer size) system initialization parameter influences the initial maximum size of the dynamic log buffer area by means of an algorithm. You 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/ESA 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 91 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/ESA 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. Similar data is available for standard-labeled tape journals, from the volume manager. For information about CEMT commands, see the *CICS/ESA CICS-Supplied Transactions* manual; for programming information about equivalent EXEC CICS commands, see the *CICS/ESA 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/ESA 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 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 will 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/ESA 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.

# Effect of day-light saving time changes

Many countries operate a policy of adjusting clocks by one hour at the beginning and end of Summer to effect what is commonly referred to as ‘day-light saving’. These time changes are also applied to the local time held in computers.

In this discussion about the effect of time changes on CICS, it is assumed that the machine, or hardware, time is never adjusted, and that only the local time offset is altered. Typically, most hardware clocks are set to Greenwich Mean Time (GMT), with an offset value to indicate local time.

## Adjusting local time forwards

A local time change forwards has no effect on CICS operations. For example, moving the clock forward an hour has no effect on the way CICS reads back through the system log during an emergency restart.

To ensure your CICS regions use the correct local time after a time change, without the stopping and restarting the region, issue the CEMT PERFORM RESET command to resynchronize CICS time with the MVS TOD clock.

Adjusting local time backwards

A local time change backwards can affect the operation of a CICS emergency restart in the event of an abnormal termination that occurs after the time change. If the system log at the time of the CICS failure contains active units-of-work that span the point at which local time was changed, CICS emergency restart will fail to recover the relevant data for transaction backout owing to the discontinuity in time-stamps in the system log.

A local time change could also affect the operation of forward recovery utility programs on forward recovery logs.

All CICS system log and journal records are time stamped in local time. For a CICS emergency restart to work correctly, the time stamps of the active records on the system log must not go backwards in time–that is, the records must be in chronological order. If you adjust local time backwards by an hour while CICS is running, the time stamps on system log and journal records become out of sequence. You can avoid problems with emergency restarts during a time change by adopting one of the following procedures.

Procedure 1 (recommended method):

1. Shut down all CICS regions, either normally or forced with the IMMEDIATE option, before changing local time.
2. Adjust the local time backwards.
3. Wait for the length of time by which the local time was adjusted.
4. Restart the CICS regions.

Procedure 2

1. Shut down all CICS regions normally with a successful warm key point.
2. Ensure the system log data sets are archived.
3. Adjust the local time backwards.
4. Re-format the system log data sets.
5. Warm restart the CICS regions, but cold start temporary storage using the TS=(,COLD) system initialization parameter.

Offline processing of log and journal records

Offline processing of CICS system logs and journals by the CICS journal utility program (DFHJUP) is not affected by time changes because DFHJUP does not check the time stamps.

However, user- or vendor-written journal utilities or DFHJUP exit programs, may be sensitive to time changes, and should be checked to ensure that there are no problems posed by time changes.

Forward recovery utilities may also be sensitive to the time sequence of forward recovery log data. If they are, you may need to adopt one of the procedures outlined above, and also consider taking new data set backups.

Chapter 9. Recovering resources

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

- Data files and databases
- Temporary storage
- Intrapartition transient data
- Messages
- Extrapartition transient data.

Recovery of local DL/I resources is described in Part 4, “Recovery in a DL/I environment” on page 193.

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 8-character file name. A CICS file is associated with a VSAM or BDAM data set by one of the following:

- The DSNAME parameter in the file definition
- A CEMT SET FILE DSNAME(name) command
- An EXEC CICS SET FILE DSNAME(name) command
- A DD JCL 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 VSAM access method services (AMS).

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.

Use of DL/I

Decide which databases are to be implemented with DL/I. A decision regarding the use of DL/I and DBCTL is needed early in the design cycle so that database design and implementation can be initiated. The *CICS/ESA CICS-IMS Database Control Guide* describes the advantages of DBCTL compared with local DL/I.

Use of DB2 facilities

Decide which databases are to be implemented with DB2 facilities. A decision regarding the use of DB2 facilities is needed early in the design cycle so that database design and implementation can be initiated. CICS, IMS, TSO, and batch jobs may access DB2 facilities concurrently.

VSAM files

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

VSAM-KSDS
VSAM-RRDS
VSAM-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 such as CICSVR 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 81)
- 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. CICSVR automatically constructs the backout job for you, using an ISPF dialog interface (see Chapter 12, "Forward recovery and backout with CICSVR" on page 125).

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 13, "User exits for transaction backout during emergency restart" on page 151 and "User exits in DFHDBP" on page 94.
- 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 39, 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 "User exits in DFHDBP" on page 94 and Chapter 13, "User exits for transaction backout during emergency restart" on page 151.)

- For all types of VSAM data set:

  - A backout utility such as CICSVR 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. CICSVR automatically constructs the backout job for you, using an ISPF dialog interface (see Chapter 12, "Forward recovery and backout with CICSVR" on page 125).

Basic direct access method (BDAM)

For BDAM files, there is no support for forward recovery via the DFHFCT RECOVERY=ALL and FWDRECOVLOG options. You can implement your own forward recovery support using automatic journaling options.

Backout for BDAM 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 167).

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 77.)

- 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 85).

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, or an online backup can be made using the backup-while-open (BWO) facility. Recovery is always done offline. The BWO facility is described in Chapter 11, “Backup while open (BWO)” on page 97.

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 60). 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 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 and the DFHFCT macro. (Although the DFHFCT macro allows both VSAM and BDAM files to be defined, only BDAM files will be installed by CICS from an assembled FCT at cold-start time. VSAM file entries in an assembled FCT are allowed for migration to the CSD only, and are ignored by CICS at cold-start time.)

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.
3. For information about the backup-while-open (BWO) facility, see Chapter 11, “Backup while open (BWO)” on page 97.

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 in the DFHFCT macro for BDAM files. JNLUPDATE(YES) combined with JNLADD(BEFORE) is equivalent to JREQ=(WU,WN), 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/ESA 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/ESA System Definition Guide* for further information.

Backout of changes to files

To make files backward recoverable, use RECOVERY(ALL|BACKOUTONLY) in a CEDA DEFINE FILE command or LOG=YES in the DFHFCT macro for BDAM files. 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 BDAM and VSAM-ESDS data sets (see “User exits in DFHDBP” on page 94).
2. At **emergency restart**, CICS uses information from the system log. DFHFCBP requires exit code to handle the special case of BDAM and VSAM-ESDS flag deletions (see Chapter 13, “User exits for transaction backout during emergency restart” on page 151).

RECOVERY(ALL|BACKOUTONLY) and 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 173. 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 INQUIRE DSNAME 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

A global user exit, XFCNREC, is provided for any user who wishes 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 INQUIRE DSNAME 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 DSNAME REMOVE or EXEC CICS SET DSNAME REMOVE command or COLD START.

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 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/ESA Customization Guide* for programming information about XFCNREC.

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, if you set SERVREQ=ADD, DELETE, or UPDATE. If you set only SERVREQ=READ and/or BROWSE, 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, 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):
    - FCT entry defined with RECOVERY(NONE): the open proceeds.
    - FCT entry defined with RECOVERY(BACKOUTONLY): 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):
    - FCT entry defined with RECOVERY(NONE): 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): the open proceeds.
    - FCT entry defined with RECOVERY(ALL): the open fails.
  - Base cluster has RECOVERY(ALL):
    - FCT entry defined with RECOVERY(NONE): the open fails.
    - FCT entry defined with RECOVERY(BACKOUTONLY): 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 SET DSNAME REMOVE or EXEC CICS SET DSNAME REMOVE commands. This deletes 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 JID and JREQ settings.

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/ESA Customization Guide*.

Implementing forward recovery with CICS VSAM Recovery MVS/ESA

You can use CICS VSAM Recovery MVS/ESA (CICSVR) to recover lost or damaged VSAM data sets. For details, see Chapter 12, “Forward recovery and backout with CICSVR” on page 125.

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/ESA 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 89.

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 DFHINTRA. (For more information about allocation and space requirements, see the *CICS/ESA 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 will 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 “User exits in DFHDBP” on page 94).

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 13, “User exits for transaction backout during emergency restart” on page 151).

    b. To message caches in temporary storage: one cache for each terminal (see Chapter 17, “Using message caches after emergency restart” on page 183).

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 43).

       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 *CICS/OS/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 will need to 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 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 READQ operations is written to a journal data set, together with the relevant destination identifications. This journaling should be done **immediately** before 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 READQ operations performed on the relevant extrapartition input data set during previous execution of application programs. The same number of 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 could just choose 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** PLT 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/ESA Resource Definition Guide*. For programming information about the special conditions that apply to PLT programs, see the *CICS/ESA Customization Guide*.

Chapter 10. Dynamic transaction backout (DTB)

This chapter describes how to specify DTB. It then goes on to describe the opportunities for including your own logic in the processing of a transaction abend after program level exit code has determined that the task should terminate. You can include logic in exits in the transaction restart program (DFHREST) and in the dynamic transaction backout program (DFHDBP).

This chapter contains Product-sensitive Programming Interface information.

Specifying DTB

"Dynamic transaction backout (DTB)" on page 50 describes the way that DTB works for various resources. The specification of **basic** recovery and restart facilities is described on page 59. In addition, you should note the following:

- DTB is the default for all transactions. For other resources, it is up to you to decide whether to make a resource recoverable. For files, for example, you need to specify RECOVERY(ALL|BACKOUTONLY) if you are using CEDA to define the file, or LOG=YES in the DFHFCT macro for BDAM files.
- 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 BDAM or VSAM-ESDS data sets, you must prepare your own code for the file error exit from DFHDBP (see "User exits in DFHDBP" on page 94).
- 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. You do this by changing the CSD definition of the 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.
- If you specify STAE in the PLIXOPT statement in PL/I programs, the transaction and related resource will be subject to DTB. But DTB will run only if running the PL/I exit routine causes a CICS transaction abend (see "PL/I programs and error handling" on page 173).

Specifying automatic transaction restart

To specify automatic transaction restart:

1. Ensure that you define your resources for recovery.
2. Specify RESTART(YES) in the CEDA DEFINE TRANSACTION command for the transactions that are to be candidates for automatic restart (see "Definitions for transactions and programs" on page 62).
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 85, and “Implementing recoverability of intrapartition transient data” on page 86).

- If a 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 95.

User exits in DFHDBP

DFHDBP has four exits:

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 that can be returned by the exit programs, see the *CICS/ESA Customization Guide*.

If the default return code **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 18, “Backout failure” on page 189. 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.

Return code **UERCBYP** indicates that the error is ignored and backout continues. The file is not flagged as “backout failed”.

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.

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

Coding DFHDBP exits

You may modify recoverable resources in DFHDBP 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 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/ESA Customization Guide*.

Chapter 11. Backup while open (BWO)

This chapter describes the CICS **backup while open (BWO)** facility. It tells you about:

- What benefits BWO offers you
- What other products you need to use BWO
- What data sets can use BWO
- How you request BWO
- What procedures the system administrator needs to consider
- How BWO processing is performed
- BWO and concurrent copy
- An example of a call to DFSMS Callable Services.

Many CICS applications depend on their data sets being open for update over a long period of time. Normally, a backup copy of the data set cannot be taken while the data set is open. Thus, if a failure occurs that requires forward recovery, all updates that have been made to the data set since it was opened must be recovered. This means that all forward recovery logs that have been produced since the data set was opened must be kept. For a heavily used data set that has been open for update for several days or weeks, much forward recovery could be needed.

The BWO facility, together with other system facilities and products, allows you to take a backup copy of a VSAM data set while it remains open for update. Then, only the updates that have been made since the last backup copy was taken need to be recovered. This could considerably reduce the amount of forward recovery that is needed.

What is needed to use BWO

To use the backup-while-open (BWO) support provided by CICS, you can use the following individual IBM-licensed products (or program products that provide equivalent function):

- **MVS/Data Facility Product (MVS/DFP), Version 3 Release 2** (program number 5665-XA3), with the PTFs supplied for the DFP Information APAR II04910, installed on the processor where CICS is running.

  The MVS/DFP IGWAMCS2 callable services module must be installed in the link pack area (LPA). The IGWABWO module, supplied in the SYS1.CSSLIB, must be installed in the LPA or SYS1.CSSLIB must be included in the link list (the library should not be included in the STEPLIB or JOBLIB library concatenations).

- **Data Facility Hierarchical Storage Manager (DFHSM), Version 2 Release 5** (program number 5665-329), which backs up and migrates data within a hierarchy of storage, must be installed on the processor where the BWO backup is made. For more information, see *Data Facility Hierarchical Storage Manager General Information*. If you do not have DFHSM, you must provide automatic class selection (ACS) routines that fulfil the SMS requirements for BWO support.

- **Data Facility Data Set Services (DFDSS), Version 2 Release 5** (program number 5665-327), must be installed on the processor where the BWO backup is made.

Note: During initialization, CICS determines the availability of BWO support by issuing calls to the callable services modules IGWAMCS2 and IGWABWO. CICS also checks on the DFDSS release level by calling the DFDSS modules, ADRRELVL and ADRMCLVL. If access to these DFDSS modules is strictly controlled by an external security manager, such as RACF, security violation messages are issued against the CICS userid, unless the CICS region userid is authorized to access these modules.

In previous releases of CICS, even though BWO was originally shipped with DFP 3.2, DFDSS 2.5, and DFHSM 2.5, it required the installation to use the function in a controlled environment to avoid the data security and data integrity problems documented in the informational APAR II04910. These problems have been solved in Data Facility Storage Management Subsystem/MVS (DFSMS/MVS) Version 1 Release 2, program number 5695-DF1.

Instead of the separate storage management products (MVS/DFP, DFHSM, and DFDSS), you can use DFSMS/MVS 1.2, which is an integration of the separate storage management products that were available earlier. However, if this alternative option is selected, all three DFSMS components listed below must be installed together.

Table 7 cross-references the storage management component names of DFSMS/MVS to the previous names of the individual licensed programs:

Table 7. Cross-reference of DFSMS/MVS product terminology

Component name	**Full DFSMS/MVS name**	**Previous product name**
DFSMSdfp	DFSMS/MVS Data Facility Storage Management Subsystem data facility product	MVS/DFP
DFSMShsm	Data Facility Storage Management Subsystem hierarchical storage manager	DFHSM
DFSMSdss	Data Facility Storage Management Subsystem data set services	DFDSS

Throughout this book, these new DFSMS/MVS product names are used in place of the previous names of the individual licensed programs.

Note that **CICS VSAM Recovery MVS/ESA (CICSVR)**, which performs forward recovery, must be installed on the processor where forward recovery is to be done.

What data sets can use BWO

You can use BWO only for:

- Data sets that are on SMS-managed storage and that have an integrated catalog facility (ICF) catalog.
- VSAM data sets accessed by CICS file control and for the CICS system definition (CSD) file. ESDS, KSDS, and RRDS are supported. ESDS and

KSDS are supported both with and without alternate indexes (AIX). DFSMShsm imposes a limit of many thousands on the number of AIXs for a data set.

BWO is supported at the VSAM sphere level; thus you cannot take BWO backup copies of some sphere components and not others. The first data set opened for update against a VSAM base cluster sets the BWO eligibility for the sphere. This includes base clusters that are accessed through a VSAM path key. For example, if the first data set is defined as eligible for BWO, CICS fails the file-open operation for any subsequent data set that is opened for update against that cluster and which is not defined as eligible for BWO.

You can take BWO volume backups if all data sets that are open for update on the volume are eligible for BWO.

VSAM control interval or control area split

A KSDS (or an ESDS with AIX) that has frequent insert or update activity might not be eligible for BWO, or might be eligible only during periods of reduced activity (for example, overnight). This is because it is possible for a VSAM control interval or control area split to occur during a BWO backup. Then, the integrity of the backup copy cannot be guaranteed because DFSMSdss copies the data set sequentially, and so certain portions of the data set might be duplicated or not represented at all in the backup copy.

DFSMSdfp 3.2 indicates in the ICF catalog that a split has occurred. DFSMShsm and DFSMSdss check the ICF catalog at the start and end of a backup. If a split is in progress at the start of a backup, the backup is not taken. If a split has occurred during a backup, or a split is still in progress at the end of a backup, the backup is discarded.

So, to take a BWO backup successfully, the normal time between splits must be greater than the time taken for DFSMShsm and DFSMSdss to take a backup of the data set.

Data tables: You can use BWO with CICS-maintained data table base clusters. But you cannot use BWO with user-maintained data tables, because no forward recovery support is provided.

AIX: CICS normally uses a base key or a path key to access data in a VSAM base cluster data set. It is also possible, but not normal, for CICS to access AIX records by specifying the AIX name as the data set name. If an AIX data set is used in this way, you cannot define the AIX as eligible for BWO. Instead, the AIX adopts the BWO characteristics already defined for the VSAM sphere.

How you request BWO

You define a file as eligible for BWO with the BACKUPTYPE attribute on the CEDA DEFINE FILE command. You cannot define the backup type with the FCT macro.

If you specify BACKUPTYPE(DYNAMIC), the file is defined as eligible for BWO when the data set is opened. You must also specify RECOVERY(ALL) and FWDRECOVLOG(nn) to request forward recovery support.

BACKUPTYPE(STATIC), the default, defines a file as not eligible for BWO. In this case, if DFSMShsm is to back up a data set, all CICS files currently open for update against that data set must be closed before the backup can start.

All files opened against the same VSAM base cluster must have the same BACKUPTYPE value. That value is established by the first file opened against the cluster; it is stored in the CICS data set name block (DSNB) for the cluster. If the value for a subsequent file does not match, the file-open operation fails.

The BACKUPTYPE value in the DSNB persists across warm and emergency restarts. It is removed by a CICS cold start (unless a backout failure occurs) or by issuing EXEC CICS SET DSNAME ACTION(REMOVE) (or the CEMT equivalent) for the base cluster data set. To do this, all files that are open against the base cluster and via path definition must be closed, and the DSNB must have FILECOUNT = 0 and NORMALBKOUT state.

The BACKUPTYPE attribute is ignored for user-maintained data table base clusters, because no forward recovery support is provided.

BWO for CSD: To use BWO for the CSD file, specify the CSDBKUP=DYNAMIC system initialization parameter. You must also specify CSDRECOV=ALL and CSDFRLOG=nn to request forward recovery support.

Systems administration

The systems administrator must decide which VSAM user data sets are eligible for BWO and then set up the appropriate operating procedures for taking the BWO backup copies and for forward recovery. These procedures should include:

- How to forward recover a data set by using the BWO backup copy, the forward recovery log, and the forward recovery utility to bring the data set to a point of consistency. Users must not have access to the file during the recovery process.
- How to forward recover a data set that may have been damaged while allocated to CICS. This operation may require backout of partially committed units of work during CICS emergency restart, after forward recovery has been done.

The procedures are simpler when using BWO than when not because:

- Backups can be taken more frequently, so there are fewer forward recovery logs to manage. This also reduces the amount of processing that is required to forward recover the data set.
- The point from which forward recovery should start is recorded in the ICF catalog. The forward recovery utility uses this value to automate this part of the forward recovery process. This recovery point is saved with the backup copy and subsequently replaced in the ICF catalog when the backup copy is restored. For more information, see “Recovery point” on page 113.
- During data set restore and forward recovery processing, CICS does not allow files to be opened for the same data set.

Batch jobs

During the batch window between CICS sessions, it is possible for batch jobs to update a data set. Because batch jobs do not create forward recovery logs, any update that is made while a BWO backup is in progress, or after it has completed, would not be forward recoverable. Therefore, non-BWO backups should be taken, at least:

- At the start of the batch window so that, if a batch job fails, it can be restarted; and
- At the end of the batch window, for use with CICS forward recovery processing.

All update activity against the data set must be quiesced while the backups are taken, so that DFSMShsm can have exclusive control of the data set.

After an uncontrolled or immediate shutdown, further BWO backups might be taken by DFSMShsm, because the BWO status in the ICF catalog is not reset. These backups should be discarded; only the non-BWO backups taken at the end of the batch window should be used during forward recovery, together with the CICS forward recovery logs.

Data set security: CICS must have RACF ALTER authority for all data sets that are defined as BACKUPTYPE(DYNAMIC) because CICS needs to update the BWO attributes in the ICF catalog. The authority must apply either to the data set or to the ICF catalog in which the data set is cataloged. For information on defining RACF ALTER authority, see the *CICS/ESA CICS-RACF Security Guide*.

BWO processing

The information in the remainder of this chapter might be required by the system administrator to recover from error situations due to incorrect operating procedures or hardware failures.

The main data fields used by the BWO facility are:

- Attribute flags in the ICF catalog, to control BWO activity. Table 8 on page 102 shows the possible states for these flags. The DFSMSdfp field dictionary name is VVRSMFLG.
- Recovery point in the ICF catalog. This point is the time from which the forward recovery utility must start applying log records. It is always before the time that the last backup was taken. It is recorded in local time format (0CYYDDDF HHMMSSTF) where:

  ```
  C   = century (0=1900, 1=2000, etc.)
  YY  = year
  DDD = day
  HH  = hour
  MM  = minute
  SS  = second
  T   = tenth of a second
  F   = + sign
  ```

  The DFSMSdfp field dictionary name is VVRRDATA.

- The BACKUPTYPE attribute in the file control table (FCT). This is defined by CEDA DEFINE FILE.
- The BACKUPTYPE attribute in the CICS data set name block (DSNB) for the base cluster. This is set at file-open time according to the value in the FCT.

The attribute flags and recovery point in the ICF catalog reside in the SMS subcell of the primary data VSAM volume record (VVR) of the base cluster in the VSAM volume data set (VVDS). There is only one primary base cluster VVR for each VSAM sphere, which is why BWO eligibility is defined at the sphere level. For more information, see the *MVS/ESA Catalog Administrator Guide*.

Table 8. BWO attribute flags in ICF catalog

CICS BWO state	**Flag 1**	**Flag 2**	**Flag 3**
BWO disabled	off	off	off
BWO enabled	on	off	off
BWO enabled and VSAM split in progress	off	on	off
BWO enabled and VSAM split occurred	on	on	off
BWO disabled and VSAM split occurred	off	on	on
Data set is back level. BWO backup restored by DFSMShsm.	on	off	on
Data set is back level. BWO backup restored by DFSMShsm. Forward recovery started but not ended.	off	off	on
Not valid for CICS BWO	on	on	on

BWO processing affects the following operations in a CICS system:

- File opening
- File closing
- Shutdown and restart
- Data set backup and restore
- Journaling and forward recovery logging
- Forward recovery.

Each of these operations is discussed in the following sections. Also, Table 9 on page 105 and Table 10 on page 107 summarize the processing that is done for file opening and file closing.

File opening

Different processing is done for the following three cases when a file is opened for update:

- First file opened for update against a cluster.
- Subsequent file opened for update against a cluster while the previous file is still open (that is, the update use count in the DSNB is not zero).
- Subsequent file opened for update against a cluster after all previous files have been closed (that is, the update use count in the DSNB is zero).

Each of these cases is described later in this section and are referred to as types 1, 2, and 3 respectively in the summary table (Table 9 on page 105).

In all three cases, CICS issues warning message DFHFC5812 if BACKUPTYPE(DYNAMIC) is specified for a VSAM AIX data set that is being opened as a standalone base cluster data set. The AIX data set must default to the BACKUPTYPE already defined for the sphere.

Also, if the file-open operation fails during BWO processing, the ACB will be open. So CICS closes the ACB before indicating the file-open operation has failed. This affects CICS statistics.

If the file is opened for read-only, and the data set ICF catalog indicates that the data set is back level (type 4 in Table 9 on page 105), the file-open operation fails.

Back-level data sets

In all cases, a file-open operation fails with error messages if the ICF catalog indicates that the data set is back level. A back-level data set is one that:

- Has been restored from a backup copy, but not forward recovered
- Has been forward recovered, but the forward recovery operation has not completed successfully
- The ICF catalog indicates is corrupted.

Note: This check[6] occurs irrespective of whether BACKUPTYPE(DYNAMIC) or BACKUPTYPE(STATIC) is specified. Before CICS/ESA 3.2.1, this condition would not have been detected; a back-level data set would have opened successfully.

First file opened against a cluster

The following processing is done for the first file that is opened for update against a VSAM base cluster data set after a CICS cold start. (In this case, the update use count in the DSNB for the base cluster is always zero.)

CICS calls the DFSMSdfp IGWABWO callable service to inquire on the BWO attributes in the ICF catalog.

- If the file was defined with BACKUPTYPE(DYNAMIC), CICS calls IGWABWO to make the data set eligible for BWO and to set the recovery point to the current time. CICS also sets the BACKUPTYPE attribute in the DSNB to indicate eligibility for BWO.

  However, if the ICF catalog indicates that the data set is already eligible for BWO, IGWABWO just sets the recovery point to the current time. CICS issues a warning message; you should discard any BWO backup copies already taken in a previous batch window.

- If the file was defined with BACKUPTYPE(STATIC) and the ICF catalog indicates that the data set is already ineligible for BWO, CICS sets the BACKUPTYPE attribute in the DSNB to indicate ineligibility for BWO.

  However, if the ICF catalog indicates that the data set is currently eligible for BWO, IGWABWO makes it ineligible for BWO and sets the recovery point to

[6] On an initial access to the data set after a restart, this chack also help CICS to detect a previous abend in an application or system during control-area or control-interval splits, and helps prevent possible data-integrity problems—particularly when alternate indices exist as part of the upgate set. This helps the administrator to decide whether to recover from forward-recovery logs or perform transaction backout.

the current time. CICS issues a warning message; you should discard any BWO backup copies already taken in a previous batch window.

If BWO support is requested and the appropriate level of DFSMSdfp (as described in “What is needed to use BWO” on page 97) is not correctly installed on the processor where CICS is running, the first file-open operation fails with error message DFHFC5811. Subsequent file-open operations are allowed, but CICS issues a warning message.

CICS also issues a warning message (DFHFC5813) for the first file-open operation if the appropriate levels of DFSMShsm and DFSMSdss are not installed on the processor where CICS is running. You should ensure that they are installed on the processor where the BWO backup is to be made.

Subsequent files opened when use count is not zero

The following processing is done when a subsequent file is opened for update against a VSAM base cluster data set and the update use count in the DSNB for the base cluster is not zero.

The ICF catalog has already been validated and set by the first file-open operation, so CICS just checks the BACKUPTYPE attributes in the FCT and the DSNB. If they are not consistent, the file-open operation fails with error messages. You must then either correct the CEDA definition, or REMOVE the DSNB after closing all files that are open against the base cluster data set.

Subsequent files opened when use count is zero

The following processing is done when a subsequent file is opened for update against a VSAM base cluster data set and the update use count in the DSNB for the base cluster is zero. This situation could exist in the following cases:

- After a warm or emergency restart of CICS, because the BACKUPTYPE attribute in the DSNB is cataloged in the global catalog (GCD) and is restored at CICS restart
- When all files that are open against a base cluster are closed and then one or more are reopened.

CICS checks the BACKUPTYPE attributes in the FCT and the DSNB. If they are not consistent, the file-open operation fails with error messages. You must then either correct the CEDA definition, or REMOVE the DSNB after closing all files that are open against the base cluster data set. If they are consistent, CICS uses the DFSMSdfp IGWABWO callable service to inquire on the BWO attributes in the ICF catalog.

- If the file was defined with BACKUPTYPE(DYNAMIC), IGWABWO makes the data set eligible for BWO and sets the recovery point to the current time.

  However, if the ICF catalog indicates that the data set is already eligible for BWO, IGWABWO resets the recovery point to the current time. CICS issues a warning message; you should discard any BWO backup copies already taken in a previous batch window.
- If the file was defined with BACKUPTYPE(STATIC) and the ICF catalog indicates that the data set is already ineligible for BWO, the ICF catalog is not updated.

However, if the ICF catalog indicates that the data set is currently eligible for BWO, IGWABWO makes it ineligible for BWO and sets the recovery point to the current time. CICS issues a warning message; you should discard any BWO backup copies already taken in a previous batch window.

Table 9. Summary of BWO processing for file opening

Open Type	BACKUPTYPE in FCT	BACKUPTYPE in DSNB	BWO status in ICF catalog	CICS BWO processing
1	either	undefined	data set back level	File-open operation fails with DFHFC5801 or DFHFC5802 message
1	DYNAMIC	undefined	BWO disabled	Enable BWO and update recovery point in the ICF catalog
1	DYNAMIC	undefined	BWO enabled	Update recovery point in the ICF catalog. Issue DFHFC5808 warning message
1	STATIC	undefined	BWO disabled	No action
1	STATIC	undefined	BWO enabled	Disable BWO and update recovery point in the ICF catalog. Issue DFHFC5809 warning message
2	DYNAMIC	BWO	BWO enabled	No action
2	DYNAMIC	no BWO	BWO disabled	File-open operation fails with DFHFC5807 message
2	STATIC	BWO	BWO enabled	File-open operation fails with DFHFC5807 message
2	STATIC	no BWO	BWO disabled	No action
3	either	either	data set back level	File-open operation fails with DFHFC5801 or DFHFC5802 message
3	DYNAMIC	no BWO	not referenced	File-open operation fails with DFHFC5807 message
3	DYNAMIC	BWO	BWO disabled	Enable BWO and update recovery point in the ICF catalog.
3	DYNAMIC	BWO	BWO enabled	Update recovery point in the ICF catalog. Issue DFHFC5808 warning message
3	STATIC	BWO	not referenced	File-open operation fails with DFHFC5807 message
3	STATIC	no BWO	BWO enabled	Disable BWO and update recovery point in the ICF catalog. Issue DFHFC5809 warning message
3	STATIC	no BWO	BWO disabled	No action
4	n/a	n/a	data set back level	File-open operation fails with DFHFC5801 or DFHFC5802 message

Explanation of fields

Open type

1. First file opened for update against a VSAM base cluster data set after a CICS cold start
2. Subsequent file opened for update against a VSAM base cluster data set after a CICS cold start and update use count in DSNB is not zero

3. Subsequent file opened for update against a VSAM base cluster data set after a CICS cold start and update use count in DSNB is zero
4. First file opened for read-only against a VSAM base cluster data set

BACKUPTYPE in FCT

DYNAMIC
File is defined as eligible for BWO

STATIC
File is defined as not eligible for BWO

either
Either DYNAMIC or STATIC

n/a
Not applicable

BACKUPTYPE in DSNB

BWO
Data set is eligible for BWO

no BWO
Data set is not eligible for BWO

undefined
DSNB BACKUPTYPE is not known until first open for update

either
Either BWO or no BWO

n/a
Not applicable

BWO status in ICF catalog

data set back level
Data set backup copy has been incorrectly restored or the forward recovery is incomplete

BWO disabled
The BWO attributes indicate that the base cluster is not eligible for BWO

BWO enabled
The BWO attributes indicate that the base cluster is eligible for BWO

not referenced
The BWO attributes are not checked because of a mismatch between the BACKUPTYPE attributes in FCT and DSNB

File closing

When the last file that is open for update is closed against a VSAM base cluster data set, CICS uses the DFSMSdfp IGWABWO callable service to update the ICF catalog to indicate that the data set is no longer eligible for BWO and to reset the recovery point to the current time.

If a VSAM split has occurred while a file was open, CICS calls IGWABWO at file-close time to update the ICF catalog to prevent further BWO backups. If DFSMShsm is currently taking a BWO backup, it will discard the backup at the end of the backup operation.

The BWO attributes that indicate that a split has occurred and that the data set is eligible for BWO are restored when the next file is opened for update against the data set. This ensures that DFSMShsm takes the correct action if a split occurs during backup processing which spans CICS updating a file (causing a VSAM split), the file being closed, and then the file being reopened.

When CICS is terminated by a normal shutdown, all CICS files are closed. The ICF catalog is updated to suppress BWO activity during the batch window between CICS sessions. After an uncontrolled or immediate shutdown, or if there is a failure during file closing, the data set remains open and the BWO flags are not reset. See “Shutdown and restart” on page 108.

Table 10. Summary of BWO processing for file closing

Close Type	BACKUPTYPE in FCT	BWO status in ICF catalog	CICS BWO processing
1	DYNAMIC	BWO enabled	Disable BWO and update recovery point in the ICF catalog
1	DYNAMIC	BWO enabled and VSAM split occurred	Disable BWO, remember split, and update recovery point in the ICF catalog
1	STATIC	not referenced	No action
2	DYNAMIC	not referenced	No action
2	STATIC	not referenced	No action
3	n/a	not referenced	No action

Explanation of fields

Close Type

1. The file that is being closed is the last one that is open for update against the VSAM base cluster data set
2. The file that is being closed is not the last one that is open for update against the VSAM base cluster data set
3. The file that is being closed is open for read-only

BACKUPTYPE in FCT

DYNAMIC
File is defined as eligible for BWO

STATIC
File is defined as not eligible for BWO

n/a
Not applicable

BWO status in ICF catalog

BWO enabled
The BWO attributes indicate that the base cluster is eligible for BWO

BWO enabled and VSAM split occurred
The BWO attributes indicate that the base cluster is eligible for BWO and that a VSAM control interval or control area split has occurred while the data set is defined as eligible for BWO

not referenced

The BWO attributes are not checked because other files are open for update against the base cluster or because the file that is being closed is defined as read-only

Restriction for VSAM upgrade set

In some circumstances, it might not be possible to take either BWO or non-BWO backups of a data set. The VSAM UPDATE ACB ENQs for a sphere might remain, even though there are no files open for update against this sphere. This could happen if a VSAM sphere contains an AIX in the upgrade set and the following actions occur:

1. The sphere is opened for update via a VSAM path. This causes VSAM to open for update all upgrade clusters for this sphere.
2. A file is opened for read-only against this sphere.
3. The original VSAM path is closed.

The data set is now ineligible for BWO backups because CICS file control has reset the BWO attributes in the ICF catalog. But, until all open ACBs in the sphere are closed, VSAM will not close the internal ACBs that are open for update, and thus it is not possible to take non-BWO backups either.

To remedy this situation, either:

- Close all ACBs for the sphere, or
- Open for update against the base cluster data set a file that is defined with BACKUPTYPE(DYNAMIC).

Shutdown and restart

CICS shutdown processing was changed for CICS/ESA 3.2.1. File closing is affected by whether the shutdown is **controlled** or **immediate or uncontrolled** (for example, a system abnormal termination that terminates CICS).

Controlled shutdown

During a controlled shutdown, CICS closes all open files defined in the FCT. This ensures that, for files that are open for update and eligible for BWO, the BWO attributes in the ICF catalog are set to a ‘BWO disabled’ state.

If a failure occurs during shutdown so that CICS is unable to close a file, CICS issues warning message DFHFC5804. In this case, you should check the BWO attributes and, if necessary, either use DFSMSdfp IGWABWO callable service to set the attributes, or discard any BWO backups that are taken in the batch window that follows the shutdown.

Immediate or uncontrolled shutdown

During an immediate or uncontrolled shutdown, CICS does not close the files defined in the FCT, and so the BWO attributes in the ICF catalog are not updated.

| You should use the DFSMSdfp IGWABWO callable service to set the attributes
| (see “An assembler program that calls to DFSMS callable services” on page 119
| for an example of how to do this). You should not run any batch jobs before the

next CICS restart. If you do, for releases prior to DFSMS 1.2, discard any BWO backups that are taken in the batch window.

For DFSMS 1.2 onward, the controls in DFSMS allow DFSMSdss to detect a backup that is invalidated if CICS applications are shut down (normally or abnormally) and if batch programs are executed that update the data set while the BWO backup is in progress. This allows DFSMSdss to discard the backup, which prevents DFSMShsm from erroneously discarding the oldest valid backup from the inventory maintained by DFSMShsm.

Restart

At the next CICS restart, the following BWO-dependent actions can occur when a data set is opened for update:

- If the BWO attributes in the ICF catalog are set to the ‘BWO enabled’ state, CICS issues warning message DFHFC5808.
- If the file has been redefined as BACKUPTYPE(STATIC), and CICS has been cold started, and the original base cluster DSNB has been discarded, and the BWO attributes in the ICF catalog are set to the ‘BWO enabled’ state, CICS issues warning message DFHFC5809.
- If the file has been redefined as BACKUPTYPE(STATIC), and CICS has been warm or emergency restarted, and the original base cluster DSNB has been kept, CICS fails the file-open operation with message DFHFC5807.

Data set backup and restore

BWO backups are taken at the VSAM sphere level. You can use DFSMShsm or DFSMSdss to take the backup copy. You are recommended to use DFSMShsm because, without DFSMShsm installed, you must supply automatic class selection (ACS) routines to fulfil the SMS requirements for BWO support.

When you use DFSMShsm, you must still use DFSMSdss as the data mover. You can specify this using the DFSMShsm SETSYS command:

```
SETSYS DATAMOVER(DSS)
```

The DFSMS processing at the start of backup is dependent on the DFSMS release level. For releases prior to DFSMS 1.2, DFSMSdss first checks the BWO attributes in the ICF catalog to see if the data set is eligible for BWO. If it is, the backup is made without attempting to obtain exclusive control and to serialize updates to this data set.

For DFSMS 1.2 onward, DFSMSdss first tries to obtain exclusive control of the data set. If DFSMSdss succeeds, an enqueued form of backup will take place. If this serialization fails, DFSMSdss will check the BWO attributes in the ICF catalog to see if the data set is eligible for BWO. If it is, a BWO backup will be attempted. If it is not eligible, the backup attempt will fail.

This change will prevent DFSMS starting a BWO backup after CICS has abnormally terminated.

At the end of the BWO backup, DFDSS again checks the BWO attributes. If the data set is no longer eligible for BWO, the backup is discarded. Events that cause this situation are:

- File closing during BWO, which sets the ‘BWO disabled’ state
- Start of VSAM split, which sets the ‘BWO enabled and VSAM split in progress’ state
- End of VSAM split, which sets the ‘BWO enabled/disabled and VSAM split occurred’ state.

At the start of a backup, if the state is ‘BWO enabled and VSAM split occurred’, DFSMSdss resets the state to ‘BWO enabled’. Then, if another VSAM split occurs, the backup will be discarded at the end of the backup operation.

VSAM Access Method Services: For releases prior to DFSMS 1.2, BWO is not supported by AMS import and export operations. AMS always serializes the data set before exporting it; and when AMS is used for import, the BWO attributes in the ICF catalog are not updated.

For DFSMS 1.2 onward, AMS supports the import and export of BWO attributes.

Invalid state changes for BWO attributes

CICS, DFSMSdfp, and DFSMSdss can all update the BWO attributes in the ICF catalog. To prevent errors, DFSMSdss fails a BWO backup if one of the following state changes are attempted during the backup:

- From ‘BWO enabled and VSAM split in progress’ to ‘BWO enabled’. This state change could be attempted if:

  1. At the start of data set backup processing, a request is issued to change the ‘BWO enabled and VSAM split occurred’ state to the ‘BWO enabled’ state.
  2. But then, before the ‘BWO enabled’ state is set, a VSAM split occurs and sets the ‘BWO enabled and VSAM split in progress’ state.

  DFSMSdfp must now disallow the pending change to ‘BWO enabled’ (and DFSMSdss fail the backup) because if the split did not finish before the end of the backup, the invalid backup would not be discarded.

- From ‘BWO disabled and VSAM split occurred’ to ‘BWO enabled’. This state change could be attempted if:

  1. At the start of data set backup processing, a request is issued to change the ‘BWO enabled and VSAM split occurred’ state to the ‘BWO enabled’ state.
  2. But then, before the ‘BWO enabled’ state is set, CICS closes the last file opened for update and the data set becomes ineligible for BWO. CICS sets the ‘BWO disabled and VSAM split occurred’ state to ensure that the BWO backup is discarded and that no more BWO backups are taken.

  DFSMSdfp must now disallow the pending change to ‘BWO enabled’ (and DFSMSdss fail the backup) to prevent the possibility of a BWO backup being taken during a subsequent batch window.

Data set restore

When a BWO backup copy of a data set is restored, using DFSMShsm RECOVER or DFSMSdss RESTORE, the data set must be serialized to prevent any updates during the restore operation. When the restore is complete, the BWO attributes in the ICF catalog are changed to a ‘Backup restored by DFSMShsm’ state. CICS cannot open the data set until forward recovery has been completed successfully.

DFSMSdss also resets the recovery point in the ICF catalog to the value it contained when the backup was made. This ensures that forward recovery starts at the correct point. This value should not be used for forward recovery of a non-BWO backup.

Non-SMS managed storage: If a BWO backup is restored in storage that is not SMS-managed, the BWO attributes in the ICF catalog are lost. Thus forward recovery is not possible.

DFSMS 1.2 serialization for BWO during backup and restore

Figure 3 on page 112 shows the BWO serialization for dumping data sets that are open for update and, at the same time, ensures that any update activity that may invalidate the dump is detected. Simultaneous recovery or deletion of the data set while it is being dumped is also prevented.

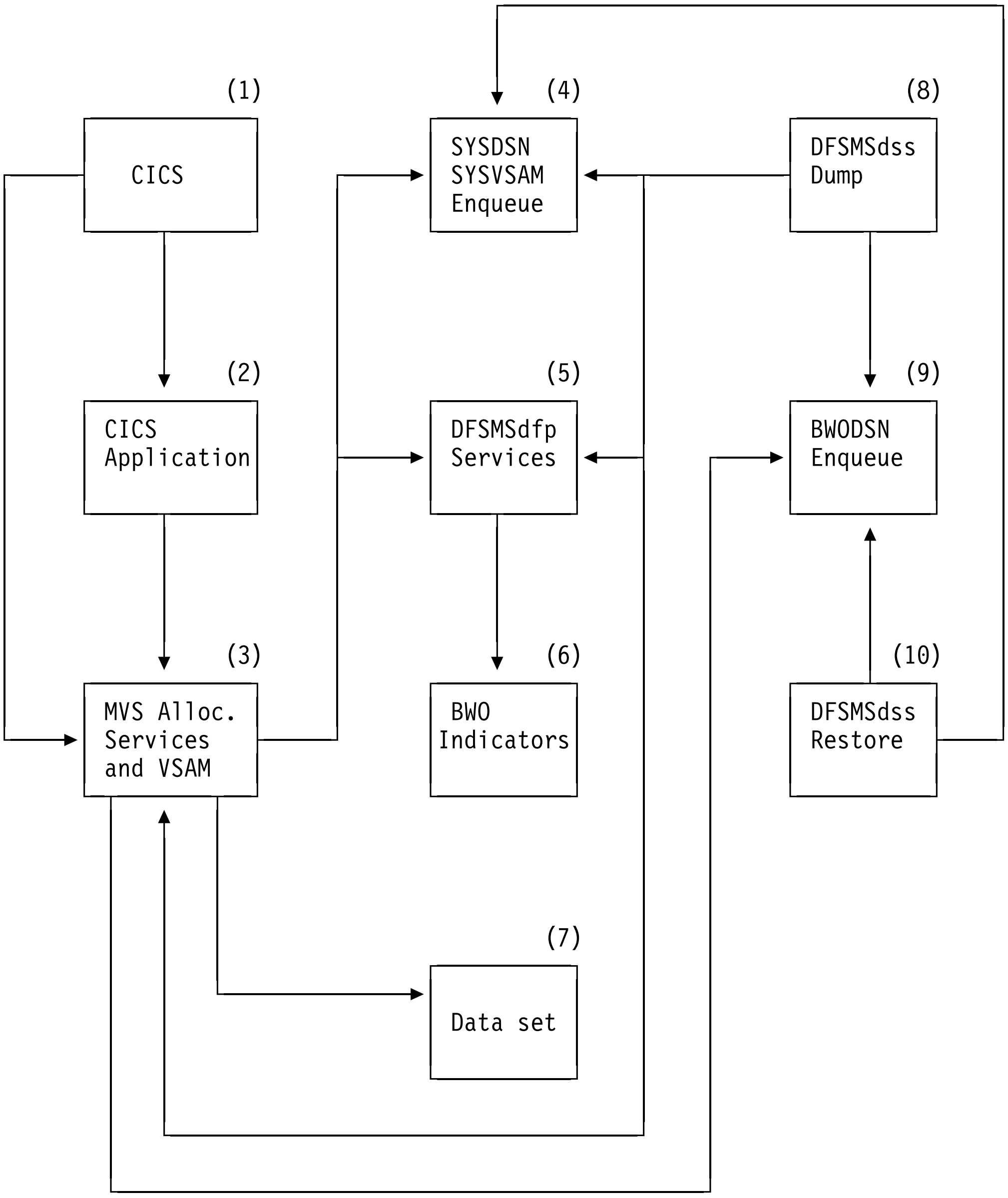


Figure 3. Block diagram for BWO serialization

In Figure 3, a VSAM file control data set (7) is allocated for CICS (1) through MVS allocation services (3) using JCL or dynamic allocation methods. This results in serialization through an enqueue on the name of the data set and the resource name (SYSDSN) (4).

When the VSAM data set (3) is opened, another level of serialization occurs through an enqueue on the names of the components of the VSAM data set and the resource name SYSVSAM (4). For eligible data sets, CICS uses DFSMSdfp services (5) to set a status in the BWO indicators (6) in the catalog entry for the data set.

For a dump operation, DFSMSdss (8) attempts to acquire the SYSDSN, SYSVSAM, and BWO enqueues (9) for the data set. When the enqueue on the cluster name of the data set and resource name of BWODSN is acquired, but not the enqueues for both SYSDSN and SYSVSAM, DFSMSdss uses DFSMSdfp services to get the BWO indicators, and starts to dump the open data set if it is BWO-eligible.

The BWO enqueue is used to prevent more than one DFSMSdss operation, such as a simultaneous dump and restore (10), and to prevent the data set from being deleted while it is being dumped by DFSMSdss.

While the data set is being dumped, a CICS application may update the data set in a manner that invalidates the data set. For example, a control-interval or control-area split may occur. When this happens, VSAM uses DFSMSdfp services to change the BWO status. When the backup of the open data set is completed, DFSMSdss obtains the current BWO indicators and invalidates the dump of the data set if the indicators are different from when the dump was started. When concurrent copy is also used, updates while the data set is being dumped do not cause the dump to be invalidated (see “BWO and concurrent copy” on page 115).

Journaling and forward recovery logging

The forward recovery utility uses the forward recovery logs to recover a base cluster. To do this, it must know:

- The data set to associate with each record on the logs
- The point from which to start recovery.

Data sets

Each data set after-image record on the log is associated with a file name. But there might be many files associated with the same data set. So, when a file is opened, the association between the file and the data set is recorded on the log by a tie-up record (TUR). For non-BWO backups, the forward recovery utility uses this TUR to apply the log records to the correct data sets.

For BWO backups, it is usually not necessary for the forward recovery utility to process a log from file-open time. So, the TURs for all open files are written regularly on the log during activity-keypoint processing, and the time that they are written is recorded. To reduce the number of TURs if the activity keypoint frequency is high (such as for some large XRF systems), CICS ensures that there is at least 30 minutes separation between sets of TURs on the log.

If you use more than one forward recovery log, the set of TURs written at an activity keypoint will be divided between them.

Recovery point

The recovery point is a time that can be converted to a position on a forward recovery log. Recovery of the data set requires only the records that are written after that position. Thus all previous records can be ignored by the forward recovery utility.

The recovery point is stored in the ICF catalog. It is set when the first file is opened for update against the data set, and updated during activity-keypoint processing and when the file is closed.

The recovery point is not the time of the current keypoint, as there might still be some uncommitted log records that have not been forced. Instead, it is the time of the start of the last keypoint that wrote a complete set of TURs and that completed earlier than the oldest uncommitted write to a forward recovery log. For more information, see “How CICS calculates recovery points” on page 116.

Notes:

1. Only one new recovery point is calculated during an activity keypoint. It is used for all data sets that are open for update and eligible for BWO. Thus a long-running task that updates a data set that uses BWO will affect the amount of forward recovery needed for all data sets.
2. If you disable activity keypointing in your system (by specifying the AKPFREQ system initialization parameter as =0), BWO support is seriously affected because, after the file-open operation, no more TURs are written and the recovery point is not updated. So, forward recovery of a BWO data set must take place from the time that the data set was first opened for update.

Forward recovery

CICSVR fully supports BWO. If you do not use CICSVR, you must ensure that your forward recovery utility is able to:

- Recognize whether a backup was made with BWO or not. The DFSMShsm ARCXTRCT macro can be used to determine this.
- Use the BWO attributes and recovery point in the ICF catalog. It should use the DFSMSdfp IGWABWO callable service to do this. See “An assembler program that calls to DFSMS callable services” on page 119 for a sample program that will do this.
- Recognize the additional TURs on the forward recovery logs.
- Recognize after-images that have already been applied to the data set.

The forward recovery utility should ALLOCATE, with DISP=OLD, the data set that is to be recovered. This prevents other jobs accessing a back-level data set and ensures that data managers such as CICS are not still using the data set.

Before the data set is opened, the forward recovery utility should set the BWO attribute flags to the ‘Forward recovery started but not ended’ state (see Table 8 on page 102). This prevents DFSMShsm taking BWO backups while forward recovery is in progress. To prevent CICS opening a back-level data set, the utility should perform this state change for all data sets in a system that supports BWO, even if some do not use BWO.

The forward recovery utility should use the BWO time stamp for the data set in the ICF catalog, set by DFSMSdss when the data set is restored, to determine the starting point in the forward recovery log to start the forward recovery.

If forward recovery completes successfully, the utility should set the BWO attributes to the ‘BWO disabled’ state before the data set is closed.

If forward recovery does not complete successfully, the utility should leave the BWO attributes in the ‘Forward recovery started but not ended’ state to ensure that CICS does not open a back-level data set.

If forward recovery does not complete successfully, you should:

1. Determine and correct the reason for the failure
2. Delete the partially-recovered data set

3. Restore the backup copy
4. Reattempt forward recovery.

Note: The EXEC CICS SET DSNAME RECOVERED system programmer command (or the CEMT equivalent) resets the BWO attributes in the ICF catalog to indicate ‘BWO disabled’. If you use this command for a data set that has been restored but not forward recovered, and then you subsequently open this data set, CICS will be unaware that forward recovery has been overridden and CICS may access back-level data. However, in exceptional circumstances, it may be necessary to allow CICS to access back-level data. This command has been provided to allow this to happen.

Alternatively, if you use a VSAM forward recovery utility that does not update the BWO attributes during forward recovery, you may use these commands to reset the backup-restored-by-DFSMShsm state prior to subsequent CICS file control access.

Recovering VSAM spheres with AIXs

Before you can forward recover a data set that was restored from a copy made using BWO, ensure that no AIXs are in the upgrade set. CICSVR checks the upgrade set of data sets restored from BWO copies and issues a message if AIXs are found.

To forward recover such a data set, after the restore, use the AMS ALTER or DELETE command to remove or delete the AIXs from the upgrade set. After forward recovery has completed successfully, you can re-create the upgrade set by rebuilding the AIXs using the AMS BLDINDX command.

BWO and concurrent copy

Concurrent copy improves BWO processing by eliminating the invalidation of a BWO dump because of updates to the data set. The following is a comparison of various kinds of dumps you can ask for.

- Normal dump. Use of the data set must be quiesced so that serialization is obtained, the data set is dumped, and serialization is released. The data set cannot be used for the entire time.
- Concurrent copy dump. Use of the data set must be quiesced so that serialization is obtained, concurrent copy utilization is completed within a very short time (compared with the actual time to dump the data set), serialization is released, and the data set is dumped. The data set can be used after concurrent copy initialization is complete.
- BWO dump. Serialization is attempted but is not required, and the data set is dumped. If it is eligible for BWO, the data set is dumped without serialization and can remain in use for the entire time, but the dump can be invalidated by update activity to the data set.
- BWO dump using concurrent copy. Serialization is attempted but is not required, concurrent copy initialization is completed, and the data set is dumped. If it is eligible for BWO, the data set is dumped without serialization and can remain in use for the entire time, and updates that occur do not cause the dump to be invalidated.

To use concurrent copy, specify the CONCURRENT keyword when you use DFSMShsm to dump BWO data sets.

BWO and backups

- Backups taken without the BWO function require CICS to close VSAM files. Concurrent copy without BWO requires CICS to close the VSAM files when the concurrent copy session is being established in the control unit, which may require: a CICS shutdown to prevent:

  – In-flight transaction failures

  – Disruption of DB2 and/or IMS DL/I data access if the transactions map to data residing in DB2 or DL/I

  – Disruption of related applications.

  These are some of the major limitations of backups taken without BWO and concurrent copy.

- The BWO function allows backups to be taken by DFSMSdss when applications are running in a continuous 7x24 or 5x24 operation while the data set is open for update with full data integrity of copied data. This is feasible only for CICS VSAM file control data sets for which CICS creates forward-recovery logs. Long-running transactions, automated teller machines, and 7x24 or 5x24 applications require the database to be up and running when the backup is being taken.

- The concurrent copy function used along with BWO by DFSMSdss allows backups to be taken with integrity even when control-area and control-interval splits and data set additions (new extents or add-to-end) are occurring for VSAM key sequenced data sets.

Program requirements

CICS/ESA 3.3 (program number 5685-083) or higher with APAR PN48447 or CICS/ESA 4.1 is required for the definition of the VSAM files to be backed up with the BWO and concurrent copy functions of DFSMS/MVS 1.2

CICS VSAM Recovery (CICSVR) Version 2 (program number 5695-0120) or higher is required for forward recovery and backout of CICS VSAM data sets backed up with BWO and concurrent copy functions of DFSMS/MVS 1.2.

Hardware requirements

The concurrent copy function is supported by the IBM 3990 Model 3 with the extended platform and the IBM 3990 Model 6 control units.

How CICS calculates recovery points

This section contains Diagnosis, Modification, or Tuning information.

When the recovery point is reset during activity-keypoint processing, it is always set to the time that an activity keypoint started to write a complete set of TURs to the forward recovery logs. CICS selects the latest, previous activity keypoint that fulfils both of the following conditions:

- It completed before the oldest uncommitted write to a forward recovery log.
- It wrote a complete set of TURs to the logs.

How CICS evaluates these two conditions is described below.

Oldest uncommitted write: CICS creates a chain of journal thread control blocks (JTCs) for each logical unit of work (LUW). There is a JTC for each forward recovery log used by the LUW. Each JTC contains the time of the first write to the associated log. This time is known as the *fuzz point* because CICS does not know whether the updates described on the log have been written to the data set or not.

When the LUW is committed (that is, at the syncpoint), its JTCs are freed. Thus, the oldest time that is recorded in the existing JTCs (for all LUWs) defines the time of the oldest uncommitted write. At each activity keypoint, this time is found and stored in the JCT header prefix. It is known as the *minimum fuzz point*.

Complete set of TURs: CICS creates a chain of keypoint directory elements (KPDEs). Each KPDE records the time that a complete set of TURs was written on the forward recovery logs.

CICS sets the recovery point, in the ICF catalog, from the KPDE with the time that is earlier than and nearest to the minimum fuzz point. This time is also stored in the JCT header prefix, where it is known as the *clean keypoint time*. Then all KPDEs that are older than this time are removed from the chain.

Notes:

1. CICS ensures that at least 30 minutes elapse between BWO processing at activity keypoints.
2. If no KPDE has a time earlier than the minimum fuzz point, CICS uses the clean keypoint time from the previous activity keypoint. If no time has been set yet, the recovery point is not updated.
3. No BWO processing is done at the first activity keypoint, during system initialization.

Figure 4, together with the explanation below, describes how CICS calculates recovery points at activity keypoints.

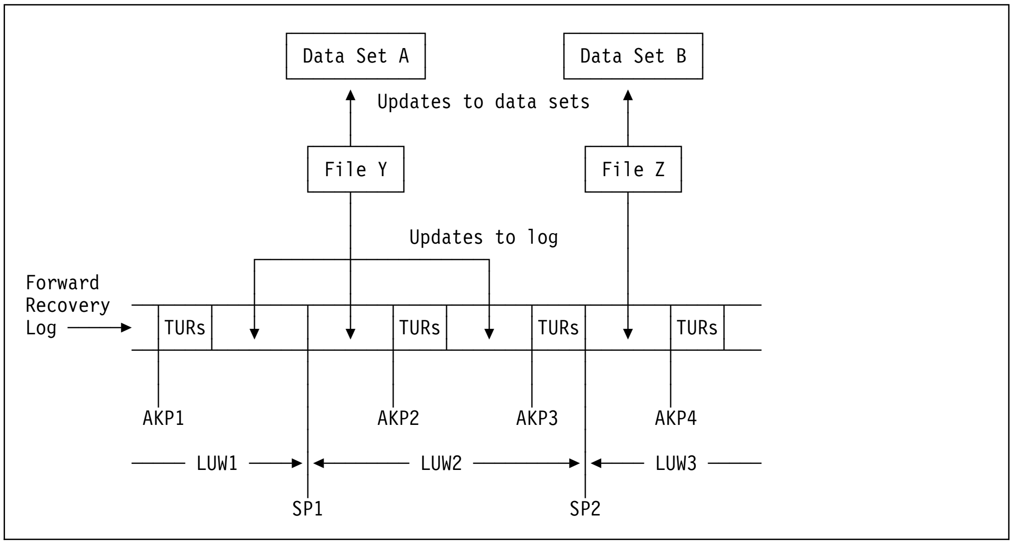


Figure 4. How CICS calculates recovery points

Explanation of Figure 4 on page 117

1. For simplicity, the figure shows only one active task (comprising LUW1, LUW2, and LUW3) and only one forward recovery log (used by two data sets).

2. The figure shows four activity keypoints (AKP1, AKP2, AKP3, and AKP4) and two syncpoints (SP1 and SP2).

3. It is assumed that at least 30 minutes elapse between the activity keypoints. Therefore, each activity keypoint writes a complete set of TURs to the forward recovery log and, if appropriate, updates the recovery point.

4. At AKP2, the fuzz point for LUW2 is the time of the second update of the log for file Y. This is also the minimum fuzz point when no other LUWs are running. The KPDE created by AKP1 contains the time that is earlier than and nearest to the minimum fuzz point. So AKP2 sets the clean keypoint time in the JCT header prefix and the recovery point in the ICF catalog to the time in that KPDE. It then frees the KPDE.

5. At AKP3, the fuzz point and minimum fuzz point are still the time of the second update of the log for file Y. However, there are now no KPDEs with a time earlier than the minimum fuzz point. So the clean keypoint time and recovery point are not changed.

6. At AKP4, the fuzz point for LUW3 is the time of the first update of the log for file Z. This is also the minimum fuzz point when no other LUWs are running. The KPDE created by AKP3 contains the time that is earlier than and nearest to the minimum fuzz point. So AKP4 sets the clean keypoint time in the JCT header prefix and the recovery point in the ICF catalog to the time in that KPDE. It then frees the KPDEs created by AKP2 and AKP3.

An assembler program that calls to DFSMS callable services

```
*ASM XOPTS(CICS,NOEPILOG,SP)
*
*   A program that can be run as a CICS transaction to Read and Set
*   the BWO Indicators and BWO Recovery Point via DFSMS Callable
*   Services (IGWABWO).
*
*   Invoke the program via a CICS transaction as follows:
*
*   Rxxx STEVE.BAKERS.DATASET
*   Sxxx 100 STEVE.BAKERS.DATASET
*
*   Where:
*    Rxxx and Sxxx are the names of the transactions that will invoke
*    this program. Specify Rxxx to read and Sxxx to set the BWO
*    attributes.
*    STEVE.BAKERS.DATASET is the name of your data set
*    100 is the value the BWO indicators will be set to.
*   The BWO Recovery Point time will be set to the current date and
*   time returned from the CICS ASKTIME function.
*
DFHEISTG DSECT
INDATA   DS   0CL53    Input data stream
*
* First character of tran id indicates transaction function
*
TRANFUNC DS   C        First char of tran id - S=SET R=READ
         DS   4C       Remainder of tran id and space
BWOC1    DS   C        First BWO indicator
BWOC2    DS   C        Second BWO indicator
BWOC3    DS   C        Third BWO indicator
         DS   C        Space
DSNAMES  DS   44C      Target data set name 1-44 chars
*
* 2 possible formats of input line, so overlay the definitions
*
         ORG  INDATA
         DS   5C       Tran id and space
DSNAMER  DS   44C      Target data set name 1-44 chars
         DS   4C       Filler
*
INLENGTH DS   H        Length of input data stream
*
```

```
* Parmlist for IGWABWO call
*
PARMLST  DS 10A
RETCODE  DS   F        Return code
REASON   DS   F        Reason
PROBDET  DS   D        Problem determination code
FUNC     DS   F        Function
READ     EQU  0           Read
SET      EQU  1           Set
DSNLEN   DS   F        Data set name length
DSN      DS   44C      Data set name
BWOFLAGS DS   03F      BWO indicator flags
         ORG  BWOFLAGS
BWOF1    DS   F           BWO indicator 1
BWOF2    DS   F           BWO indicator 2
BWOF3    DS   F           BWO indicator 3
BWOTIME  DS   D           BWO recovery point time
RESERVED DS   2D          Reserved
*
* Success message
*
SUCMSG   DS 0CL66         Define storage for success message
         DS 30C
DATEVAL  DS 8C            Date value from BWO recovery point
SUCMSG1  DS 8C            Message text
TIMEVAL  DS 8C            Time value from BWO recovery point
SUCMSG2  DS C             Message text
READMSG  DS 0CL11         If function = READ put out BWO flags
         DS 7C            Message text
BWOVAL1  DS C             BWO indicator 1
BWOVAL2  DS C             BWO indicator 2
BWOVAL3  DS C             BWO indicator 3
         DS C             Message text
*
DATETIME DS D             Current date and time value
*
RECOVPT  DS 0D            BWO recovery point
DTZERO   DS B                Date dword
DTCENTRY DS B
DTDATE   DS 5B
DTSIGN1  DS B
*
DTTIME   DS 6B               Time dword
DTTENTHS DS B
DTSIGN2  DS B
*
RECOVPTP DS 0D            Packed recovery point
DATEPACK DS F             Packed version of date
TIMEPACK DS F             Packed version of time
*
         DFHREGS
SMBAKER  CSECT
SMBAKER  AMODE 31
*
```

```
* Initialise INTO field for RECEIVE
*
         MVC  DSNAMER(48),BLANKS
         MVC  INLENGTH(2),INMAXLEN
*
         EXEC CICS RECEIVE INTO(INDATA) LENGTH(INLENGTH)
*
         CLI  TRANFUNC,C'S'       Set or Read call?
         BNE  SMBREAD
*
* Set up the parameters for a SET call
*
         SR   R4,R4
         LA   R4,SET(0)
         ST   R4,FUNC             Set function
         MVC  DSN(44),DSNAMES     Set data set name
         LH   R4,INLENGTH
         S    R4,PRELENS          Subtract tran id + space + BWO ind
         ST   R4,DSNLEN           Set data set name length
*
         EXEC CICS ASKTIME ABSTIME(DATETIME)
         EXEC CICS FORMATTIME ABSTIME(DATETIME) YYDDD(DTDATE)          *
              TIME(DTTIME)
*
         PACK KEYWORK(5),RECOVPT(9)   Packed date field
         MVC  DATEPACK(4),KEYWORK
         PACK KEYWORK(5),RECOVPT+8(9) Packed time field
         MVC  TIMEPACK(4),KEYWORK
         XC   RECOVPTP(1),RECOVPTP    Set century 0=1900, 1=2000
         OI   RECOVPTP+3,X'0F'        Set +ve sign for date
         OI   RECOVPTP+7,X'0F'        Set +ve sign for time
         MVC  BWOTIME(8),RECOVPTP     Set BWO recovery point time
*
         EXEC CICS SYNCPOINT
*
         MVC  BWOFLAGS(12),ZEROES
         LA   R4,1(0)
         CLI  BWOC1,C'0'
         BE   SMBBIT2
         ST   R4,BWOF1            Set BWO indicator 1 if required
SMBBIT2  DS   0H
         CLI  BWOC2,C'0'
         BE   SMBBIT3
         ST   R4,BWOF2            Set BWO indicator 2 if required
SMBBIT3  DS   0H
         CLI  BWOC3,C'0'
         BE   SMBCONT
         ST   R4,BWOF3            Set BWO indicator 3 if required
         B    SMBCONT
SMBREAD  DS   0H
         CLI  TRANFUNC,C'R'
         BNE  SMBABORT            If tran id not R or S then abort
*
```

```
* Set up the parameters for a read call
*
         SR   R4,R4
         LA   R4,READ(0)
         ST   R4,FUNC             Set function
         MVC  DSN(44),DSNAMER     Set data set name
         LH   R4,INLENGTH
         S    R4,PRELENR          Subtract tran id + space
         ST   R4,DSNLEN           Set data set name length
SMBCONT  DS   0H
*
* OK, our parameters are set up, so create the address list, and make
* the call
*
         LOAD EP=IGWABWO,ERRET=SMBABORT
         LR   R15,R0
         LA   R1,PARMLST          R1 -> parmlist
         LA   R4,RETCODE
         ST   R4,0(R1)            Pass addr of return code
         LA   R4,REASON
         ST   R4,4(R1)            Pass addr of reason code
         LA   R4,PROBDET
         ST   R4,8(R1)            Pass addr of problem determination
         LA   R4,FUNC
         ST   R4,12(R1)           Pass addr of function required
         LA   R4,DSNLEN
         ST   R4,16(R1)           Pass addr of data set name length
         LA   R4,DSN
         ST   R4,20(R1)           Pass addr of data set name
         LA   R4,SEL
         ST   R4,24(R1)           Pass addr of selection mask
         LA   R4,BWOFLAGS
         ST   R4,28(R1)           Pass addr of BWO flags
         LA   R4,BWOTIME
         ST   R4,32(R1)           Pass addr of BWO recovery point
         LA   R4,RESERVED
         ST   R4,36(R1)           Pass addr of reserved field
         BALR 14,15               Call IGWABWO
*
* Back from the call, check return code
*
         SR   R4,R4
         CL   R4,RETCODE          Check return code
         BNE  SMBABORT
*
```

```
* All OK, set up minimum success message, decide if we need more
*
         MVC  SUCMSG(38),SUCTXT       Set up message text
         MVC  SUCMSG1(8),SUCTXT1
         MVC  SUCMSG2(1),SUCTXT2
         UNPK KEYWORK(9),BWOTIME(5)   Make date printable
         TR   KEYWORK(8),HEXTAB-C'0'
         MVC  DATEVAL(8),KEYWORK
         UNPK KEYWORK(9),BWOTIME+4(5) Make time printable
         TR   KEYWORK(8),HEXTAB-C'0'
         MVC  TIMEVAL(8),KEYWORK
         CLI  TRANFUNC,C'S'           If READ then print BWO flags
         BNE  SMBREADO
*
* Got all the info we need, so put it out and exit
*
         EXEC CICS SEND TEXT FROM(SUCMSG) LENGTH(55) ERASE WAIT
*
         B    SMBEXIT
*
* It's a read so we also need the BWO flags for output
*
SMBREADO DS   0H
         MVC  READMSG(11),READTXT     Set up message text
         MVC  BWOVAL1,BWOF1+3
         OI   BWOVAL1,X'F0'           Set BWO indicator 1
         MVC  BWOVAL2,BWOF2+3
         OI   BWOVAL2,X'F0'           Set BWO indicator 2
         MVC  BWOVAL3,BWOF3+3
         OI   BWOVAL3,X'F0'           Set BWO indicator 3
*
* Now send the message
*
         EXEC CICS SEND TEXT FROM(SUCMSG) LENGTH(66) ERASE WAIT
*
SMBEXIT  DS 0H
         EXEC CICS RETURN
*
SMBABORT DS  0D
         EXEC CICS SEND TEXT FROM(FAILMSG) LENGTH(19) ERASE WAIT
*
         EXEC CICS RETURN
*
```

```
* Constant declarations
BLANKS   DC   48C' '
INMAXLEN DC   H'53'
ZEROES   DC   3F'0'
PRELENS  DC   F'9'
PRELENR  DC   F'5'
SUCTXT   DC   C'IGWABWO call completed Date = '
SUCTXT1  DC   C' Time = '
SUCTXT2  DC   C'.'
READTXT  DC   C' BWO =    .'
FAILMSG  DC   C'IGWABWO call failed'
KEYWORK  DC   CL9' '
HEXTAB   DC   C'0123456789ABCDEF'
*
* Constant for IGWABWO SELECT parameter
*
SEL      DC   F'3'         Interested in BWO flags & recov point
*             F'1'         Interested in BWO flags
*             F'2'         Interested in BWO recovery point
*             F'3'         Interested in BWO flags & recov point
         END SMBAKER
```

Chapter 12. Forward recovery and backout with CICSVR

This chapter describes how you can use IBM CICS VSAM Recovery MVS/ESA (CICSVR), Program Number 5695-010, to recover lost and damaged VSAM data sets.

Why should you have a forward recovery and backout utility?

Without a forward recovery or backout utility, lost and damaged data sets must be recovered manually. If each update is reentered manually, the possibility of introducing errors increases, threatening the integrity of your VSAM data.

If you have high transaction volumes between backups, you want to avoid long work interruptions due to failed or lost data sets. CICSVR minimizes your recovery labor and eliminates the possibility of introducing data errors in your recovery of data sets.

How CICSVR can help you

CICSVR has automated forward recovery and backout functions, and helps you recover your CICS VSAM data sets in these situations:

- Physical loss or damage of VSAM data, which might be because of:
  - Damage caused by a device failure
  - A part of a disk that is inaccessible
  - Erasure of the data
  - A disaster such as a fire
- Loss of logical data integrity because a failure occurred during CICS transaction backout, or during emergency restart backout
- Loss of logical data integrity because a CICS application incorrectly updates your VSAM spheres

CICSVR recovers all updates made before the problem occurred and, if either dynamic transaction backout (DTB) or emergency restart backout has run and subsequently failed, CICSVR backs out partially completed transactions.

CICSVR features

CICSVR:

- Automatically determines the backups and journals required for recovery, if you have DFSMShsm
- Has an ISPF dialog interface that complies with Common User Access (CUA)
- Performs forward recovery to recover VSAM data sets that have been lost or damaged
- Provides backout to remove uncommitted updates present in the VSAM data sets
- Works with backups taken using the backup-while-open (BWO) facility
- Updates VSAM spheres directly to the base cluster. It applies all changes that were made to the base cluster, including those that were made to paths.

- Recovers multiple data sets in a single run
- Recovers VSAM KSDS, ESDS, and RRDS
- Can work independently of CICS
- Produces reports showing job statistics

CICSVR benefits

CICSVR:

- Helps you recover your data quickly and easily
- Supports 24-hour CICS availability
- Reduces the risk of human error by creating the job for you
- Saves time by reducing the VSAM data set outage time
- Stores all the information you need for recovery

How CICSVR protects your data

CICSVR recovers lost and damaged VSAM data sets by reapplying all changes to your data after the most recent backup. CICSVR also ensures that your data is logically consistent by backing out any uncommitted updates. CICSVR consists of these recovery functions:

- Complete recovery
- Forward recovery
- Backout

Complete recovery

Sometimes you must back out uncommitted changes after forward recovery of a data set. Assume that a VSAM data set is damaged while CICS is updating it. The task will encounter an I/O error when accessing the data set. The program will abend if it is written in the recommended way.

At this point, if the task left uncommitted data on the data set, CICS will run dynamic transaction backout (DTB). Because the data set is damaged, DTB will encounter the same I/O error and will not succeed in backing out the updates. To summarize:

- The data set is damaged and unusable.
- DTB has failed to back out some changes.
- The log shows uncommitted updates on the data set.
- CICS has closed the file because of DTB failure, and has marked the data set as requiring backout.

To recover the data set, you need CICSVR complete recovery. Complete recovery consists of forward recovery followed by, if needed, backout.

CICSVR restores the data set from a DFSMShsm backup or dump, if one exists. If you do not use DFSMShsm, restore the backup before running CICSVR. This re-creates the original data. Then CICSVR forward recovery applies the after-images to the restored data set. Note that after-images for uncommitted updates are applied just as all other after-images. The data set will be at exactly the same level as it was when the damage occurred—it will contain all the uncommitted updates.

Next, the CICSVR backout function removes the uncommitted updates. CICSVR reads the before-images from the system log and applies them to the restored, forward-recovered data set. This removes the uncommitted updates from the data set.

For instructions on running CICSVR complete recovery, see “Running CICSVR complete recovery” on page 132. For more information about CICSVR complete recovery, see the *IBM CICS VSAM Recovery MVS/ESA User’s Guide and Reference*.

Forward recovery

CICSVR forward recovery reapplies all changes made to the sphere after the backup.

If a VSAM sphere that CICS updated is damaged and unusable, CICSVR forward recovery is needed to re-create the VSAM sphere. This is achieved by:

- Restoring the VSAM sphere from the backup copy
- Applying the after-images from the archived forward-recovery logs to the restored VSAM sphere, using the CICSVR forward-recovery function

After you have restored the data set, the CICSVR forward-recovery function reads the forward-recovery log, looking for after-images that are applicable to the VSAM sphere. It will update the restored copy of the original data by applying each after-image to it. When the forward-recovery run is finished, CICSVR has re-created the updated data.

The CICSVR forward-recovery function is described in the *IBM CICS VSAM Recovery MVS/ESA User’s Guide and Reference*.

Backout

The CICSVR backout function reruns a backout that failed. It is needed if a failure occurs during DTB or during transaction backout at emergency restart. The CICSVR backout function (also called batch backout) removes uncommitted updates from a VSAM sphere.

A data set contains uncommitted updates. The situation is:

- A task abended, leaving uncommitted data on the data set.
- DTB failed, for example, because of a temporary I/O error on the a data set.
- CICS backout-failure control closed the file, and set the data set status to FAILED.

The CICSVR backout function is needed to remove the uncommitted updates.

You run the CICSVR backout function, which updates the data set. It reads before-images from the archived system log backward, and applies them to the data set. The result is that all uncommitted updates are backed out, and CICS can use the data set again.

After completing the backout, the CICS console operator, or a CICS transaction, issues a command to reopen the files to CICS.

The CICSVR backout function is described in the *IBM CICS VSAM Recovery MVS/ESA User’s Guide and Reference*.

Storing information for recovery using the CICSVR archive utility

Use the CICSVR archive utility to extract information from CICS journals and to store it in the recovery control data set (RCDS). CICSVR lets you manage the contents of the RCDS using the dialog interface. You can remove information about data sets and journals you no longer need. You can also set an automatic function to delete and uncatalog CICS journals after a specified time. For added security, three copies of the recovery control data set are maintained.

Using CICSVR with DFSMShsm also ensures that you have all information about backups. You need not keep a separate record of backups or journals for each data set. CICSVR reduces the risk of errors in recording dates, times, and data set names.

The archive utility can also copy a CICS journal and register this information in the RCDS. You should create your CICSVR archive JCL in the CICS journal partitioned data set (JPDS), for use by the CICS automatic journal submission program.

For more information about the CICSVR archive utility, see the *IBM CICS VSAM Recovery MVS/ESA Implementation Guide*.

Automated job creation

To recover data, start the CICSVR dialog interface. Select the data sets you want to work with, and then select a CICSVR function (see “Running CICSVR complete recovery” on page 132).

Choose between forward recovery, backout, or complete recovery. Select complete recovery to perform both forward recovery and backout on the data sets you have selected. CICSVR will check which functions are needed for your data sets when the job is created. The complete recovery option will also restore your data sets, if you are using DFSMShsm.

CICSVR guides you through the creation of your recovery job. An extensive help function ensures that you always know what to do next.

Use default values stored by CICSVR on previous runs, or use your own values. You decide which values to save for next time. You can browse and, if necessary, edit the JCL before you submit it. CICSVR produces updated data sets and a report showing the job statistics.

CICSVR and 24-hour CICS availability

You need not withdraw services and close the CICS system to protect your data. You can maintain 24-hour availability of your data and ensure its integrity with CICSVR.

With BWO, you can back up data while the data sets are open and being updated. CICSVR can recover using BWO copies, ensuring that the data is protected from loss at all times.

If the data should become damaged or inaccessible, CICSVR can use BWO backups to retrieve the data. Because you can make backups without interrupting your CICS systems, you can make them more often.

Planning for CICSVR

You must complete some tasks and make some decisions before you start using CICSVR. CICSVR is part of the recovery strategy for your CICS applications. Recovery consists of several tasks—one of which is the execution of CICSVR—which must all be performed correctly to ensure successful recovery of your VSAM data.

To implement a backup and recovery strategy for your organization, using CICSVR, consider these points:

- Making backups
- Journaling
- Identifying which VSAM spheres you need to protect
- Deciding which CICSVR exits to use
- Creating operational procedures and assigning responsibilities

Establishing backup procedures

Establish procedures that produce regular backup copies of the VSAM spheres you want to protect. You need these backups for forward recovery, but not for backout.

Schedule a regular time to back up your VSAM spheres. If you change the physical characteristics of a VSAM sphere, such as the control interval (CI) size or the length or location of a key field, make a new backup of the VSAM sphere. This ensures that the latest attributes are preserved if you must restore the VSAM sphere.

If your CICS system is operating only for a certain number of hours in each day, make backups of your VSAM spheres outside these hours, when the spheres are unavailable to users. If your CICS system operates continuously, or a VSAM sphere is otherwise in constant use, agree on a time with users when the sphere can be withdrawn from service.

Consider using BWO to have continuous availability of VSAM spheres.

If applications update your VSAM spheres in batch mode (offline to CICS), CICS cannot record the after-images on a forward-recovery log, so take special precautions to ensure protection of these VSAM spheres. Two possible ways to do this are:

1. Change the batch updating program to write CICS-style after-images to a batch journal for use in a forward-recovery run if needed.
2. Back up the sphere before and after batch updates. This protects the sphere by:
   - Providing a backup if the batch program fails. You can restore the VSAM sphere from the first backup and rerun the program after the problem is fixed.
   - Providing a backup for CICSVR to use if the sphere becomes unavailable during later CICS processing.

To take full advantage of CICSVR recovery control, use DFSMShsm to make your backups. CICSVR complete recovery automatically restores the DFSMShsm backup before starting the recovery.

Journaling considerations

You must specify to CICS what kind of journals will hold the after-images and before-images. CICSVR uses the automatic journaling options in the file definition parameters.

For journaling specifically related to CICSVR, you must:

- Specify which journal after-images for forward recovery should be written to, using the FWDRECOVLOG parameter of the file definition.
- Decide if you want to operate with one or two disks or tapes.
- Specify whether you want CICS to operate a cyclic or linear system of tape allocation, if journals are on tape.

For details of the file and journal definitions CICSVR requires, see the *IBM CICS VSAM Recovery MVS/ESA Implementation Guide*.

Archiving your journals

You should use the CICSVR archive utility to archive your journals. The CICSVR archive utility reads a CICS log and stores information in the RCDS. Archive can also copy the CICS log and register this information in the RCDS. The archive utility is described in "Storing information for recovery using the CICSVR archive utility" on page 128, and in the *IBM CICS VSAM Recovery MVS/ESA Implementation Guide*.

Which VSAM spheres to protect

You will probably want to forward recover and back out all VSAM spheres that are defined as CICS files. Of course, data sets that are read-only or browse-only do not need forward recovery or backout, but you should still back them up regularly in case they become unusable.

To use CICSVR to recover your data, you must specify the correct recovery attributes when you define your files to CICS. For details about the recovery attributes required by CICSVR, see the *IBM CICS VSAM Recovery MVS/ESA Implementation Guide*.

CICSVR exit considerations

Four exits are available for use with CICSVR forward recovery or backout:

- The **preapply** exit lets you bypass or modify journal records. It is used for each journal record that will result in an update to the VSAM sphere.
- The **error** exit lets you force CICSVR to continue processing after an I/O error. It is used when an I/O error occurs while CICSVR is reading a journal or accessing a VSAM data set.
- The **ESDS delete** exit lets you logically delete records from an ESDS. If you were adding records to an ESDS during the preceding CICS run, you can use this exit, during CICSVR backout, to delete these records. To logically delete records from an ESDS, you must provide an ESDS delete exit.
- The **termination** exit lets you modify the return code that CICSVR terminates with.

Except for the ESDS delete exit, the use of CICSVR exits is optional.

Two exits are available for use with the CICSVR archive utility:

- The **precopy** exit lets you examine, and act on a copy of, a journal record. You cannot change the original record. It is used for each journal record that is copied to an output data set.
- The **termination** exit lets you terminate operations performed by the precopy exit.

For more information about the CICSVR exits, see the *IBM CICS VSAM Recovery MVS/ESA Implementation Guide*.

Considerations for operational procedures and responsibilities

When you are ready to use CICSVR, define a concise set of operational procedures for use if VSAM data is lost or damaged. Document the procedures required for a CICSVR user to recover from all error situations concerning CICS VSAM data. Here are some considerations that will affect the design of your operational procedures:

Technical responsibilities

- Who is responsible for the different duties associated with CICSVR recovery? These duties include:
  - Monitoring the CICS system for messages indicating a VSAM problem
  - Informing users of the unavailability of a VSAM data set and later availability after successful recovery
  - Authorizing a CICSVR recovery run
  - Locating journals
  - Performing the CICSVR run
  - Examining the CICSVR reports
  - Investigating the reasons for an unsuccessful CICSVR run, if necessary

User considerations

- If you are not going to use CICSVR to provide a recovery capability for all your VSAM spheres, ensure that the instructions to users are clear, when they must reenter data.
- Users might need new instructions to follow if their VSAM data is damaged. In the past, in organizations that had no forward-recovery capability, the usual way to recover from damaged data was for users to repeat the lost work. With CICSVR, the instructions and guidelines for users must be changed, because users need not be involved in the recovery process.
- You might need to send messages from the CICS console operator to users, to tell them that their data is temporarily unavailable during recovery. A message to users after the recovery is complete is also advisable.

Running CICSVR complete recovery

This section describes how to use CICSVR complete recovery. For more information about complete recovery, and for instructions on how to run forward recovery and backout, see the *IBM CICS VSAM Recovery MVS/ESA User's Guide and Reference*.

Complete recovery consists of forward recovery and backout, if required. The panels and secondary windows shown here are in the sequence that they appear for a complete recovery run.

The sequence of events that you perform to use CICSVR complete recovery are:

1. Archive the relevant active CICS log for the VSAM spheres that you want to recover.
2. Select option 1 from the main menu. A list of VSAM spheres appears.
3. Select the VSAM spheres you want to recover.
4. Select the Utilities pull-down from the menu bar, or press the CompRec key (F4).
5. Select option 1 from the pull-down. A secondary window appears.
6. Enter the information that you need in the secondary window, for the VSAM sphere that is displayed. Repeat this step until you have supplied the recovery parameters that you need for this run, for all selected VSAM spheres.

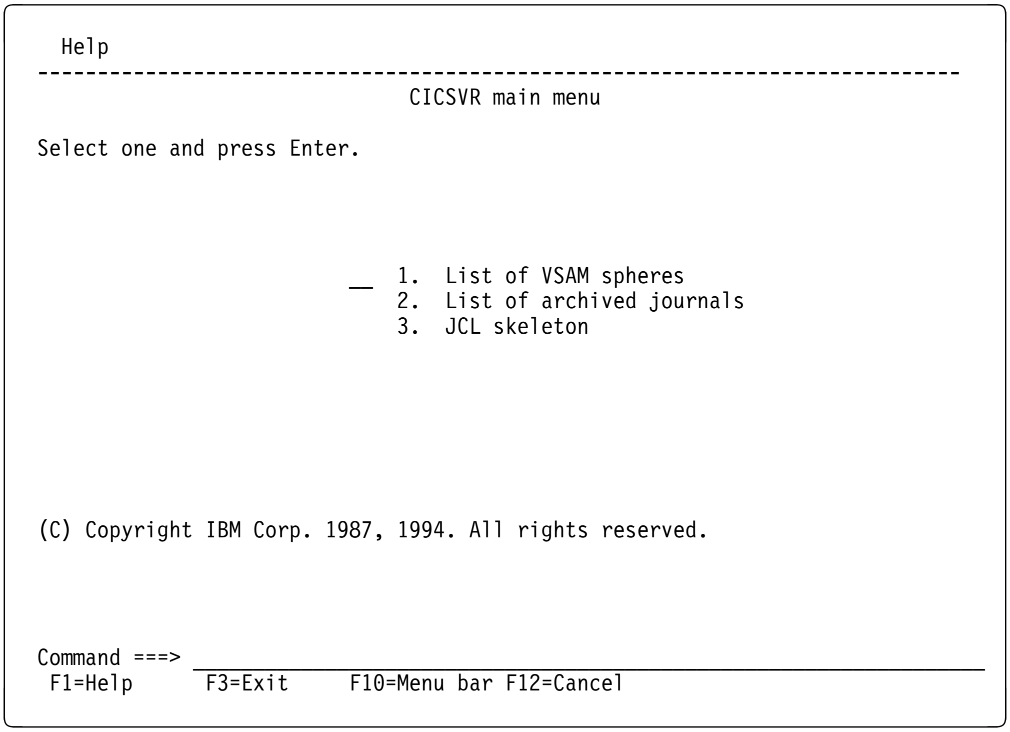
   After the last panel in the sequence appears, you will be asked to wait while the complete recovery job is being constructed.

   If the VSAM spheres you are recovering have been updated by more than one CICS system, the job might consist of multiple steps.
7. Submit the job that CICSVR creates for you.
8. If any of the VSAM spheres that you recovered contained alternate indexes (AIXs) in the upgrade set, use the access method services (AMS) BLDINDX command to recreate the AIXs.

Using the main menu

When you start CICSVR, the main menu appears. To begin complete recovery:

1. Select option 1 and press Enter.



```
  Help
------------------------------------------------------------------------------
                               CICSVR main menu

Select one and press Enter.




                           __  1.  List of VSAM spheres
                               2.  List of archived journals
                               3.  JCL skeleton







(C) Copyright IBM Corp. 1987, 1994. All rights reserved.




Command ===> ____________________________________________________________________
 F1=Help      F3=Exit     F10=Menu bar F12=Cancel
```

Figure 5. Main menu panel

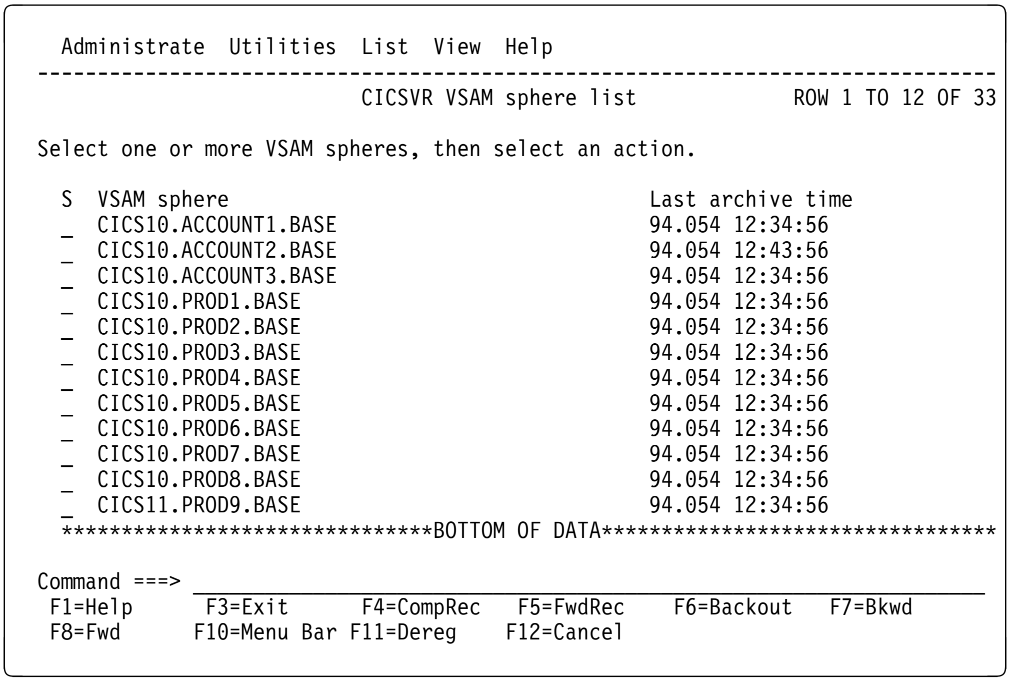
From this panel, you select these objects: a list of VSAM spheres, a list of journal data sets or the JCL skeleton secondary window.

Selecting from the VSAM sphere list

To continue complete recovery:

1. Type S beside the VSAM spheres that you want to recover.
2. Press F10 to get to the menu bar, place the cursor under the Utilities action and press Enter.

When the action is completed, the letter S appears in the select column beside the sphere, showing that this item has been selected. This is the CICSVR VSAM sphere list panel:



```
  Administrate  Utilities  List  View  Help
--------------------------------------------------------------------------------
                          CICSVR VSAM sphere list            ROW 1 TO 12 OF 33

Select one or more VSAM spheres, then select an action.

  S  VSAM sphere                                  Last archive time
  _  CICS10.ACCOUNT1.BASE                         94.054 12:34:56
  _  CICS10.ACCOUNT2.BASE                         94.054 12:43:56
  _  CICS10.ACCOUNT3.BASE                         94.054 12:34:56
  _  CICS10.PROD1.BASE                            94.054 12:34:56
  _  CICS10.PROD2.BASE                            94.054 12:34:56
  _  CICS10.PROD3.BASE                            94.054 12:34:56
  _  CICS10.PROD4.BASE                            94.054 12:34:56
  _  CICS10.PROD5.BASE                            94.054 12:34:56
  _  CICS10.PROD6.BASE                            94.054 12:34:56
  _  CICS10.PROD7.BASE                            94.054 12:34:56
  _  CICS10.PROD8.BASE                            94.054 12:34:56
  _  CICS11.PROD9.BASE                            94.054 12:34:56
  ******************************BOTTOM OF DATA********************************

Command ===> __________________________________________________________________
 F1=Help      F3=Exit      F4=CompRec   F5=FwdRec    F6=Backout   F7=Bkwd
 F8=Fwd      F10=Menu Bar F11=Dereg    F12=Cancel
```

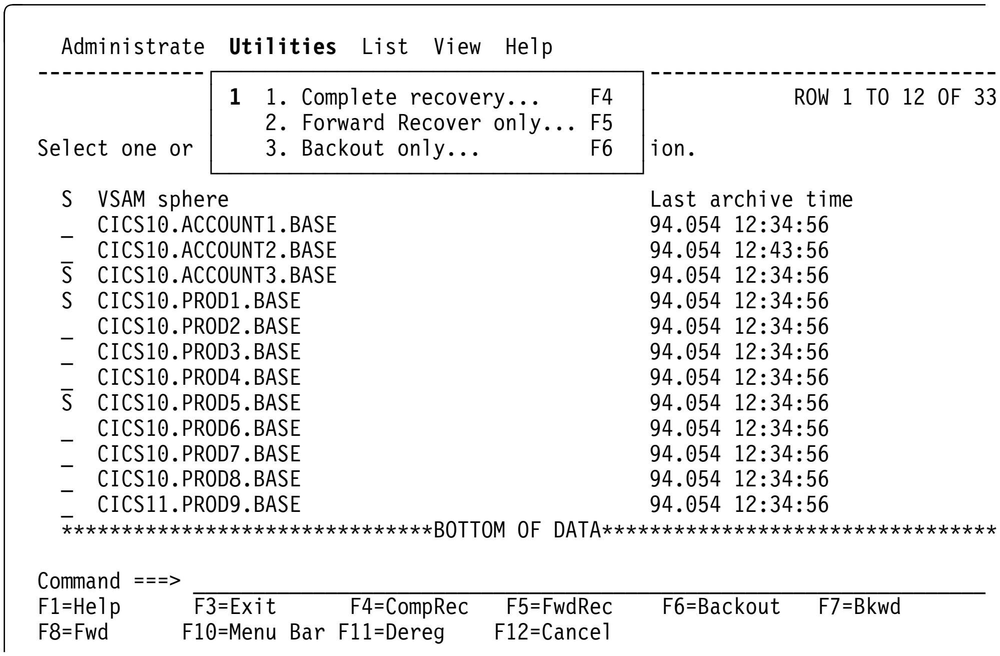
Figure 6. VSAM sphere list panel. Use S in the first column to select a VSAM sphere.

From the CICSVR VSAM sphere list panel, you can select an action by using one of the shortcut function keys or select these pull-downs from the menu bar.

- Administrate
- Utilities
- List
- View
- Help

Using the VSAM sphere list Utilities pull-down

Figure 7 shows a section of the CICSVR VSAM sphere list panel with the Utilities pull-down:



```
  Administrate  Utilities  List  View  Help
------------- ┌────────────────────────────────────┐ -----------------------------
              │ 1  1. Complete recovery...    F4   │            ROW 1 TO 12 OF 33
              │    2. Forward Recover only... F5   │
Select one or │    3. Backout only...         F6   │ ion.
              └────────────────────────────────────┘
  S  VSAM sphere                                     Last archive time
  _  CICS10.ACCOUNT1.BASE                            94.054 12:34:56
  _  CICS10.ACCOUNT2.BASE                            94.054 12:43:56
  S  CICS10.ACCOUNT3.BASE                            94.054 12:34:56
  S  CICS10.PROD1.BASE                               94.054 12:34:56
  _  CICS10.PROD2.BASE                               94.054 12:34:56
  _  CICS10.PROD3.BASE                               94.054 12:34:56
  _  CICS10.PROD4.BASE                               94.054 12:34:56
  S  CICS10.PROD5.BASE                               94.054 12:34:56
  _  CICS10.PROD6.BASE                               94.054 12:34:56
  _  CICS10.PROD7.BASE                               94.054 12:34:56
  _  CICS10.PROD8.BASE                               94.054 12:34:56
  _  CICS11.PROD9.BASE                               94.054 12:34:56
  ******************************BOTTOM OF DATA*********************************

Command ===> __________________________________________________________________
F1=Help      F3=Exit      F4=CompRec   F5=FwdRec    F6=Backout   F7=Bkwd
F8=Fwd      F10=Menu Bar F11=Dereg    F12=Cancel
```

Figure 7. VSAM sphere list—Utilities pull-down

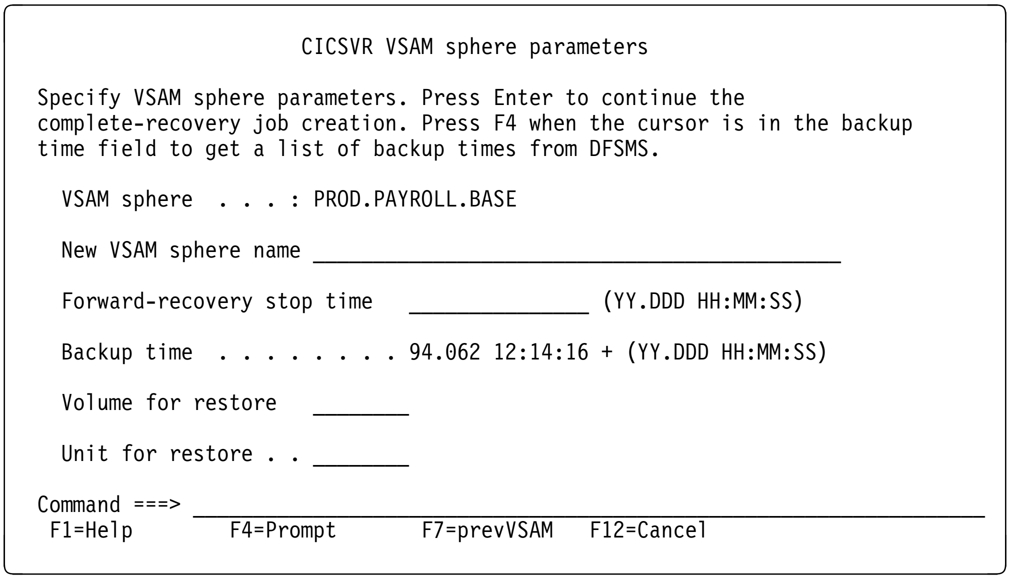
From this pull-down, you can select Complete recovery for the VSAM sphere that you selected by using one of these methods:

- Select option 1
- Press the CompRec key (F4)
- Type comprec on the command line
- Move the cursor to the Complete recovery item in the pull-down, and press enter

To get information about each menu pull-down item, move the cursor to an item and press the Help key (F1).

Providing VSAM sphere complete recovery parameters

When you have selected Complete recovery from the Utilities pull-down, you get this secondary window:



```
                         CICSVR VSAM sphere parameters

 Specify VSAM sphere parameters. Press Enter to continue the
 complete-recovery job creation. Press F4 when the cursor is in the backup
 time field to get a list of backup times from DFSMS.

   VSAM sphere  . . . : PROD.PAYROLL.BASE

   New VSAM sphere name ____________________________________________

   Forward-recovery stop time    _______________ (YY.DDD HH:MM:SS)

   Backup time  . . . . . . . . 94.062 12:14:16 + (YY.DDD HH:MM:SS)

   Volume for restore   ________

   Unit for restore . . ________

 Command ===> __________________________________________________________________
  F1=Help       F4=Prompt       F7=prevVSAM   F12=Cancel
```

Figure 8. VSAM sphere parameters for complete recovery secondary window

Here you can specify VSAM sphere parameters for inclusion in the complete recovery run. These parameters are:

- A new name for the recovered VSAM sphere
- The stop time for forward recovery
- The backup to use
- The volume for the restored copy of the data set
- The unit for the restored copy of the data set

Recovering with DFSMShsm backups: During a complete recovery run, CICSVR restores the VSAM sphere from a DFSMShsm backup, if one exists for the VSAM sphere that you are recovering. The latest DFSMShsm backup will appear as the default in the VSAM sphere parameters secondary window. If you provide a new VSAM sphere name in this secondary window, the recovered data set will have this name.

Recovering with DFSMShsm full volume dumps: If you are using a DFSMShsm full volume dump, these dumps will not appear in the backup list (Figure 12 on page 139). Enter the date of the full volume dump and the start time for the recovery in the backup time field in the VSAM sphere parameters secondary window. If a new VSAM sphere name appears in the secondary window, the recovered data set will have this name.

For more information about using DFSMShsm with CICSVR, see the *IBM CICS VSAM Recovery MVS/ESA Implementation Guide*.

Recovering without DFSMShsm: If you do not use DFSMShsm as a backup utility, restore the backup copy before running complete recovery. If the data set was renamed during the restore process, enter this new name in the VSAM sphere parameters secondary window.

Note: If the data set you are recovering is managed by the storage management subsystem (SMS), the volume and unit values are ignored.

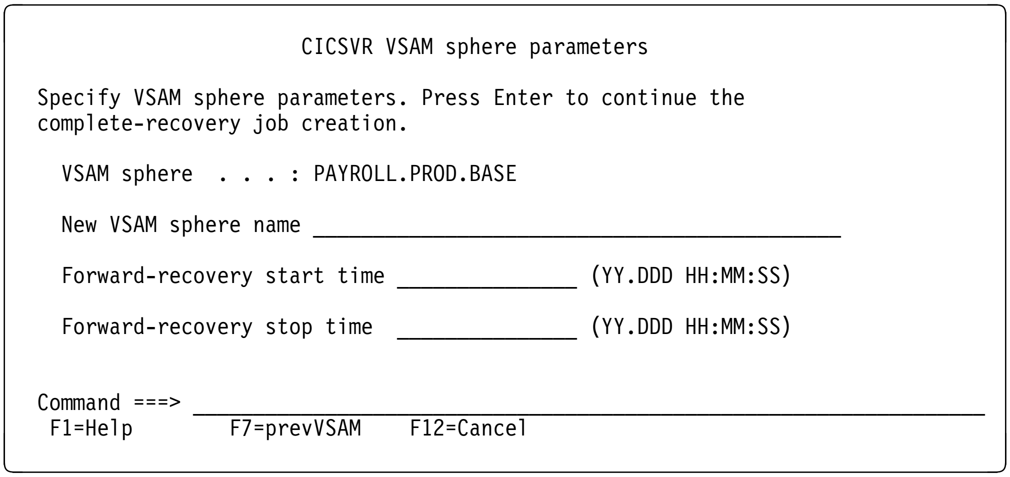
CICSVR can process DFSMShsm backups, full volume dumps, and non-DFSMShsm backups in the same run.

Press the Prompt key (F4), and a list of DFSMShsm backups (Figure 12 on page 139) appears. Press the prevVSAM key (F7), and you are returned to the previous VSAM sphere.

While CICSVR is constructing the complete recovery job, the secondary window shown in Figure 10 on page 138 appears.

For detailed help information about any of these fields, move the cursor to the field and press the Help key (F1).

If you are not using DFSMShsm, this secondary window (Figure 9) appears:



```
                       CICSVR VSAM sphere parameters

Specify VSAM sphere parameters. Press Enter to continue the
complete-recovery job creation.

  VSAM sphere  . . . : PAYROLL.PROD.BASE

  New VSAM sphere name ____________________________________________

  Forward-recovery start time _______________ (YY.DDD HH:MM:SS)

  Forward-recovery stop time  _______________ (YY.DDD HH:MM:SS)


Command ===> __________________________________________________________________
 F1=Help        F7=prevVSAM    F12=Cancel
```

(without DFSMShsm) secondary window

Figure 9. VSAM sphere parameters for complete recovery

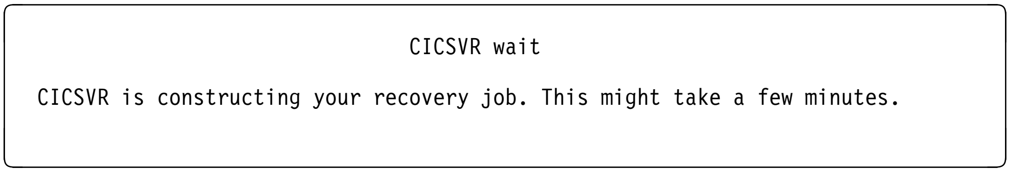
Here you can specify VSAM sphere parameters for inclusion in the complete recovery run. These parameters are:

- A new name for the recovered VSAM sphere
- The start time for forward recovery
- The stop time for forward recovery

Press the prevVSAM key (F7), and you are returned to the previous VSAM sphere.

For detailed help information about any of these fields, move the cursor to the field and press the Help key (F1).

While CICSVR is constructing the complete recovery job, this secondary window (Figure 10) appears:



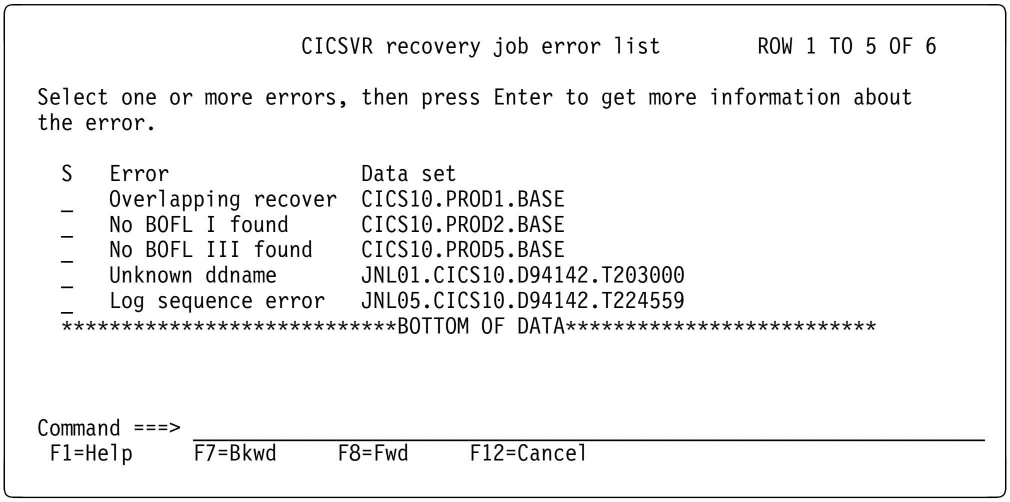
```
                                CICSVR wait

CICSVR is constructing your recovery job. This might take a few minutes.
```

Figure 10. Wait secondary window

If CICSVR detects errors while constructing the complete recovery job, the CICSVR recovery job error list (Figure 11) is displayed.

Listing recovery job errors

Use this secondary window to list errors found during the construction of the complete recovery job.



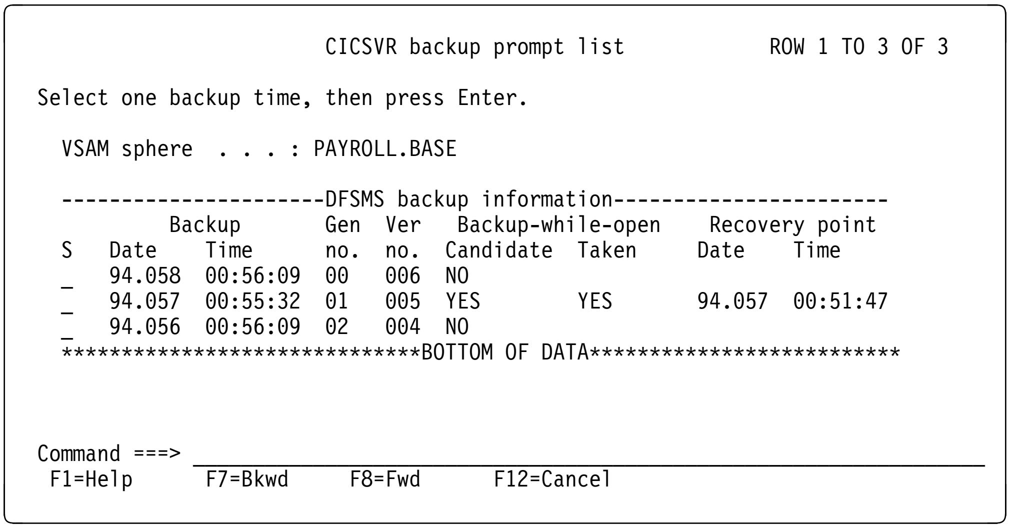
```
                      CICSVR recovery job error list        ROW 1 TO 5 OF 6

Select one or more errors, then press Enter to get more information about
the error.

  S   Error                Data set
  _   Overlapping recover  CICS10.PROD1.BASE
  _   No BOFL I found      CICS10.PROD2.BASE
  _   No BOFL III found    CICS10.PROD5.BASE
  _   Unknown ddname       JNL01.CICS10.D94142.T203000
  _   Log sequence error   JNL05.CICS10.D94142.T224559
  ****************************BOTTOM OF DATA***************************



Command ===> ___________________________________________________________________
 F1=Help     F7=Bkwd     F8=Fwd     F12=Cancel
```

Figure 11. Recovery job error list secondary window. Use S in the first column to select an error.

Select an error to get more specific information. For general help information, press the Help key (F1).

Listing DFSMShsm backups

Figure 12 shows the secondary window that appears if you press the Prompt key (F4) on the CICSVR VSAM sphere parameters panel:



```
                        CICSVR backup prompt list           ROW 1 TO 3 OF 3

Select one backup time, then press Enter.

  VSAM sphere  . . . : PAYROLL.BASE

  ----------------------DFSMS backup information-----------------------
           Backup       Gen  Ver   Backup-while-open    Recovery point
  S   Date    Time      no.  no.  Candidate  Taken     Date    Time
  _   94.058  00:56:09  00   006  NO
  _   94.057  00:55:32  01   005  YES        YES       94.057  00:51:47
  _   94.056  00:56:09  02   004  NO
  ******************************BOTTOM OF DATA***************************



Command ===> ___________________________________________________________________
 F1=Help     F7=Bkwd     F8=Fwd      F12=Cancel
```

Figure 12. Backup prompt list secondary window from Prompt key (F4). Use S in the first column to select a backup.

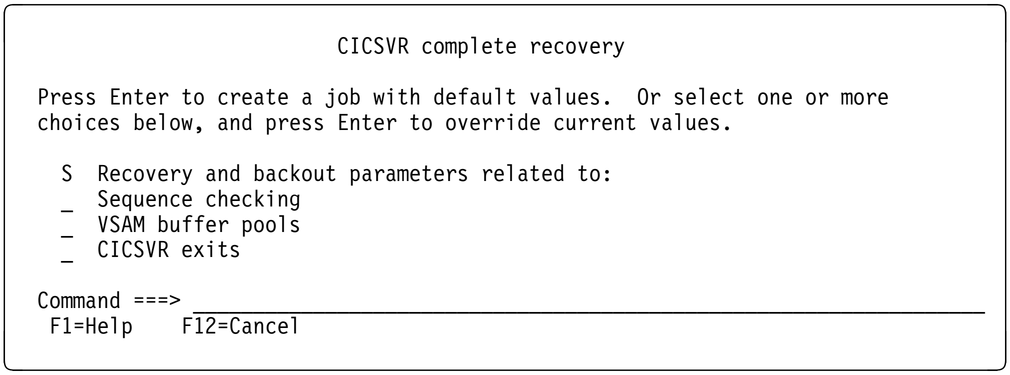
This secondary window appears once for every VSAM sphere that you select, and shows a list of DFSMShsm backups for that sphere. Select the backup that you need from this list.

For general help information, press the Help key (F1).

Entering complete recovery parameters

When you select a CICSVR parameter to change, the parameter secondary windows appear in the order in which they appear in the list.

After you press Enter in the secondary window (Figure 8), you can enter the CICSVR recovery parameters:



```
                         CICSVR complete recovery

Press Enter to create a job with default values.  Or select one or more
choices below, and press Enter to override current values.

  S  Recovery and backout parameters related to:
  _  Sequence checking
  _  VSAM buffer pools
  _  CICSVR exits

Command ===> ___________________________________________________________________
 F1=Help    F12=Cancel
```

Figure 13. Complete recovery parameters secondary window. Use S in the first column to select a choice.

From this secondary window (Figure 13 on page 139), you can create a complete recovery job for the VSAM sphere you selected. You can accept the CICSVR default values, or change the forward-recovery and backout parameters before continuing the recovery.

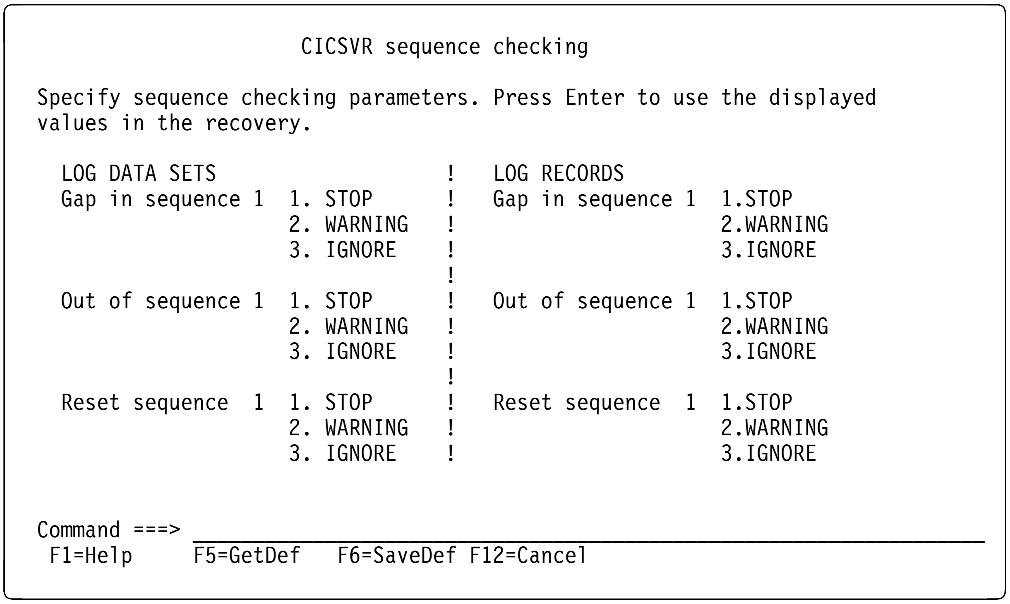
When you select Complete recovery from the Utilities pull-down, you can define these recovery and backout parameters for your run:

- Sequence checking
- VSAM buffer pool
- Exit

For detailed help information about any of these choices, move the cursor to the field and press the Help key (F1).

Sequence checking

Use this secondary window to set sequence checking parameters for your CICSVR run:



```
                      CICSVR sequence checking

Specify sequence checking parameters. Press Enter to use the displayed
values in the recovery.

  LOG DATA SETS                 !   LOG RECORDS
  Gap in sequence 1  1. STOP    !   Gap in sequence 1  1.STOP
                     2. WARNING !                      2.WARNING
                     3. IGNORE  !                      3.IGNORE
                                !
  Out of sequence 1  1. STOP    !   Out of sequence 1  1.STOP
                     2. WARNING !                      2.WARNING
                     3. IGNORE  !                      3.IGNORE
                                !
  Reset sequence  1  1. STOP    !   Reset sequence  1  1.STOP
                     2. WARNING !                      2.WARNING
                     3. IGNORE  !                      3.IGNORE


Command ===> __________________________________________________________________
 F1=Help     F5=GetDef   F6=SaveDef F12=Cancel
```

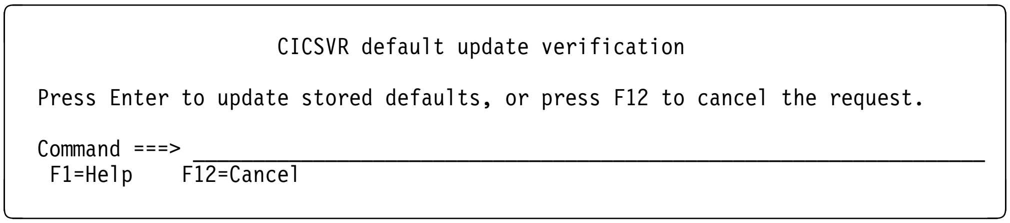
Figure 14. Sequence checking secondary window

When you first enter this secondary window, the CICSVR default values are displayed. Press F5 to get the default values from the RCDS. Press F6 to save the currently displayed values, and the default update verification secondary window (Figure 15 on page 141) appears.

For detailed help information about any of these parameters, move the cursor to the field and press the Help key (F1).

Updating the CICSVR default values

Use this secondary window to confirm an update to the CICSVR command default values:



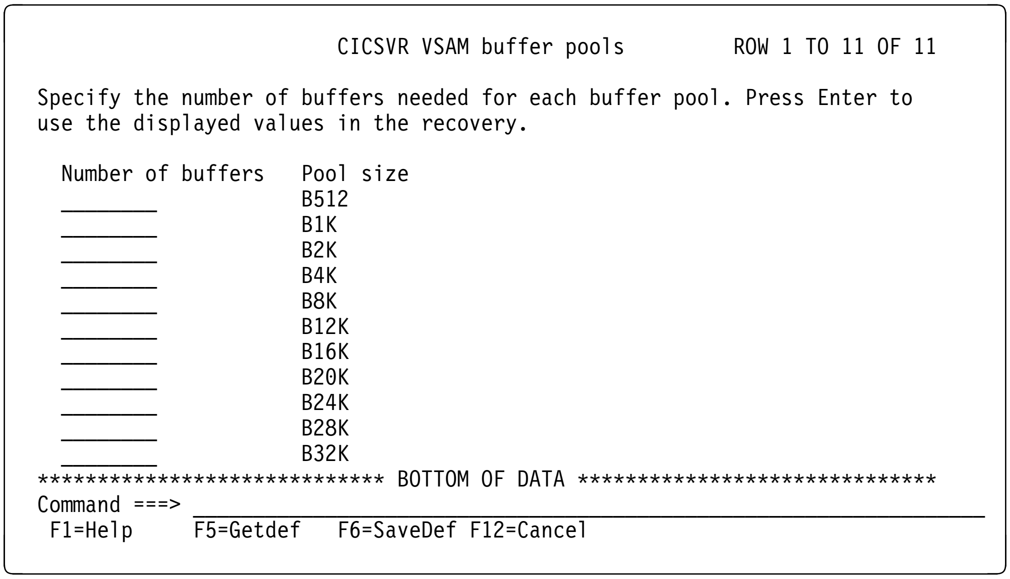
```
                    CICSVR default update verification

 Press Enter to update stored defaults, or press F12 to cancel the request.

 Command ===> ___________________________________________________________________
  F1=Help    F12=Cancel
```

Figure 15. Default update verification secondary window

For help information press the Help key (F1).

Defining the VSAM buffer pools

Use this secondary window to tune your CICSVR run by changing the number of buffers in the VSAM buffer pools:



```
                         CICSVR VSAM buffer pools         ROW 1 TO 11 OF 11

 Specify the number of buffers needed for each buffer pool. Press Enter to
 use the displayed values in the recovery.

   Number of buffers   Pool size
   ________            B512
   ________            B1K
   ________            B2K
   ________            B4K
   ________            B8K
   ________            B12K
   ________            B16K
   ________            B20K
   ________            B24K
   ________            B28K
   ________            B32K
****************************** BOTTOM OF DATA ******************************
 Command ===> ___________________________________________________________________
  F1=Help     F5=Getdef   F6=SaveDef F12=Cancel
```

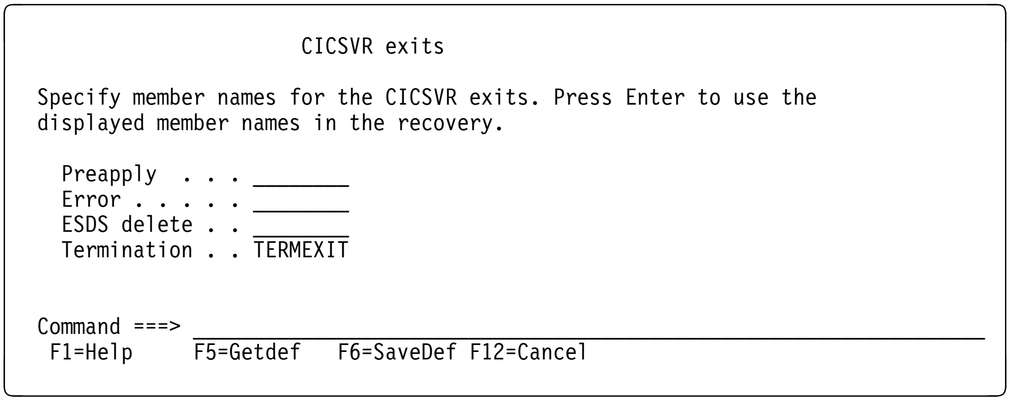
Figure 16. VSAM buffer pools secondary window

When you first enter this secondary window, the CICSVR default values are displayed. Press F5 to get the default values from the RCDS. Press F6 to save the currently displayed values, and the default update verification secondary window (Figure 15) appears.

For detailed help information about any of these fields, move the cursor to the field and press the Help key (F1).

Defining exits

Use this secondary window to define exits for the complete recovery job:



```
                    CICSVR exits

Specify member names for the CICSVR exits. Press Enter to use the
displayed member names in the recovery.

  Preapply  . . . ________
  Error . . . . . ________
  ESDS delete . . ________
  Termination . . TERMEXIT


Command ===> ____________________________________________________________________
 F1=Help     F5=Getdef   F6=SaveDef F12=Cancel
```

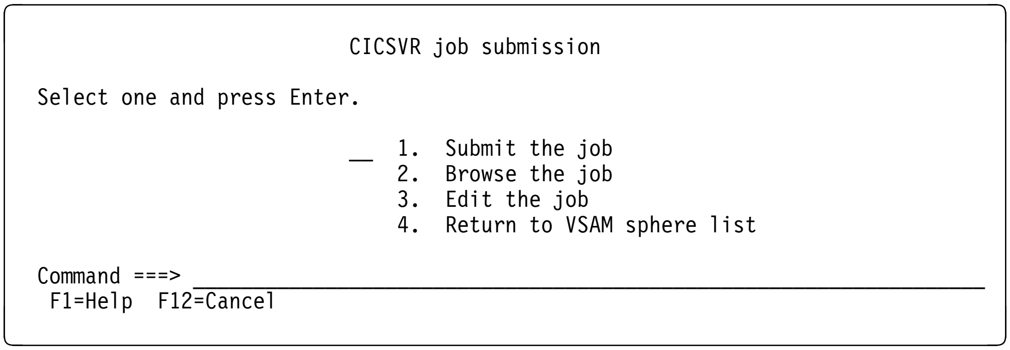
Figure 17. Exits secondary window

When you first enter this secondary window, the CICSVR default values are displayed. Press F5 to get the default values from the RCDS. Press F6 to save the currently displayed values, and the default update verification secondary window (Figure 15 on page 141) appears.

For detailed help information about any of these fields, move the cursor to the field and press the Help key (F1).

Submitting the job

When you have entered the CICSVR parameters that you need, press Enter to get the job submission secondary window (Figure 18). Use this secondary window to browse or edit the job before you submit it, or return to the VSAM sphere list:



```
                        CICSVR job submission

Select one and press Enter.

                        __  1.  Submit the job
                            2.  Browse the job
                            3.  Edit the job
                            4.  Return to VSAM sphere list

Command ===> ____________________________________________________________________
 F1=Help  F12=Cancel
```

Figure 18. Job submission secondary window

For detailed help information about any of these choices, move the cursor to the field and press the Help key (F1).

CICSVR reports

Examples of reports from forward recovery, backout, and archive are provided here. See “Archive reports” on page 147 for examples of the archive reports.

Forward recovery reports

The CICSVR forward recovery and backout functions each produce three statistical reports:

- Journal data set statistics
- Statistics on data sets that have been backed out or recovered
- Exit action statistics

Report of journal data set statistics

```
CICSVR - CICS VSAM RECOVERY                                                DATE : 94/06/07     TIME : 09:05:31     PAGE : 1


JOURNAL DATA SET STATISTICS:
----------------------------

KEY TO FIELD IDENTIFIERS
-----------------------------------------
UPD-AFTER  UPDATE AFTER IMAGE
ADD-AFTER  ADD AFTER IMAGE
DEL-AFTER  DELETE AFTER IMAGE
DSNAME     DDNAME TO SPHERE AND PATH NAME
-----------------------------------------


                                               NO OF RECORDS         NO OF         NO OF         NO OF         NO OF
NAME OF JOURNAL DATA SET                           PROCESSED        DSNAME     UPD-AFTER     ADD-AFTER     DEL-AFTER
--------------------------------------------   -------------   -----------   -----------   -----------   -----------
RETAIL.ACCOUNTS.J02A21.CUST                               11             3             2             5             1
RETAIL.ACCOUNTS.J02A22.CUST                               44            12             5            21             6
--------------------------------------------   -------------   -----------   -----------   -----------   -----------
TOTAL                                                     55            15             7            26             7
--------------------------------------------   -------------   -----------   -----------   -----------   -----------
```

Figure 19. RECOVER—Journal data set statistics

Report of recovered data set statistics

```
CICSVR - CICS VSAM RECOVERY                                                DATE : 94/06/07     TIME : 09:05:31     PAGE : 2


STATISTICS OF RECOVERED DATA SETS
---------------------------------

BASE NAME OF RECOVERED DATA SET: RETAIL.ACCOUNTS.MAIN

BASE NAME OF ORIGINAL DATA SET: RETAIL.ACCOUNTS.MAIN

THE FOLLOWING ASSOCIATED PATHS ARE DEFINED IN THE VSAM CATALOG:
    RETAIL.ACCOUNTS.CUSTNO

FIRST AND LAST RECORDS APPLIED:
                                   DATE    TIME
     RECORDS                       YY/DDD  HH.MM.SS
     ----------------------------  ------  --------
     FIRST JOURNAL RECORD APPLIED  94/154  13:19:59
     LAST JOURNAL RECORD APPLIED   94/155  15:44:59
     ----------------------------  ------  --------


NAME OF RECOVERED DATA SET: RETAIL.ACCOUNTS.MAIN

                    :--------- RECORDS FOUND ON THE LOG(S) ----------:  :----- CHANGE RECORDS APPLIED  -----:  :-- CHANGES
DATASET  FCT ENTRY                                                                                                 IGNORED
TYPE     NAME            DSNAME    UPD-AFTER    ADD-AFTER    DEL-AFTER         ADDS      UPDATES      DELETES      BY EXIT
-------  ---------  -----------  -----------  -----------  -----------  -----------  -----------  -----------  -----------
BASE     MAIN                 4            2           14            3            2            2            1            0
-------  ---------  -----------  -----------  -----------  -----------  -----------  -----------  -----------  -----------
TOTAL                         4            2           14            3            2            2            1            0
-------  ---------  -----------  -----------  -----------  -----------  -----------  -----------  -----------  -----------

------------------  -----------  -----------  -----------  -----------  -----------  -----------  -----------  -----------
OVERALL TOTAL                 4            2           14            3            2            2            1            0
------------------  -----------  -----------  -----------  -----------  -----------  -----------  -----------  -----------

------------------  -----------  -----------  -----------  -----------  -----------  -----------  -----------  -----------
GRAND TOTAL                   4            2           14            3            2            2            1            0
------------------  -----------  -----------  -----------  -----------  -----------  -----------  -----------  -----------
```

Figure 20. RECOVER—Statistics of recovered data sets

Report of exit action statistics

```
CICSVR - CICS VSAM RECOVERY                                                DATE : 94/06/07     TIME : 09:05:31     PAGE : 3


EXIT ACTION STATISTICS
----------------------


                    :----------NUMBER OF ACTIONS TAKEN--------:

EXIT NAME           RECORD CHANGE       CONTINUE         IGNORE
------------------  -------------  -------------  -------------
PREAPPLY                        0              0              0
ESDS DELETE         EXIT NOT TAKEN
------------------  -------------  -------------  -------------


                    :--NUMBER OF ACTIONS TAKEN-:

EXIT NAME                CONTINUE         IGNORE
------------------  -------------  -------------
ERROR               EXIT NOT TAKEN
------------------  -------------  -------------


                    :--NUMBER OF ACTIONS TAKEN-:

EXIT NAME            CODE CHANGED       CONTINUE
------------------  -------------  -------------
TERMINATION         EXIT NOT TAKEN
------------------  -------------  -------------
```

Figure 21. RECOVER—Exit action statistics

Backout reports

Report of journal data set statistics

```
CICSVR - CICS VSAM RECOVERY                                                DATE : 94/06/07     TIME : 09:05:31     PAGE : 1


JOURNAL DATA SET STATISTICS:
----------------------------

KEY TO FIELD IDENTIFIERS
------------------------------------------
UPD-BEFORE  DELETE OR UPDATE BEFORE IMAGE
ADD-BEFORE  ADD BEFORE IMAGE
BOFLGREC    BACKOUT FAILURE RECORD
------------------------------------------


                                               NO OF RECORDS         NO OF         NO OF         NO OF
NAME OF JOURNAL DATA SET                           PROCESSED      BOFLGREC    UPD-BEFORE    ADD-BEFORE
--------------------------------------------   -------------   -----------   -----------   -----------
RETAIL.ACCOUNTS.J02A31.CUST                                4             2             2             0
--------------------------------------------   -------------   -----------   -----------   -----------
TOTAL                                                      4             2             2             0
--------------------------------------------   -------------   -----------   -----------   -----------
```

Figure 22. BACKOUT—Journal data set statistics

Report of backed-out data set statistics

```
CICSVR - CICS VSAM RECOVERY                                                DATE : 94/06/07     TIME : 09:05:31     PAGE : 2


STATISTICS OF BACKED OUT DATA SETS
----------------------------------

BASE NAME OF BACKED OUT DATA SET: RETAIL.ACCOUNTS.MAIN

BASE NAME OF ORIGINAL DATA SET: RETAIL.ACCOUNTS.MAIN

THE FOLLOWING ASSOCIATED PATHS ARE DEFINED IN THE VSAM CATALOG:
    NO PATHS DEFINED

FIRST AND LAST RECORDS APPLIED:
                                   DATE    TIME
     RECORDS                       YY/DDD  HH.MM.SS
     ----------------------------  ------  --------
     FIRST JOURNAL RECORD APPLIED  94/155  11:03:36
     LAST JOURNAL RECORD APPLIED   94/155  10:59:59
     ----------------------------  ------  --------


NAME OF BACKED OUT DATA SET: RETAIL.ACCOUNTS.MAIN

                    :-------- NUMBER OF RECORDS --------:  :---- NUMBER OF CHANGED RECORDS ----:  :-- CHANGES
DATASET  FCT ENTRY                                                                                    IGNORED
TYPE     NAME         BOFLGREC   UPD-BEFORE   ADD-BEFORE         ADDS      UPDATES      DELETES      BY EXIT
-------  ---------  -----------  -----------  -----------  -----------  -----------  -----------  -----------
BASE     CUSTNO               1            2            0            1            1            0            0
-------  ---------  -----------  -----------  -----------  -----------  -----------  -----------  -----------
TOTAL                         1            2            0            1            1            0            0
-------  ---------  -----------  -----------  -----------  -----------  -----------  -----------  -----------

------------------  -----------  -----------  -----------  -----------  -----------  -----------  -----------
OVERALL TOTAL                 1            2            0            1            1            0            0
------------------  -----------  -----------  -----------  -----------  -----------  -----------  -----------

------------------  -----------  -----------  -----------  -----------  -----------  -----------  -----------
GRAND TOTAL                   1            2            0            1            1            0            0
------------------  -----------  -----------  -----------  -----------  -----------  -----------  -----------
```

Figure 23. BACKOUT—Statistics of data sets that have been backed out

Report of exit action statistics

```
CICSVR - CICS VSAM RECOVERY                                                DATE : 94/06/07     TIME : 09:05:31     PAGE : 3


EXIT ACTION STATISTICS
----------------------


                    :----------NUMBER OF ACTIONS TAKEN--------:

EXIT NAME           RECORD CHANGE       CONTINUE         IGNORE
------------------  -------------  -------------  -------------
PREAPPLY                        0              2              0
ESDS DELETE         EXIT NOT TAKEN
------------------  -------------  -------------  -------------


                    :--NUMBER OF ACTIONS TAKEN-:

EXIT NAME                CONTINUE         IGNORE
------------------  -------------  -------------
ERROR               EXIT NOT TAKEN
------------------  -------------  -------------


                    :--NUMBER OF ACTIONS TAKEN-:

EXIT NAME            CODE CHANGED       CONTINUE
------------------  -------------  -------------
TERMINATION         EXIT NOT TAKEN
------------------  -------------  -------------
```

Figure 24. BACKOUT—Exit action statistics

Archive reports

The CICSVR archive utility can produce up to three different reports:

- Automatic deregister statistics
- Archive statistics
- Information for a forward recovery and backout

Report of automatic deregister statistics

```
CICSVR -  ARCHIVE UTILITY                                             DATE : 94/05/27     TIME : 17:50:36     PAGE : 1
AUTOMATIC DEREGISTER STATISTICS
================================
JOURNALS DEREGISTERED FROM RCDS
-------------------------------
CICSID                 : CICS01
MVSID                  : MVS1
JOURNAL ID             : 01
CICS VERSION           : 3
FIRST TIME             : 94.117 13:24:18
LAST TIME              : 94.117 17:36:04
                                                            -- SVC 99 ERROR CODES --    LOCATE  CATALOG
                                                                     INFO    ERROR      MACRO   MACRO
                                              DELE- UNCA-    RETURN  REASON  REASON     RETURN  RETURN
JOURNAL DATA SET NAME                         TED   TALOG    CODE    CODE    CODE       CODE    CODE
--------------------------------------------  ----- -----    ------  ------  ------     ------  -------
CICS01.DFHJ01A.D94117.T132418                  YES   YES      0000    0000    0000       0000    0000
```

Figure 25. ARCHIVE—Automatic deregister statistics

Report of archive statistics

```
CICSVR -  ARCHIVE UTILITY                                             DATE : 94/05/27     TIME : 18:14:10     PAGE : 1

ARCHIVE STATISTICS FOR A VERSION 3 JOURNAL
==========================================

CICSID                 : CICSPROD
MVSID                  : MVS2
JOURNAL ID             : 02
CICS VERSION           : 3
FIRST TIME             : 94.147 14:15:30
LAST TIME              : 94.147 17:23:28
FIRST SEQUENCE NUMBER  :        737
LAST SEQUENCE NUMBER   :        994
INPUT JOURNAL NAME     : CICSPROD.DFHJ02A
OUTPUT JOURNAL NAME(S) : CICSPROD.JNL02.D94147.T141530

CICSVR -  ARCHIVE UTILITY                                             DATE : 94/05/27     TIME : 18:14:10     PAGE : 2

VSAM DATA SET STATISTICS
========================

                                              DATE/TIME OF     DATE/TIME OF
VSAM DATA SET NAME                            FIRST REFERENCE  LAST REFERENCE
--------------------------------------------  ---------------  ---------------
CICS1.PROD1.BASE                              94.147 14:19:23  94.147 16:47:18
CICS1.PROD2.BASE                              94.147 14:21:37  94.147 17:23:27
CICS1.PROD3.BASE                              94.147 14:44:02  94.147 16:23:05

CICSVR -  ARCHIVE UTILITY                                             DATE : 94/05/27     TIME : 18:14:10     PAGE : 3

TIE-UP STATISTICS
=================

FCT       DATE/TIME OF     FCT
NAME      TIE-UP RECORD    FORMAT  BASE DATA SET NAME                            PATH DATA SET NAME
--------  ---------------  ------  --------------------------------------------  ------------------------------------------
BASE1A    94.147 14:19:23   FIX    CICS1.PROD1.BASE                              CICS1.PROD1.PATHA
BASE2     94.147 14:21:37   FIX    CICS1.PROD2.BASE                              CICS1.PROD2.PATH
BASE3     94.147 14:44:02   VAR    CICS1.PROD3.BASE                              CICS1.PROD3.PATH
BASE1B    94.147 15:07:14   FIX    CICS1.PROD1.BASE                              CICS1.PROD1.PATHB

CICSVR -  ARCHIVE UTILITY                                             DATE : 94/05/27     TIME : 18:14:10     PAGE : 4

REDO FILE ID WITHOUT TIE-UPS
============================

FCT       DATE/TIME OF     DATE/TIME OF
NAME      FIRST REFERENCE  LAST REFERENCE
--------  ---------------  ---------------
BASE1C    94.147 14:17:33  94.147 14:38:21

CICSVR -  ARCHIVE UTILITY                                             DATE : 94/05/27     TIME : 18:14:10     PAGE : 5

BACKOUT FAILING LOG RECORDS FOUND
=================================

                                                                 RECORD
                                              DATE/TIME OF       TYPES
BASE DATA SET NAME                            LAST BOFLGREC      MISSING
--------------------------------------------  ----------------  -------
CICS3.PROD2.BASE                              94.147 17:23:27      3
CICS3.PROD3.BASE                              94.147 15:04:37     NONE
```

Figure 26. ARCHIVE—Archive statistics for a Version 3 journal

Report of information for a forward recovery and backout

```
CICSVR -  ARCHIVE UTILITY                                             DATE : 94/05/27     TIME : 18:20:10     PAGE : 1

INFORMATION FOR A FORWARD RECOVERY OF PAYROLL.TEST
==================================================

JOB STEP 1

START TIME       STOP TIME
---------------  ---------------
94.147 10:15:30  94.147 15:01:12

JOURNALS NEEDED
---------------
CICSPROD.JNL03.D94147.T101530
CICSPROD.JNL03.D94147.T121530
CICSPROD.JNL03.D94147.T141530

CICSVR -  ARCHIVE UTILITY                                             DATE : 94/05/27     TIME : 18:20:10     PAGE : 1

INFORMATION FOR BACKOUT OF PAYROLL.PROD
=======================================

JOB STEP 1

START TIME       STOP TIME
---------------  ---------------
N/A              94.147 15:20:24

JOURNALS NEEDED
---------------
CICSPROD.JNL01.D94147.T141530
```

Figure 27. ARCHIVE—Information needed for a forward recovery and backout

Chapter 13. User exits for transaction backout during emergency restart

This chapter describes the opportunities for including your own logic in 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 39. Transient data and temporary storage backout do not have any exits.

This chapter contains Product-sensitive Programming Interface information. For additional programming information, see the *CICS/ESA Customization Guide*.

Where you can add your own code

At emergency restart, you may add your own code in postinitialization programs that you nominate in the program list table (described on page 91).

You may want to include functions in 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).

Note: After a local DL/I backout error, if you are using DBRC, CICS informs DBRC of the error.

The transaction backout programs have four exits:

1. XRCINIT—initialization and termination exit
2. XRCINPT—input exit
3. XRCFCER—file error exit (only for DFHFCBP)
4. XRCOPER—open error exit (only for DFHFCBP).

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 with TBEXITS=(name1,name2,name3,name4), where name1, name2, name3, and name4 are the names of your programs for XRCINIT, XRCINPT, XRCFCER, and XRCOPER.

Figure 28 on page 152 shows which programs the user exits are invoked in, and the order in which they are invoked.

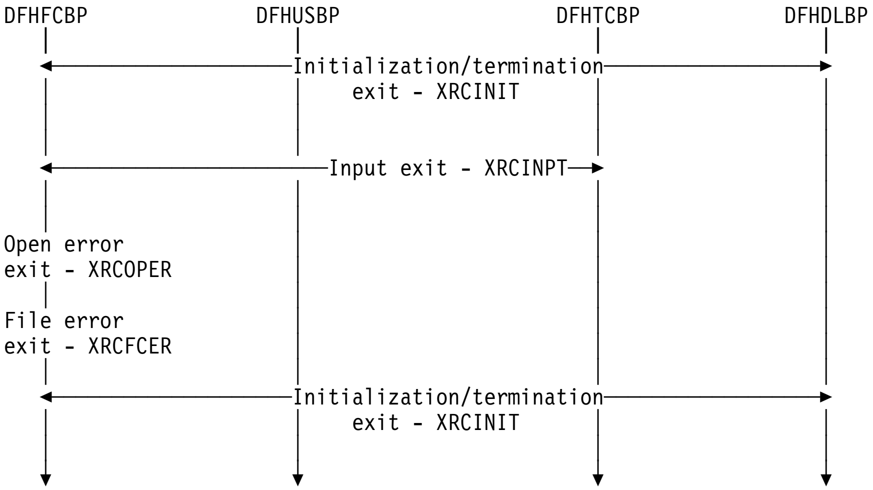

DFHFCBP
DFHUSBP
DFHTCBP
DFHDLBP
Initialization/termination exit - XRCINIT
Input exit - XRCINPT
Open error exit - XRCOPER
File error exit - XRCFCER
Initialization/termination exit - XRCINIT

Figure 28. User exits for backout at recovery

For programming information on the identity of the invoking program and, for the initialization/termination exit, the time of invocation, as indicated to the exit programs by parameters, see the *CICS/ESA Customization Guide*.

Exit details

For programming information on the generalized interface for exits, the writing of exit programs, and details of the input parameters and the return codes for each exit, see the *CICS/ESA Customization Guide*.

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; and
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 file backout table (FBO), the message backout table (MBO), and the DL/I backout table (DBO). 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 **local 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 IMS-DL/I backout processing is stopped in this way, CICS safeguards DL/I 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 40.

Note that 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 BDAM 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.

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 such as CICSVR. For more information about flagging backout errors, see Chapter 18, “Backout failure” on page 189.

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.

Return code **UERCBYP** indicates that the error is ignored and backout continues. The data set is not flagged as “backout failed”.

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.

The 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 the operator chooses CANCEL, 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 14. 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 way these programs work, and some design considerations for them, are described in Chapter 6, “Communication error processing” on page 55.

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

Communication design

Communication design is discussed under the following headings:

- “Communications-related programming considerations”
- “Journaling of messages” on page 158
- “Handling communication breaks” on page 158.

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 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 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. (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 a 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/ESA Application Programming Reference* manual.
3. Specify maximum protection (for VTAM messages) by PROTECT(YES) in the CEDA DEFINE TRANSACTION command. 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 87).

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

CICS supplies sample communication error programs: a node error program (NEP) for VTAM terminals, and a terminal error program (TEP) for non-VTAM terminals. For programming information on the sample NEP and TEP, see the *CICS/ESA Customization Guide*. The following text describes the main reasons why you might want to tailor the supplied NEP or TEP. 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.

Possible reasons for editing a NEP or TEP

Reasons for wanting to edit a NEP or a TEP include the following:

- 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 55 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/ESA 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)—TCAM 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 55 introduces the TEP, and “Handling communication breaks” on page 158 offers some design ideas.

The TEP can be:

- The CICS sample TEP
- Your own TEP.

The CICS sample TEP

The CICS sample TEP is supplied in the pregenerated system. 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 15. Recovery coding in application programs

This chapter describes the following aspects of recovery coding in application programs:

- “Application design”
- “Program design” on page 163
- “Coping with transaction failures” on page 168
- “Enqueuing in application programs” on page 173.

Before reading on, note these terms:

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.)

You will need 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 165).

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/ESA 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 164).

  We recommend short LUWs 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 60), 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 162). 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, 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 165) 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 165), 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. You must 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 167.

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)
- Transient data queues
- User files and DL/I and DB2 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): 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 (DATAID or QUEUE name) 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 and DB2 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, DL/I, and DB2 databases can be forward recovered.

Designing to avoid transaction deadlock

To avoid transaction deadlock (see “Possibility of transaction deadlock” on page 178), 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:
  1. Enqueue by the same character string
  2. 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:

  1. 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.)

  2. 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.

  3. The application program issues a 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 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 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.

You should also think about 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. You should also ensure that either the user or master terminal operator can take appropriate action to repeat the updates. You could, for example, allow the user to reinitiate the task.

An alternative solution is for the started transaction to issue a START command specifying its **own** TRANSID. Immediately before issuing the 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 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 85.)

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/ESA 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 will be 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 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 here under the following headings:

- Transaction failures
- System failures.

The actions taken by CICS are described under Chapter 5, “Abend processing” on page 47 and “Processing of operating-system abends and program checks” on page 53.

Transaction failures

When a transaction fails, the following CICS facilities can be invoked during and after the abend process:

- CICS condition handling
- HANDLE ABEND commands, and user exit code
- The 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, a 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 HANDLE CONDITION commands. Review these for each new application.

HANDLE ABEND commands

As described in “How CICS handles transaction abends” on page 47, 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 73.

Your installation may have standards relating to the use of 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 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, you should 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/ESA 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 50.)

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 93.

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 93.

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 49). 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 95.

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 181. (DFHPEP is invoked during abnormal task termination as described at “Abnormal termination of a task” on page 49.)

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 BDAM or VSAM-ESDS files. (see Chapter 13, “User exits for transaction backout during emergency restart” on page 151 for further information).
  - File errors during transaction backout.
  - Journal records transferred from the system log to the restart data set 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 47 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:

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 ASRASPC	The type of space in control at the time of the last ASRA, ASRB, AICA, or AEYD abend, if any
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 (we recommend, 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.
- You may here 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 a 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 description of program level abend processing at “How CICS handles transaction abends” on page 47.)

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 *CICS/ESA Messages and Codes* manual 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/ESA Application Programming Reference* manual. For background information, see the *CICS/ESA Application Programming Guide*.

Processing the IOERR condition

Any program that attempts to process an IOERR condition for a recoverable resource must not issue a RETURN or SYNCPOINT command, but must be terminated by issuing an 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 will 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 85).

   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 and 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.

PL/I programs and error handling

On-units are a standard method of error-handling in PL/I programs. If the execution-time option STAE is specified, CICS program control services set up an exit routine that activates the PL/I on-units.

This exit routine can handle:

- All PL/I errors
- CICS abends that occur in the PL/I program and in associated CICS services
- Program checks.

Note that, under CICS, PL/I execution-time options can be specified only by the PLIXOPT character string.

For details of PL/I coding restrictions in a CICS environment, see the appropriate PL/I programmer’s guide for your compiler.

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 178.)

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 163.

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 BDAM files that are nonrecoverable (that is, LOG=YES is not specified in the FCT entry), CICS itself provides no exclusive control over records that are being updated. You may specify the use of BDAM exclusive control, in which case CICS will specify exclusive control on a READ UPDATE BDAM request, and release control either on the associated WRITE, or by issuing a RELEX macro in response to RELEASE command, or at syncpoint.

For nonrecoverable VSAM files, VSAM locks the control interval during an update.

Figure 29 illustrates the extent of exclusive control for nonrecoverable files.

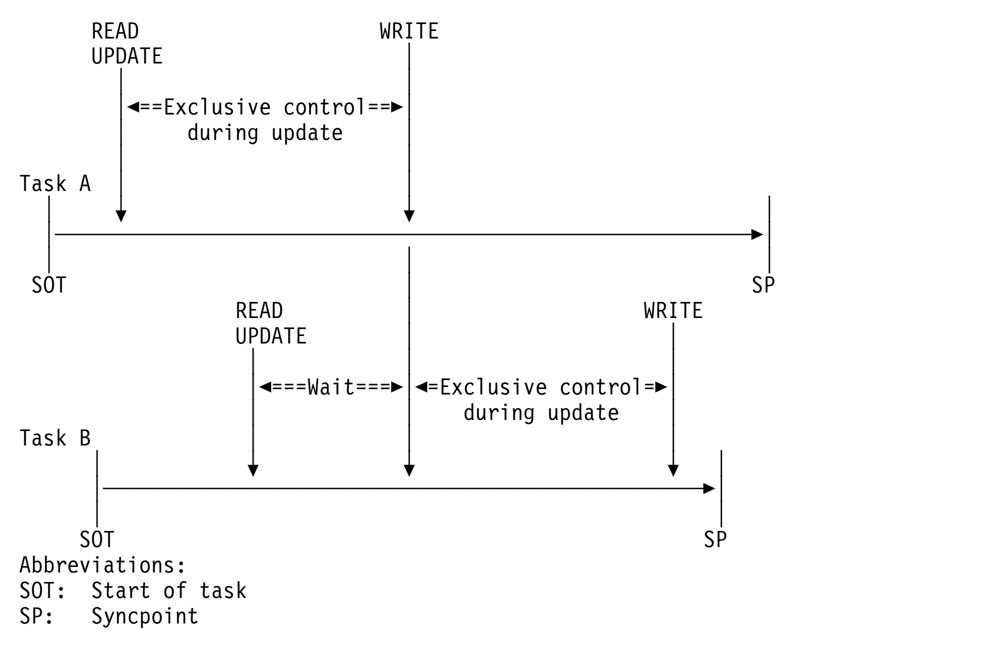


Figure 29. Exclusive control during updates to nonrecoverable files. This figure illustrates two tasks updating the same record or control interval. Task A is given exclusive control of the record or control interval between the READ UPDATE and WRITE commands. During this period, task B waits.

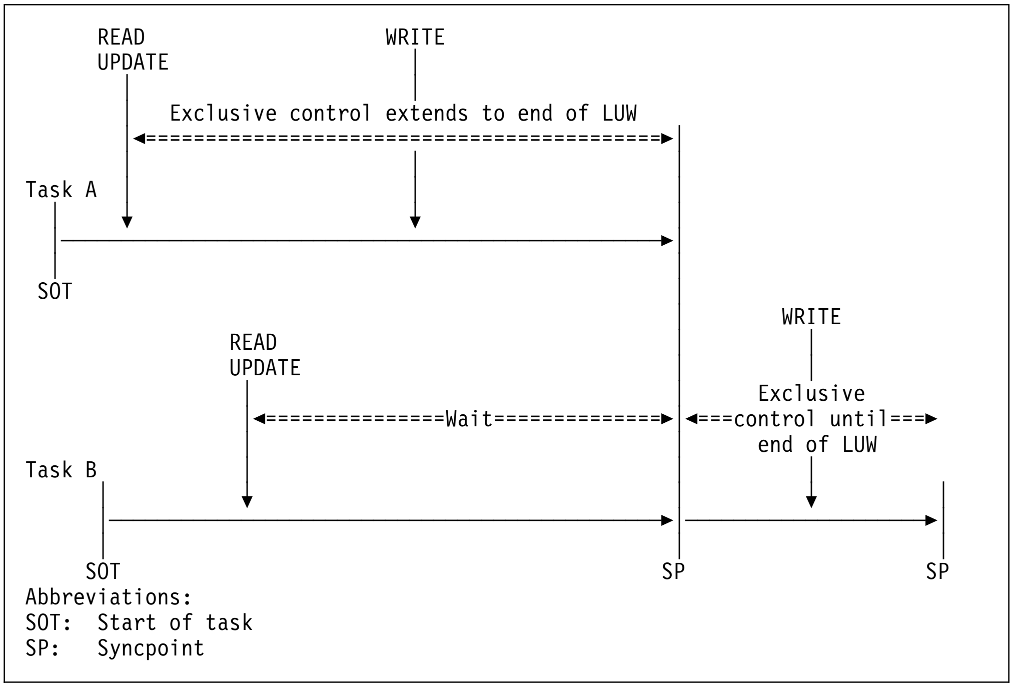


Figure 30. Enqueuing (exclusive control) during updates to recoverable files. This figure illustrates two tasks updating the same record or control interval. Task A is given exclusive control of the record until the update is committed (at the end of the LUW). During this period, task B waits.

Recoverable files

For VSAM or BDAM files designated as recoverable, CICS extends the duration of its enqueuing action as shown in Figure 30. 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 29 on page 174 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 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 178).
2. The spheres of CICS exclusive control are the physical block for BDAM 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 a READ UPDATE command (for any file, recoverable or not), VSAM maintains its exclusive control of the control interval containing the record until a 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:

- WRITEQ TD
- READQ TD
- DELETEQ TD

Thus, for example:

- If a task issues a 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 a 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 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 queue(s) with the same temporary storage identifier. (See "Explicit enqueuing (by the application programmer)" on page 178.)

Temporary storage control commands that invoke implicit enqueuing are:

- WRITEQ TS
- DELETEQ TS

Implicit enqueuing on local DL/I databases

There are two distinct cases—program isolation scheduling and intent scheduling. Each is discussed separately in the sections that follow. You can also use the internal resource lock manager (IRLM) facility provided by DB2 and IMS. For a summary of the facilities provided by IRLM, see Chapter 22, "Recovery and restart in an IMS data-sharing environment" on page 209.

| Note, however, that under IMS/ESA Version 4.1, you must use program isolation
| scheduling.

Program isolation scheduling

When a task accesses a segment by a DL/I 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 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 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.

Note: IMS/ESA Version 4.1 does not support intent scheduling.

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 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 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.**[7]

[7] Transaction deadlock is sometimes known as **enqueue deadlock**, **enqueue interlock**, or **deadly embrace**.

As shown in Figure 31, 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.)

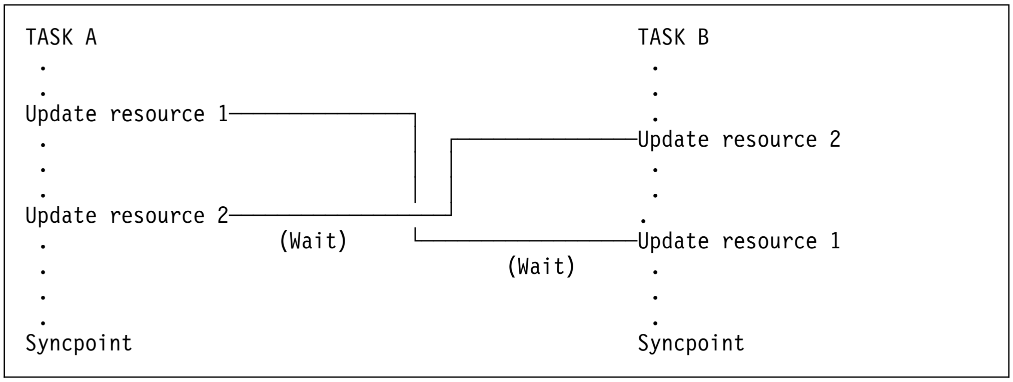


Figure 31. Transaction deadlock (generalized)

If transaction deadlock occurs, one task abends and the other proceeds.

- If both deadlocked resources are CICS resources, or one is CICS and the other 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 one of them is cancelled by a master terminal command.
- If the resources are both DL/I databases and:
  - No DTIMOUT period is specified for either task and
  - Program isolation scheduling is not being used

  both tasks remain suspended indefinitely, unless one is cancelled by a master terminal command.

  However, if program isolation scheduling is being used, DL/I itself detects the potential deadlock as a result of the tasks issuing their scheduling calls. In this case, DL/I causes CICS to abend the task that has the least update activity (abend code ADLD for CICS local DL/I, and ADCD for DBCTL).

The abended task may then be backed out by dynamic transaction backout, as described in “Dynamic transaction backout (DTB)” on page 50. (Under certain conditions, the transaction can be automatically restarted, as described under “Abnormal termination of a task” on page 49. Alternatively, the terminal operator may restart the abended transaction.)

For more information, see “Designing to avoid transaction deadlock” on page 166.

Chapter 16. 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 53.

This chapter contains Product-sensitive Programming Interface information. For programming information to complement the information in this book, see the *CICS/ESA Customization Guide*, which contains detailed advice on writing these error programs.

Program error program (DFHPEP)

As described on page 53, 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 DFHMISC CICS-supplied group, 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. You may add code to take actions you choose as 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/ESA 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 17. 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 87 for details of this option.

Logic of inquiry program

The inquiry program should inspect the message cache[8] 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/ESA Application Programming Reference* manual.)
- If the terminal **did** 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 184).

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 terminal operator has the necessary authority (based on CICS transaction attach security or operator class in the CICS segment of the user's profile in the RACF database). Processing could then take place as if the transaction request had just been entered.

Notes:

1. To identify the type of message in a message cache, see "Message cache records" on page 187.
2. Assuming that the message cache temporary storage queues are recoverable, there may be messages for more than one task in the cache. We recommend deleting a message cache immediately after use. (You can do this with a DELETEQ TS command in the inquiry program.)

[8] During emergency restart, logged messages are copied from the restart data set into message caches, as described on page 40.

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/ESA 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 the RACF CICS segment and CICS transaction attach security, see the *CICS/ESA CICS-RACF 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 11 on page 185 shows the main actions performed by CICS during the execution of a single-LUW message-protected transaction.
- Table 12 on page 185 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.

You can see in these figures, 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 43) 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 11. CICS actions during execution of a single-LUW message-protected task. 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.

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 12. Contents of message cache after emergency restart for a single-LUW message-protected task. The step numbers in the first column of this figure refer to the step numbers in the previous figure.

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 184.
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 in the text.
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/ESA Customization Guide*.

Chapter 18. 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 explicitly to 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 68.)

A backout utility such as CICSVR 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 transaction. 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 the utility.

CICSVR automatically keeps track of the archived logs needed for a recovery run, and constructs the recovery jobs for you using an ISPF dialog interface (see Chapter 12, “Forward recovery and backout with CICSVR” on page 125).

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/ESA CICS-Supplied Transactions* manual).

Chapter 19. 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 60.

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 60), 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, you should consider using BWO to take backup copies of the data sets (see Chapter 11, “Backup while open (BWO)” on page 97). Alternatively, you will 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.

You will need to check the above timetable again when more detailed design work has been done.

Offline recovery

VSAM data sets and DL/I 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 18, “Backout failure” on page 189 and, for DL/I dynamic allocation support, see “Dynamic allocation and deallocation of local DL/I databases” on page 197. Operators should be well-practiced in offline procedures so that recovery is not delayed.

Part 4. Recovery in a DL/I environment

This part:

1. Gives an overview of how to implement local DL/I recovery
2. Describes local DL/I recovery processes
3. Tells you what DBRC and data sharing are, and helps you decide whether you want to consult the IMS library for further information
4. Highlights points that are relevant to CICS as an IMS subsystem.

This part assumes that you are familiar with associated IMS documentation. The IMS library describes DBRC and data-sharing in detail. You should also consult your IBM representative on the availability of pertinent IBM System Center documents.

This part has three chapters:

Chapter 20, “Implementing local DL/I recovery and error processing” on page 195 gives a brief overview of how to implement local DL/I recovery, and then describes local DL/I recovery processes.

Chapter 21, “Using IMS DBRC for recovery control” on page 203 summarizes the recovery facilities of DBRC and their use with CICS.

Chapter 22, “Recovery and restart in an IMS data-sharing environment” on page 209 introduces data sharing, and summarizes the recovery implications of running CICS as a data-sharing subsystem.

This part describes local DL/I support, but does **not** discuss CICS working with DBCTL (for information about this, see the *CICS/ESA CICS-IMS Database Control Guide*).

Chapter 20. Implementing local DL/I recovery and error processing

This chapter describes how to implement local DL/I recovery and some of the processes that handle the relationship between CICS and local DL/I. It assumes that you are familiar with the relevant IMS publications, listed in “Books from related libraries” on page xiii.

The information is divided as follows:

- “Backward recovery of local DL/I databases” on page 196
  - “Backward recovery of local DL/I databases” on page 196
  - “Forward recovery of DL/I databases” on page 196
  - “Dynamic allocation and deallocation of local DL/I databases” on page 197
  - “Program isolation or intent scheduling for local DL/I” on page 197.
- “Local DL/I recovery processes” on page 198
  - “IMS pseudo-abends causing transaction failure” on page 198
  - “DL/I abends causing CICS failure” on page 198
  - “DL/I backout failure” on page 198
  - “Write I/O error on a database” on page 199.

Design factors

A design factor relating to recovery and restart of a DL/I database 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. In addition, program isolation scheduling is the only method supported by IMS/ESA Version 4.1.

Assuming that program isolation scheduling is to be used:

- There is a possibility that any transaction accessing DL/I resources could end with a program isolation deadlock. You should therefore make all such transactions capable of dynamic transaction backout. You can also make them restart 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 (described at “Abnormal termination of a task” on page 49) are not violated.

- A request from a program to terminate a DL/I PSB implies a CICS syncpoint, which commits both DL/I **and** non-DL/I changes.

- Transactions that update both DL/I and non-DL/I resources should always access the resources in the same sequence. In this way you avoid the possibility of a transaction deadlock between DL/I and non-DL/I resources.

If batch programs and online programs are to use a DL/I database concurrently (by means of the CICS shared database or IMS data-sharing facility), make checkpoint requests at appropriate intervals (seconds, rather than minutes). Frequent checkpoints:

- Minimize the risk that the DL/I enqueue pool will fill and cause failure
- Help avoid unnecessary delays in response to users.

Batch programs must be able to start from a checkpoint (see the IMS manuals for information on writing restartable programs.)

Implementing recoverability of local DL/I databases

DL/I running locally writes both before- and after-images of changed segments to the CICS system log, providing records to support both backward and forward recovery. For this reason, good operational control of the system log files is vital. Loss or destruction of the system log could jeopardize database integrity.

Backward recovery of local DL/I databases

During a task abend or an emergency restart, **dynamic transaction backout** (DTB) backs out, or undoes, changes made to local DL/I databases.

If a failure occurs while backing out local DL/I database changes, CICS stops all databases that have been accessed by the LUW, and abends all tasks currently scheduled to any of the stopped databases. This enables offline forward recovery (if necessary) and backout (see “DL/I backout failure” on page 198).

Forward recovery of DL/I databases

IMS provides utilities for forward recovery of DL/I databases. During implementation, it is necessary to establish procedures for the integration of the system log produced during CICS execution with those logs produced by batch job execution. You need these procedures to ensure a coherent and complete set of forward recovery information.

Notes:

1. You do not need the CICS system log for the forward recovery of remote DL/I or DBCTL databases.
2. If you implement the CICS system log on **disk**, archive each system log data set before overwriting it; otherwise you will lose recovery information collected on the system log. See “System log” on page 67 for information about using the AUTOARCH or PAUSE option to prevent loss of information.
3. If you implement the CICS system log on **tape**, do not reuse tapes until their recovery information is no longer needed. (Correct use of standard-labeled tapes will help.)
4. To run the IMS forward recovery utility for local DL/I databases successfully, CICS must close the log and, if you are using DBRC, inform DBRC of the log closure. For **disk** only, to cater for an abnormal CICS termination that does not

close the log, you can use the START=LOGTERM system initialization parameter as an alternative to an emergency restart.

Either way, you will close the log and inform DBRC but, whereas an emergency restart backs out the resources updated by in-flight tasks, the LOGTERM system initialization parameter causes CICS to stop the subsequent emergency restart before backout processing begins, so you can then recover offline. If you have specified automatic archiving, an archive job will be submitted.

Dynamic allocation and deallocation of local DL/I databases

You can forward recover a DL/I database, or make backup copies, with the database offline. IMS provides dynamic allocation and deallocation to protect the integrity of your databases. If no DD statement has been provided for the database data sets contained in the database, allocation happens automatically when you schedule the database.

At system termination, or when you issue a CEMT SET DLIDATABASE(databasename) STOPPED|RECOVERDB command, the database is automatically deallocated whether there is a DD statement for the database or not. After a CEMT SET DLIDATABASE STARTED, the first reference to the database dynamically allocates it. For more information about CEMT, see the *CICS/ESA CICS-Supplied Transactions* manual.

When you have finished using the database offline, you can bring it back online and make it accessible to CICS transactions again.

CICS provides a dynamic allocation and deallocation sample program that can be used with DL/I databases. For programming information about the dynamic allocation and deallocation sample program, see the *CICS/ESA Customization Guide*.

Program isolation or intent scheduling for local DL/I

You specify program isolation or intent scheduling by the PISCHD=YES|NO system initialization parameter. However, if you are using IMS/ESA 4.1, intent scheduling is not supported, so PISCHD=YES is mandatory and need not be specified.

DL/I provides the facility to detect impending deadlock between DL/I requests. A request that would result in a deadlock causes one of the tasks to abend.

Transactions subject to program isolation deadlock can, under certain conditions, be restarted (see “Abnormal termination of a task” on page 49).

For a discussion of program isolation and intent scheduling, see page 195.

Local DL/I recovery processes

DL/I warm start

For existing databases, you should be aware of the relationship between:

1. The physical database data sets.
2. The DD names.
3. The DDIR (DMB directory).
4. The DMB (data management block) for that database.
5. The extended error queue elements (EEQEs) held for that database on the global catalog. (There is more information about EEQEs in “DL/I I/O error handling” on page 200.)

You should not change any of these elements unless you are discarding or recovering the database, because CICS needs this information to maintain data integrity. This includes integrity after write I/O errors. If any of points 1 through 4 change, the EEQEs will not correspond. CICS is not able to detect this, except when the DDIR has been deleted. In this case, the operator is prompted to delete or retain the EEQEs.

If there is no DDIR entry, the status information for that data set is discarded. Normally, the status of the following parameters of the CEMT SET DLIDATABASE command is maintained from the previous run of CICS:

- STARTED
- STOPPED
- RECOVERDB
- DUMPDB
- ACCESS value (RO, RD, UP, or EX).

IMS pseudo-abends causing transaction failure

Error conditions within DL/I (IMS pseudo-abends) may be of a type that can be transformed into a transaction abend and so do not prevent CICS from continuing. An example of this type of error is no space in the database for an insertion (see Chapter 5, “Abend processing” on page 47).

DL/I abends causing CICS failure

In situations where the DL/I-detected error is sufficiently serious, the CICS system abends to avoid compromising database integrity and to allow you to diagnose the error and recover the database.

This type of error causes a **user** abend of CICS (see “Processing of operating-system abends and program checks” on page 53).

DL/I backout failure

If a failure occurs during the backout of a local DL/I database, CICS safeguards data integrity by stopping all potentially affected databases.

If you are using DBRC, CICS informs it that a database has stopped, and prevents further use of that database until after recovery.

The stopped DL/I databases must then be taken offline so that forward recovery (if necessary) and batch backout can take place; for more information on forward recovery and batch backout, see the *CICS/ESA Operations and Utilities Guide* and the appropriate IMS manuals. You can take databases offline for recovery without bringing down CICS. See also “Dynamic allocation and deallocation of local DL/I databases” on page 197.

DL/I backout failure during dynamic transaction backout (DTB)

To safeguard the data integrity of local DL/I databases in the event of a DL/I backout failure during DTB, CICS does the following:

1. Stops **all** databases updated by the LUW that is being backed out. Stopping databases means flagging them so that future tasks cannot schedule PSBs that refer to any of those databases.
2. Abends all tasks currently scheduled to any of the stopped databases.
3. Continues to back out non-DL/I recoverable resources updated by the LUW.

DL/I backout failure during emergency restart

To safeguard the data integrity of local DL/I databases if there is a DL/I backout failure during CICS emergency restart, CICS takes actions that depend on the type of DL/I backout failure:

1. For DL/I backout failures caused by:

   - The inability to open a database
   - An I/O error on a database
   - A VSAM update error on a database

   the CICS default action is to give the operator the option to cancel CICS or to allow CICS to continue. Assuming the system console operator allows CICS to continue (by entering GO), CICS does the following:

   a. For the particular task whose DL/I backout processing has failed, CICS identifies the PSB that was in use, 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.

   b. CICS continues emergency restart processing for all other resources.

2. For other types of DL/I backout failure, or when the operator chooses to cancel CICS, CICS terminates (with an operating-system-requested shutdown).

Write I/O error on a database

This section gives some CICS-specific information. For additional information, see the *IMS/ESA Diagnosis Guide and Reference* manual for the release of IMS that you are using.

IMS I/O error handling offers advantages in terms of:

- The integrity of your data
- XRF users who cannot afford to delay a takeover to recover databases
- Non-XRF users who require the emergency restart to be as rapid as possible.

An IMS database does not have to be recovered before restarting CICS, therefore the restart is not delayed. The database may be deallocated after the restart and then recovered.

I/O error handling allows applications on an IMS subsystem such as CICS to continue processing a database after write I/O errors have occurred on it. For the life of the CICS job that had the I/O error, IMS keeps a copy of the block or control interval. The copy is then used whenever the block or control interval is referenced.

When a write I/O error occurs during CICS execution:

- CICS applications can read to and write from the affected block or control interval as though the error had not occurred.
- CICS and IMS records all the information required to physically recover the database offline on the CICS log.

When CICS is shut down, the buffer is lost. Note that in both IMS DB/DC and DBCTL, those methods recover the error data buffer across restarts.

When the database is stopped, IMS tries to correct the error on the database, using the information in the buffer. The CICS global catalog is informed that the buffer is in error. So, at the subsequent restart (cold, warm, or emergency), the error in the buffer is noted. The next section explains this process.

DL/I I/O error handling

For write I/O errors that occur on a database, CICS provides I/O error handling across takeovers for XRF systems and all restarts for non-XRF systems. This speeds up the restart process—particularly important for XRF systems. CICS records data that describes the error on the CICS global catalog. This data is called an extended error queue element (EEQE). CICS also records EEQEs for read I/O errors.

At a takeover or restart, CICS reads the EEQEs from the global catalog. EEQEs are reestablished before any DL/I backout processing, and whether DBRC is installed or not. Any attempt to access database segments protected by EEQEs receives a status code of AO. This ensures database integrity.

Deletion of EEQEs: CICS deletes EEQEs from the global catalog, using database data from the disk, only in the following circumstances, depending on whether DBRC is installed.

Databases registered with DBRC: DBRC is the central repository for EEQEs. When running with DBRC, CICS keeps EEQEs on its global catalog only for emergency restart. CICS holds EEQEs until the database has been recovered offline and the DBRC EEQE information deleted. DBRC is informed of EEQEs when write errors occur. DBRC deletes EEQEs after forward recovery.

When a database registered with DBRC is opened, the CICS list of EEQEs on the global catalog is amended to match the DBRC list. Any CICS EEQE that does not have a matching DBRC EEQE is deleted, because it is no longer required.

Databases without DBRC: All EEQEs are deleted when you issue the master terminal command CEMT SET DLIDATABASE RECOVERDB.

Do not issue this command unless you are sure of your intentions and you are going to forward recover the database that had the I/O error. CICS cannot check the validity of this command.

When the database is closed, IMS attempts to rewrite any error blocks, in case the I/O error was temporary. If it was a temporary error, the corresponding EEQEs are deleted.

Implementing I/O error handling: The DLIOLIM system initialization parameter allows you to control the number of read and write I/O errors you will accept on each database before CICS tries to change its status to STOPPED. Any new schedule requests for the database will fail, but a task that has the database scheduled may continue until it issues a DLI TERM call or a CICS syncpoint, or until completion.

You will need to increase the size of the global catalog to allow CICS to record EEQEs. The maximum number of bytes of extra space you will need is 150 multiplied by the value of DLIOLIM multiplied by the number of databases. See the *CICS/ESA System Definition Guide* for more information about allocating the global catalog.

Erroneous backout failure: There is one situation where you can get a backout failure on a DL/I database after a restart when the database is in fact in good condition. This situation occurs only if you restore a database between CICS abnormal termination and its following emergency restart, and an in-flight task existed that had updated a block or control interval that had an I/O error. If this occurs, forward recovery (if necessary) and batch backout must be performed, as described on page 198. Note that this does not involve a loss of integrity. This will occur when:

1. A write I/O error occurs on a DL/I database, and I/O error handling allows CICS applications to continue updating the database using the extended storage buffers. CICS writes EEQEs to record the error.
2. A task has updated one of the blocks or control intervals that are in error-handling mode, and CICS fails while the task is still in flight.
3. You recover the database offline before restarting CICS.

In this situation:

- CICS is not aware that the database has been recovered.
- DBRC and CICS EEQEs are not merged until emergency restart is completed, and CICS EEQEs are therefore left intact.
- DL/I backout tries to back out the in-flight task.
- Because the EEQE for the block or control interval that the in-flight task had updated is still intact, a backout failure occurs and the database is put into a status of STOPPED.
- At this stage, a batch backout will restore the databases.

Chapter 21. Using IMS DBRC for recovery control

This chapter describes how you can use the database recovery control (DBRC) feature of IMS to help you recover DL/I databases when they are owned exclusively by a single CICS system (that is, when IMS data sharing is not being used).

For detailed information about DBRC, see the following books:

- *IMS/ESA Operations Guide*
- *IMS Operator's Reference Manual*
- *IMS Utilities Reference Manual* for DBRC command syntax and execution of the DBRC utility (DSPURX00).

Introduction to database recovery control (DBRC)

IMS database recovery control (DBRC) automates many of the steps in forward recovery of DL/I databases. It can generate job streams to run the various IMS utility programs needed for forward recovery.

The major benefit of DBRC is that it reduces the risk of database corruption caused by human error. In doing this, it also eases the system programmer and operator work load.

You can use DBRC for database recovery control without using data sharing. But if you do use it with data sharing, it will also control access to databases shared by different subsystems.

Subsystems

A system that uses IMS databases under control of DBRC is called a **subsystem**. A subsystem can be one of the following:

- A CICS system with IMS DB
- An IMS DB batch region
- An IMS DC (online) system
- An IMS recovery-related utility program
- An IMS/ESA DBCTL system.

IMS database recovery control

IMS database recovery control (DBRC) is a feature of IMS, and runs in the CICS address space as an MVS subtask. (In the data-sharing environment, each subsystem that is participating has its own copy of DBRC.)

DBRC provides its recovery control facilities by recording information about DL/I databases, the subsystems that are accessing them, and any related system logs in three related VSAM key-sequenced data sets that are together referred to as the RECON (REcovery CONtrol) data set. A CICS subsystem automatically calls DBRC to update the information in the RECON data set.

Submitting DBRC commands

You communicate with DBRC by using DBRC commands as input to the DBRC recovery control utility, DSPURX00. You can also submit the DBRC commands, GENJCL, INIT, CHANGE, NOTIFY, DELETE, and LIST online using the CBRC CICS transaction for local DL/I databases (for more information about CBRC, see the *CICS/ESA CICS-Supplied Transactions* manual).

The syntax and use of the DBRC commands and utilities are described in the appropriate IMS manuals, as mentioned at the beginning of this chapter.

Utilities

DBRC makes extensive use of the following IMS utilities:

- DFSUDMP0—Image copy
- DFSUCUM0—Change accumulation
- DFSURDB0—DB recovery
- DFSBBO00—Batch backout
- DFSULTR0—Log recovery
- DFSUARC0—Archiving.

Note: DFSULTR0 and DFSUARC0 are part of the logging feature of IMS, and are not part of the base product.

The *IMS Utilities Reference Manual* for your version of IMS describes the use of these utilities.

The DSPURX00 (the DBRC recovery control utility) is provided by DBRC itself. It supports commands that create, maintain, and update the RECON data set, and which generate JCL for the data set log utility and for most of the IMS recovery-related utilities (but not batch backout or log recovery).

Planning for recovery control

This section summarizes the operational procedures that create a DBRC recovery control environment.

Initializing your RECON data sets

Before DBRC can use your RECON data sets, you must initialize them with the INIT.RECON command. The parameters of the INIT.RECON command, RECOVCTL and SHARECTL, specify recovery control only, or data sharing control, respectively.

Even if you are not using data sharing, you gain added protection for your data by specifying SHARECTL. If you register your database with a share level of 0 (see the next section) and initialize the RECON with SHARECTL, DBRC prevents concurrent access by more than one subsystem (this stops accidental loss of data integrity). If you initialize the RECON with RECOVCTL, you get no protection from concurrent access, regardless of the share level of the database.

Registering your database with DBRC

If you plan to use DBRC to control a database, either for full data sharing or just for recovery, you must register the database with the INIT.DB command, and each database data set with the INIT.DBDS command.

Taking image copies of your database

From time to time, you must take an image copy of your database as a basis for forward recovery if it becomes necessary. You can use GENJCL to generate this job.

Archiving system log data sets

To use DBRC with local DL/I, your CICS system log must be on disk or on standard-labeled tape. Before you reuse your disk log data sets, you must take archive copies. If you are using tape logging, the log tapes themselves can serve as archive volumes. For cyclic journals, you must not overwrite volumes that are still potentially required for recovery.

If you are using disk logging, we advise you to define your log with two data sets (JTYPE=DISK2) and with JOUROPT=AUTOARCH (see the *CICS/ESA Resource Definition Guide*). This automates the archiving of a full data set immediately after a log switch occurs. As alternatives, you may use the JOUROPT=PAUSE option, or write your own versions of the DFHXJCO and DFHXJCC replaceable modules to provide your own procedures for archiving logs. For programming information on the use of these modules, see the *CICS/ESA Customization Guide*.

You must take an archive copy of the CICS emergency restart log (DFHJ01X) after emergency restarts.

In IMS terms, a CICS disk log is a **system log data set** (SLDS), and an archived CICS log data set is a **recovery log data set** (RLDS).

Running change accumulation

Change accumulation merges log data sets and reduces the information they contain to the minimum required to recover a particular database data set or group of database data sets.

For change accumulation purposes, you place each database data set in a group of one or more data sets, called a change accumulation group (CAGRP), which you define to DBRC with the INIT.CAGRP command. Each run of the change accumulation utility is for a specified CAGRP. The change accumulation utility places its merged output data in a sequential data set called a change accumulation data set. You can use GENJCL to generate this job.

Database recovery

DBRC provides three types of database recovery: full forward recovery of a database to its correct current state, time-stamp recovery, and track recovery. For details of the last, consult the appropriate manual for your version and release of IMS.

Before you can forward recover, you must take your database offline (deallocate it). IMS/ESA provides dynamic allocation and deallocation support (see “Dynamic allocation and deallocation of local DL/I databases” on page 197). For programming information about the sample dynamic allocation program, see the *CICS/ESA Customization Guide*. If you are not using IMS dynamic allocation, you must allocate and deallocate each database data set individually.

When you have deallocated all the data sets in the database, you can run a separate recovery job for each data set, using the IMS database recovery utility, DFSURDB0.

Batch backout operations

In general, DBRC does not affect CICS backward recovery (backout) operations. Following an error, CICS backs out changes to a local DL/I database during transaction backout or emergency restart. If the online backout operation fails, CICS closes the database, and you must run a batch job to perform the backout. In a data sharing environment, CICS also informs DBRC, which flags the database record in the RECON data set as “needing backout”. DBRC instructs the participating subsystems to close the database, and the flagged database record in the RECON data set ensures that no system can use the database until backout-failure recovery has been performed.

You use the IMS DFSBBO00 batch backout utility to do batch backout. You must run batch backout for every failing program scheduling block (PSB). For programming information on DFSBBO00, see the *IMS Utilities Reference Manual*. Note that you cannot use the DBRC GENJCL command to generate the JCL for this utility. You must ensure that the JCL you supply to the batch backout utility is appropriate for processing the CICS log (see the appropriate *IMS Utilities Reference Manual*).

CICS system programming requirements

The principal additional system programming tasks caused by the use of DBRC are those already described in this chapter—planning and implementing (or supervising) the various DBRC jobs.

Installation

You must install IMS, including DBRC, before you generate local DL/I support in the CICS system. DBRC is an optional part of the base IMS product, and does not require separate installation (see the appropriate *IMS Installation Guide*).

Install CICS, and generate local DL/I support exactly as you would if DBRC were not being used.

The startup job stream must select a system initialization table (SIT) that has DLDBRC=YES specified. You **cannot** specify this as an override parameter.

In the CICS startup job stream, you can include DD statements for the RECON datasets, and for JCLOUT, JCLPDS, and IMS. For example:

```
//RECON1   DD DSN=IMSESA.RECON1,DISP=SHR
//RECON2   DD DSN=IMSESA.RECON2,DISP=SHR
//RECON3   DD DSN=IMSESA.RECON3,DISP=SHR
//JCLPDS   DD DSN=IMSESA.PROCLIB,DISP=SHR
//JCLOUT   DD SYSOUT=(A,INTRDR)
//IMS      DD DSN=IMSESA.DBDLIB,DISP=SHR
```

However, using the RECON DD statements prevents the dynamic allocation of the RECON data sets by MVS. Dynamic allocation is preferable, because it ensures that CICS automatically uses the correct and current RECON data sets.

You must include the JCLOUT, JCLPDS, and IMS DD statements if you intend to use GENJCL to issue DBRC commands online.

Resource definition

Here is a summary of the resource definition requirements for DBRC:

System initialization parameters
You must specify DLDBRC=YES and DLI=YES for local DL/I support. DLI=YES can be specified as a system initialization parameter.

Program entries
For local DL/I support, you should install the CICS-supplied DFHDLI group.

DFHJCT
In the entry for the system log, you must specify that it is on disk or standard-labeled tape. To facilitate log archiving (see “Archiving system log data sets” on page 205), we advise you to code JOUROPT=AUTOARCH and JTYPE=DISK2 (that is, the log is on disk).

DFHDLDBD
For each database registered with DBRC, like all other local DL/I databases, you must define a DL/I DMB directory entry with a DFHDLDBD TYPE=ENTRY macro in the DDIR.

Chapter 22. Recovery and restart in an IMS data-sharing environment

This chapter describes how data sharing is implemented, how to recover from failures while data sharing, and how to program for data sharing.

Before you read this chapter, read Chapter 21, "Using IMS DBRC for recovery control" on page 203 for an introduction to DBRC.

IMS data sharing and CICS

IMS data sharing allows more than one system (subsystem) to have concurrent access to a DL/I database. A subsystem can be a CICS or IMS DC online region, an IMS DB batch region, or an IMS/ESA DBCTL region. The subsystems can comprise any mixture of the possible types—for example, all the subsystems can be CICS systems.

All the subsystems can reside in a single host processor, or, depending on the level of sharing, in up to four processors. For interhost block level sharing, there may be two host processors.

Why use data sharing

CICS **function shipping** (in an MRO or ISC environment) allows a CICS system to access a database owned by another CICS system, by transmitting a specific request. A **CICS shared database** batch region can access a database owned by the CICS online region, but only under CICS control.

IMS data sharing, in which CICS regions can participate, also allows more than one system to access the same database. In an IMS data-sharing environment, each subsystem accesses the database **directly**, and no one subsystem owns the database. This eliminates the overhead of communication between CICS regions, at the cost of using the database recovery control (DBRC) and the internal resource lock manager (IRLM) provided by IMS. Furthermore, you obtain all the recovery facilities of DBRC.

DBCTL is the best way to share DL/I databases, but if you do not have DBCTL, data sharing offers improvements over the following methods:

- DL/I function shipping between CICS regions.

  If you have two or more CICS regions that require frequent access to the same DL/I databases, data sharing has performance advantages. On the other hand, if your CICS regions require only infrequent access to DL/I databases owned by other CICS regions, data sharing can cause a performance degradation compared with function shipping.

  The potential for improvement is greatest in installations where function-shipped DL/I requests form a large proportion of the total DL/I database requests issued.

- CICS shared database support via interregion communication.

  Data sharing between an IMS batch region and a CICS system allows you to use the full functions of IMS batch programs; the restrictions associated with CICS shared database support do not apply. These restrictions are documented in the *IMS/ESA Application Programming: DL/I Calls* and the *IMS/ESA Application Programming: EXEC DL/I Commands* manuals.

Data sharing provides certain facilities not otherwise available:

- It enables CICS to share access to a database with an IMS DC region, rather than by passing data over intersystem sessions using intersystem communication.
- It allows the sharing of access to a database across host processors between multiple CICS and IMS systems.

Levels of sharing

The SHARELVL operand of the INIT.DB command defines the permitted level of sharing for each database. ("Access authorization" on page 212 describes how DBRC controls the actual access to the data by each subsystem.)

There are four levels of data sharing:

EXCLUSIVE LEVEL (SHARELVL=0)
Only one subsystem at a time can access the database. Use this level when you do **not** wish to share access to a database.

DATABASE LEVEL (SHARELVL=1)
Multiple subsystems can have read access.

One subsystem can update the database.

There is no theoretical limit to the number of host processors. The practical limit is four due to I/O contention on DASD.

INTRAHOST BLOCK LEVEL (SHARELVL=2)
Multiple subsystems can have access (update or read).

All subsystems in one host.

INTERHOST BLOCK LEVEL (SHARELVL=3)
As SHARELVL=2, except that the subsystems can be distributed between two host processors.

Table 13 on page 211 shows the main features of each level of sharing.

Table 13. Levels of sharing

SHARELVL=	Description	Comments
0	Exclusive use of database by one subsystem at a time. No sharing is allowed.	DBRC is required.
1	Sharing at the database level. Multiple subsystems can have read access. Optionally, one subsystem can have update access. If an updater is present, the other systems must have read-only access. This means that the data integrity of records that are read is not guaranteed.	DBRC is required. Subsystems can run in several separate host processors. Both batch and online regions can share.
2 and 3	Sharing at the block level. Multiple subsystems can have read or update access. Sharing is at the physical block level for ISAM or OSAM databases and at the control interval level for VSAM databases.	DBRC and IRLM are required. For level 2 sharing, the subsystems must be within the same physical processor. For level 3 sharing, the subsystems can be distributed between two physically adjacent processors. Both batch and online regions can share.

How is data sharing implemented?

Data sharing is implemented through DBRC running in each subsystem region, and the internal resource lock manager (IRLM) running in its own region in each host processor. DBRC is a prerequisite to any level of data sharing, and IRLM is needed for block level data sharing. DBRC controls backup and recovery as in a single system with DBRC, and also controls access for all DL/I databases registered to it.

Internal resource lock manager (IRLM)

DBRC can control a complete data-sharing environment when all databases are shared at the exclusive or database level. When you use block level sharing (sharelevel 2 or 3), you also need the internal resource lock manager (IRLM) to control the locks on individual blocks of a shared database data set.

IRLM is a standalone resource-locking facility that runs in its own region in each host in an IMS data-sharing configuration. In interhost data sharing, the individual IRLMs communicate with each other to control locks across the host systems.

Registering databases

You must **register** all databases that you want to participate in data sharing (see “Registering your database with DBRC” on page 205).

Subsystem sign-on

Each subsystem must **sign on** to enable DBRC to record the name of each active subsystem. In the case of a CICS subsystem, signing on is automatic at system initialization time if the DLDBRC=YES system initialization parameter is specified. The name of the CICS subsystem is the value of the APPLID parameter.

Access authorization

When a subsystem first attempts to schedule a given database, DBRC must **authorize** its access before it can successfully schedule the database. The access intent is specified in the ACCESS operand of the DFHDLDBD macro defining the database to CICS. The possible values of the ACCESS operand, in ascending order of restrictiveness, are:

- RO (read-only, that is, read without integrity)
- RD (read, that is, read with integrity)
- UP (update)
- EX (exclusive).

The following table shows the equivalence in meaning of these values and the PROCOPT value of the PCB used by a batch DL/I job:

CICS Access Intent	**PROCOPT of PCB Used by a Batch DL/I Job**
Read-only (RO)	GO, GOT, or GON
Read (RD)	G
Update (UP)	I, R, D, or A
Exclusive (EX)	L or E

DBRC grants authorization if it does not detect any conflict. DBRC denies authorization if the database is already authorized to another subsystem at a conflicting level, or if an error condition exists. The *IMS Messages and Codes Reference Manual* describes the possible error conditions (in the discussion of the reason codes associated with message DFS047A).

Subsystem termination and access deauthorization

If a CICS (or any) subsystem terminates normally, DBRC deauthorizes the databases to which it had access. If a subsystem terminates abnormally, DBRC records the subsystem failure.

If CICS fails after updating databases, DBRC retains any authorizations held on its behalf, and IRLM retains the locks on any databases and physical blocks that CICS had control of at the time of failure. This allows CICS, on emergency restart, to back out the effects of uncompleted transactions and restore the integrity of the data.

Naming multiple CICS systems

Your operating procedures should ensure that it is not possible to start up two CICS subsystems with the same APPLID. If this happens, two subsystems are sharing the same SUBSYSTEM record in the RECON data set, and data integrity is lost.

Recovery in an IMS data-sharing environment

This section describes how to recover from failures related to the two control components of IMS data sharing, and how to do forward and backward recovery of your databases in an IMS data sharing environment. The four main headings are:

- "DBRC failures"
- "IRLM failures" on page 214
- "Database backout recovery" on page 217
- "Database forward recovery" on page 218.

DBRC failures

One logical RECON data set contains the control information for the whole data-sharing configuration. Protecting the RECON data set is therefore the most important DBRC recovery consideration.

Restart after DBRC failure

Because DBRC runs under the control of CICS when local DL/I is being used, CICS knows if it fails. If DBRC fails, CICS terminates abnormally. After the DBRC problem has been corrected, you can restart CICS using START=AUTO.

Offline actions

When DBRC is inactive, you can run an IMS utility with the execution-time parameter DBRC=NO, or you can do the equivalent of running a utility, by nonstandard means. In these cases, you must update the RECON data set, using either a **NOTIFY** command to advise DBRC of the event, or a **CHANGE** command to alter RECON information directly.

RECON data set recovery

You should plan never to have a RECON problem serious enough to stop your sharing operations or lose the recoverability of your databases. If you do lose the RECON data set, you must recover it before sharing can continue.

For maximum security:

1. Define three RECON data sets for the whole installation.
2. Place each RECON data set:
   - On a separate device
   - On a separate channel
   - In a separate user catalog.
3. In each subsystem
   - Use all three RECON data sets.
   - Dynamically allocate the RECON data sets.
4. In interhost data sharing (with recoverable VSAM catalogs)
   - Have no other VSAM data sets on the same volume as a RECON data set.

You can minimize the effect of losing all your RECON data sets, by frequently using the DBRC command, BACKUP.RECON. You can issue this command only as input to the DBRC batch utility. The appropriate IMS manual gives the procedure to follow if all your RECON data sets become unusable.

IRLM failures

Four types of IRLM failure can occur:

- A configuration with a single IRLM that fails
- A configuration with two IRLMs that fail to establish a communications session
- A configuration with two communicating IRLMs in which an IRLM fails
- A configuration with two communicating IRLMs in which the communications link fails.

A single failing IRLM

If the single IRLM in a configuration fails, each online subsystem (including any CICS subsystems) backs out the affected transactions, and continues processing (without access to DL/I databases).

The IRLM, if present, is used for all lock management, so all DL/I databases are inaccessible after the IRLM failure, until you have restarted the IRLM and reconnected the requesting subsystem.

When you have restarted the IRLM, reconnect your CICS system(s) to the IRLM with the CEMT PERFORM RECONNECT command.

Configurations with two IRLMs

In a configuration with two IRLMs, three error conditions are possible:

- The two IRLMs cannot establish an SNA session.
- One IRLM fails.
- The VTAM communications link fails.

You must decide which error has occurred. The first indication of a problem is an IRLM message, DXR025I or DXR027A.

The text of message DXR025I is:

```
DXR025I irlm SESSION LOST, SHARING STATE
             IS INITIAL|IRLM FAILED|COMM FAILED|SYSTEM FAILED
```

Interpret the sharing state as follows:

INITIAL The local IRLM has failed to establish a session with the remote IRLM.

IRLM FAILED The remote IRLM has failed. (The actions for subsystems using the failed IRLM are described under “A single failing IRLM.”)

COMM FAILED The VTAM communications link has failed.

The text of message DXR027A is:

```
DXR027A irlm SESSION LOST, SHARING STATE
             IS IN DOUBT - ACTION REQUIRED
```

If both IRLMs issue message DXR027A, the VTAM communications link has failed.

If only one IRLM issues message DXR027A, the other has probably failed. Confirm this by displaying the jobs active in its host processor (if the IRLM has failed the IRLM procedure name will not appear).

Note: If a communications failure coincides with an IRLM failure, you must take the action specified in this section for an IRLM failure. Your operational procedures should ensure that an operator never issues the MVS MODIFY command specifying a COMM failure when an IRLM has failed.

The next section, "Connection failure," tells you how to deal with all the above error conditions except a communications failure, which is in "Communications link failure" on page 216.

Connection failure

When an IRLM connection failure occurs, surviving IRLMs can be in one of four states, INITIAL, IRLM FAILED, SYSTEM FAILED, or IN DOUBT.

INITIAL: If both IRLMs have issued message DXR025I with a sharing state of INITIAL, they have failed to establish a session. Each continues as a single IRLM, and intrahost data sharing is possible in each host processor. When you have solved the underlying VTAM problem, you can establish communication and interhost data sharing by issuing a MODIFY command specifying ACTCOMM to either IRLM.

IRLM FAILED: When one IRLM fails, it attempts to inform the other IRLM that it is terminating. If this attempt is successful, the nonfailing IRLM issues message DXR025I with a sharing state of IRLM FAILED, and continues as a single IRLM (intrahost data sharing can continue in its host processor). To reestablish communication and interhost data sharing, you must restart the failing IRLM, and then issue a MODIFY command specifying ACTCOMM to the nonfailing IRLM.

SYSTEM FAILED: When one IRLM fails, it tries to inform the other IRLM that it is terminating. If this attempt is successful, and the IRLM procedures specified RULES=AVAIL, the nonfailing IRLM issues message DXR0251 with a sharing state of SYSTEM FAILED. The surviving IRLM continues to grant new lock requests, as long as they do not conflict with an update lock retained because of a previous request by a failing subsystem. In the system failed state, IMS does not stop the databases that retain such locks. When the IRLM enters system failed state, it automatically issues a MODIFY irlm, ACTCOMM, to establish a session between the two IRLMs.

IN DOUBT: When a failing IRLM does not succeed in informing the other IRLM that it is terminating, the nonfailing IRLM issues message DXR027A.

Nonfailing IRLM in IN DOUBT state

When recovering from IRLM connection failure with the nonfailing IRLM in IN DOUBT state, the procedure for IMS with RULES=COMPAT specified or defaulted is:

1. Enter to the nonfailing IRLM:

   ```
   F irlmproc SETSTATE,IRLM
   ```

   Wait for this response:

   ```
   DXR026I IRLM SETSTATE COMMAND ACCEPTED STATE IS IRLM FAILED
   ```

2. Enter to the nonfailing IRLM

   ```
   F irlmproc STATUS
   ```

   Wait for this response:

   ```
   DXR010I xxxx yyyy STATUS INITIAL
   ```

3. Restart failed IRLM.
4. Enter to the nonfailing IRLM

   ```
   F irlmproc ACTCOMM
   ```

Nonfailing IRLM in IN DOUBT state and RULES=AVAIL

When recovering from IRLM connection failure with the nonfailing IRLM in IN DOUBT state and RULES=AVAIL specified for IMS, the recovery procedure is:

1. Enter to the nonfailing IRLM:

   ```
   F irlmproc,SETSTATE,SYSTEM
   ```

   Wait for this response:

   ```
   DXR026I IRLM SETSTATE COMMAND ACCEPTED STATE IS SYSTEM FAILED
   ```

2. Restart the failed IRLM.
3. Wait for message:

   ```
   DXR002I irlm x VTAM SESSION WITH y ESTABLISHED
   ```

   before connecting CICS to the restarted IRLM.

Communications link failure

If the communications link between two IRLMs fails, each IRLM (with its subsystems) continues, although any global lock requests must wait. Each IRLM enters the IN DOUBT state, and issues the console message, DXR027A. Any batch programs or CICS transactions that make lock requests will effectively go into wait state.

The preferred method of recovery is:

1. Reestablish the communications link.
2. Issue a MODIFY ACTCOMM system console command for one or both IRLMs to reestablish IRLM communication.

The IRLMs then start servicing lock requests again, data-sharing operations resume, and only terminal response times have suffered. If a VTAM restart automatically restores the link, or if you can reestablish it quickly (for example, by issuing a VARY NET, ACTIVE command for a failing LU), you will obviously choose this method.

If you cannot restore the link quickly, you can restart the IRLMs for intrahost data sharing before restoring communication.

Restart independent IRLM processing

The procedure to restart independent IRLM processing after a communications link failure is:

1. Enter to each IRLM

   ```
   F irlmproc,SETSTATE,COMM
   ```

   Wait for this response from each IRLM:

   ```
   DXR035A xxxx ENTER SETSTATE,INIT WHEN BOTH IRLMS ISSUE THIS MESSAGE
   ```

2. Enter to each IRLM

   ```
   F irlmproc,SETSTATE,INIT
   ```

3. If necessary, run batch backout for IMS batch subsystem(s).

When you have resolved the communication link problem and restored the link, you can reestablish communication between the IRLMs. Enter the MODIFY command with the ACTCOMM parameter to either IRLM. This causes full data-sharing operations to resume.

Note: If you use the procedure in “Nonfailing IRLM in IN DOUBT state” on page 215, each shared database is deauthorized to one IRLM or the other. Here is a way to ensure that your most important subsystems retain their authorizations:

When planning your configuration, group your most important subsystems in one host. The IRLM in that host is then your “important” IRLM. In your operating instructions for a communications failure, include the following steps in the sequence given:

1. Issue SETSTATE,COMM command to “less-important” IRLM
2. Wait for message DXR035A from “less-important” IRLM
3. Issue SETSTATE,COMM command to “important” IRLM
4. Wait for message DXR035A from “important” IRLM.

These steps replace step 1 in “Restart independent IRLM processing.”

Database backout recovery

CICS dynamic backout

When backout is needed because a transaction fails or issues a ROLLBACK request, CICS backs out all uncommitted local DL/I database changes made by the transaction (just as it does in a nonsharing environment). Data integrity is assured.

Backout during CICS emergency restart

If a CICS system fails, DBRC and IRLM retain any authorizations and locks held for the failing system. This does not affect other data-sharing subsystems, except that they cannot access data for which the failed subsystem holds locks. If an access attempt is made in these circumstances, an IMS pseudo abend is issued.

During emergency restart, CICS backs out all uncompleted transactions. When emergency restart completes successfully, DBRC and IRLM release the authorizations and locks held for the failing CICS system. Data integrity is assured.

CICS backout failure

If a backout failure occurs during CICS dynamic backout or emergency restart:

- IMS stops each affected database, and, if the database is registered to DBRC, increments a “backout needed” count in the RECON entry for that database.
- CICS issues message DFHDL3907 for each stopped database, and a separate message (DFHDL3905 during dynamic backout, or DFHDL3906 during emergency restart) for each affected PSB.

To restore the local DL/I databases, you must run the IMS batch backout utility. First ensure that system execution and backout can proceed independently:

1. On the CICS system that experienced the backout failure, issue a single CEMT SET DLIDATABASE(fileid) RECOVERDB command for all the affected databases. This closes and deauthorizes the databases, and, most important, forces a single log switch.
2. If you are using disk logging, use the IMS log archive utility DFSUARC0 to copy the now inactive log to tape.
3. Run the batch backout utility (DFSBBO00) once for each PSB named in a message.

Database forward recovery

Reconstructing all or part of a lost or destroyed database in the data-sharing environment is much the same as it is in a non-data-sharing environment (see Chapter 21, “Using IMS DBRC for recovery control” on page 203). You use the IMS forward recovery utility (DFSURDB0), and give it your copy of the lost database, and all the pertinent system log data sets used since that copy was taken. Tapes produced by DFSUARC0 may be used as input to DL/I forward recovery utilities, but not to a CICS emergency restart.

For block-level sharing, you must use DFSUCUM0 (see “Utilities” on page 204) to merge the logs of each subsystem that can update the database, and use the merged log as input to forward recovery. DBRC automatically generates the JCL to include these steps if required.

Before you can do forward recovery, you must take your local DL/I database offline (deallocate it). IMS/ESA provides dynamic allocation and deallocation support. See “Dynamic allocation and deallocation of local DL/I databases” on page 197. If you are not using IMS dynamic allocation, you must allocate and deallocate each database data set individually.

CICS system programming requirements

“CICS system programming requirements” on page 206 describes the system programming requirements caused by the use of DBRC. All of these apply in a data-sharing environment. This chapter has mentioned the additional recovery and restart tasks that arise if you use data sharing—planning and implementing the various recovery and restart procedures described.

As with DBRC, you should install IRLM before you add IRLM support to your CICS system. The difference is that, unlike DBRC, IRLM is not installed as part of IMS, and you must install it separately. For information about installing IRLM, see the appropriate IMS manual.

CICS region size

Tasks will, on average, last slightly longer in a data-sharing environment due to the extra processing overhead. Therefore, to support a given transaction rate, you may need a larger CICS region.

System initialization definition

Apart from the entries needed to define DBRC, summarized in the previous chapter, note the following system initialization parameters for data sharing.

You must code DLIRLM=YES|name. The name is the MVS subsystem name and defaults to IRLM. Due to the processing needed for data-sharing control, DL/I threads remain longer in the system, leading to a need to change some CICS tuning parameters not directly connected with data sharing: DLTHRED and MXT, and the RDO transaction classes.

The APPLID system initialization parameter specifies the subsystem name of your CICS system and must be unique in the data-sharing configuration. If you start two CICS subsystems with the same APPLID, they will use the same SUBSYSTEM record in the RECON data set, and you will lose data protection. To enable you to distinguish between IMS messages for different CICS systems, specify a unique SYSIDNT at system initialization for each CICS system. To identify the subsystem to which a message refers, IMS appends the SYSIDNT value to its messages: see “Messages” on page 220.

You may also want to change some performance parameters.

Table 14. VSAM share options and JCL dispositions for shared databases

Share level	Type of sharing allowed	VSAM share option	JCL disposition
0	No sharing. One user at a time has exclusive control	(1,3)	OLD
1	Database level sharing. One updater and multiple readers	(1,3)	SHR
2	Intrahost block level. Multiple updaters and multiple readers	(3,3)	SHR
3	Intrahost block level. Multiple updaters and multiple readers	(3,3)	SHR

Starting to use data sharing

If you use DBRC for recovery control only, specify RECOVCTL in the INIT.RECON command to initialize your RECON data set. To activate data sharing, you must issue a CHANGE.RECON SHARECTL command. To avoid the risk of changing the control environment while any subsystem is accessing a registered database, enter this command in batch (see “Submitting DBRC commands” on page 204). It is also best to retain all your databases at a share level of 0 until after you have changed the DBRC control level.

To raise the share level of a database, use the CHANGE.DB SHARELVL command. Do not make this change while any subsystem is authorized to the database.

For each database data set that is to be shared, you must specify a JCL disposition consistent with its share level, and, if any of the data sets are VSAM data sets, you must allocate them with an appropriate VSAM share option. Table 14 on page 219 shows the JCL dispositions and VSAM share options that you should use with the various share levels.

Ensure that the ACCESS parameters on your DFHDLDBD TYPE=ENTRY macros defining shared databases to CICS correctly reflect the required usage. If you make the ACCESS parameter more restrictive than needed, you will unnecessarily restrict sharing. The PROCOPT values of the PSBs use by CICS or an IMS batch data-sharing region should be the same as or less restrictive than the actual access intent.

For information about the operating considerations for running CICS as a data-sharing subsystem, see the *CICS/ESA Operations and Utilities Guide*.

Messages

The following table shows the source of each message received in a data-sharing environment:

Source	Initial identifier of message
CICS	DFH
IMS	DFS
DBRC	DSP
IRLM	DXR
MVS	Ixx (xx varies)

You also need to know which subsystem a message refers to.

The subsystem identifier always follows IMS messages. For a CICS subsystem using local DL/I, this is the SYSIDNT system initialization parameter (default CICS). If your SYSIDNT is CICS, it will appear in messages as CICA.

Some CICS messages, including DFHSI1517 “CONTROL IS BEING GIVEN TO CICS”, follow the APPLID parameter (default ‘DBDCCICS’), specified as a system initialization parameter.

If a message includes neither a SYSIDNT nor an APPLID parameter, you can use the MVS job number to identify the subsystem.

Index

Sending your comments to IBM

CICS/ESA

Recovery and Restart Guide

SC33-1182-01

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book. Please limit your comments to the information in this book and the way in which the information is presented.

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, use the Readers' Comment Form
- By fax:
  - From outside the U.K., after your international access code use 44 962 870229 (after 16 April 1995, use 44 1962 870229)
  - From within the U.K., use 0962 870229 (after 16 April 1995, use 01962 870229)
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: WINVMJ(IDRCF)
  - Internet: idrcf@winvmj.vnet.ibm.com

Whichever 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.

Readers' Comments

CICS/ESA

Recovery and Restart Guide

SC33-1182-01

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

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. This form is provided for comments about the information in this manual and the way it is presented.

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.

Be sure to print your name and address below if you would like a reply.

Name

Address

Company or Organization

Telephone

Email

CICS/ESA

CICS/ESA Recovery and Restart Guide
SC33-1182-01

You can send your comments POST FREE on this form from any one of these countries:

Australia
Belgium
Bermuda
Cyprus
Denmark
Finland
France
Germany
Greece
Hong Kong
Iceland
Israel
Italy
Luxembourg
Monaco
Netherlands
New Zealand
Norway
Portugal
Republic of Ireland
Singapore
Spain
Sweden
Switzerland
United Arab Emirates
United States of America

If your country is not listed here, your local IBM representative will be pleased to forward your comments to us. Or you can pay the postage and send the form direct to IBM (this includes mailing in the U.K.).

1 Cut along this line

2 Fold along this line

By air mail
Par avion

IBRS/CCRI NUMBER: PHQ - D/1348/SO

NE PAS AFFRANCHIR

NO STAMP REQUIRED

REPONSE PAYEE
GRANDE-BRETAGNE

IBM United Kingdom Laboratories Limited
Information Development Department (MP 095)
Hursley Park
WINCHESTER, Hants
SO21 2ZZ United Kingdom

3 Fold along this line

From: Name ________________________________
Company or Organization ________________________
Address ________________________________
____
EMAIL ________________________________
Telephone ________________________________

1 Cut along this line

4 Fasten here with adhesive tape

IBM®

Program Number: 5655-018

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-1182-01