CICS Transaction Server for z/OS
Version 4 Release 1

Recovery and Restart Guide

SC34-7012-03

CICS Transaction Server for z/OS
Version 4 Release 1

Recovery and Restart Guide

SC34-7012-03

Note
Before using this information and the product it supports, read the information in "Notices" on page 243.

This edition applies to Version 4 Release 1 of CICS Transaction Server for z/OS (product number 5655-S97) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Preface

What this book is about

This book contains guidance about determining your CICS® recovery and restart needs, deciding which CICS facilities are most appropriate, and implementing your design in a CICS region.

The information in this book is generally restricted to a single CICS region. For information about interconnected CICS regions, see the *CICS Intercommunication Guide*.

This manual does not describe recovery and restart for the CICS front end programming interface. For information on this topic, see the *CICS Front End Programming Interface User's Guide*.

Who should read this book

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

What you need to know to understand this book

To understand this book, you should have experience of installing CICS 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 can find the sections relevant to your work. Readers new to recovery and restart should find the first section helpful, because it introduces the concepts of recovery and restart.

Changes in CICS Transaction Server for z/OS, Version 4 Release 1

For information about changes that have been made in this release, please refer to *What's New* in the information center, or the following publications:

- *CICS Transaction Server for z/OS What's New*
- *CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.2*
- *CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.1*
- *CICS Transaction Server for z/OS Upgrading from CICS TS Version 2.3*

Any technical changes that are made to the text after release are indicated by a vertical bar (|) to the left of each new or changed line of information.

Part 1. CICS recovery and restart concepts

It is very important that a transaction processing system such as CICS can restart and recover following a failure. This section describes some of the basic concepts of the recovery and restart facilities provided by CICS.

Chapter 1. Recovery and restart facilities

Problems that occur in a data processing system could be failures with communication protocols, data sets, programs, or hardware. These problems are potentially more severe in online systems than in batch systems, because the data is processed in an unpredictable sequence from many different sources.

Online applications therefore require a system with special mechanisms for recovery and restart that 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. Together with suitable operating procedures, these mechanisms should 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 and consistency of data
- To minimize the effect of failures

CICS provides a facility to meet these two requirements called the recovery manager. The CICS recovery manager provides the recovery and restart functions that are needed in an online system.

Maintaining the integrity of data

Data integrity means that the data is in the form you expect and has not been corrupted. The objective of recovery operations on files, databases, and similar data resources is to maintain and restore the integrity of the information.

Recovery must also ensure consistency of related changes, whereby they are made as a whole or not at all. (The term **resources** used in this book, unless stated otherwise, refers to data resources.)

Logging changes

One way of maintaining the integrity of a resource 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.

An online system can use the logged 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.

Minimizing the effect of failures

An 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 should 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.

Ideally, it should be possible to restore the data to a consistent, known state following any type of failure, with minimal loss of valid updating activity.

The role of CICS

The CICS recovery manager and the log manager perform the logging functions necessary to support automatic backout. Automatic backout is provided for most CICS resources, such as databases, files, and auxiliary temporary storage queues, either following a transaction failure or during an emergency restart of CICS.

If the backout of a VSAM file fails, CICS backout failure processing ensures that all locks on the backout-failed records are retained, and the backout-failed parts of the unit of work (UOW) are shunted to await retry. The VSAM file remains open for use. For an explanation of shunted units of work and retained locks, see “Shunted units of work” on page 13.

If the cause of the backout failure is a physically damaged data set, and provided the damage affects only a localized section of the data set, you can choose a time when it is convenient to take the data set offline for recovery. You can then use the forward recovery log with a forward recovery utility, such as CICS VSAM Recovery, to restore the data set and re-enable it for CICS use.

Note: In many cases, a data set failure also causes a processing failure. In this event, forward recovery must be followed by backward recovery.

You don't need to shut CICS down to perform these recovery operations. For data sets accessed by CICS in VSAM record-level sharing (RLS) mode, you can **quiesce** the data set to allow you to perform the forward recovery offline. On completion of forward recovery, setting the data set to **unquiesced** causes CICS to perform the backward recovery automatically.

For files accessed in non-RLS mode, you can issue a **SET DSNAME RETRY** command after the forward recovery, which causes CICS to perform the backward recovery online.

Another way is to shut down CICS with an immediate shutdown and perform the forward recovery, after which a CICS emergency restart performs the backward recovery.

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 (but user-maintained data tables are not recovered after a CICS failure, only after a transaction failure)
- Coupling facility data tables
- The CICS system definition (CSD) file
- Intrapartition transient data destinations
- Auxiliary temporary storage queues
- Resource definitions dynamically installed using resource definition online (RDO)

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 DFSMSdss, in a separate job under MVS™, to back up a 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 are recorded.

DFSMSdss is a functional component of DFSMS/MVS, and is the primary data mover. When used with supporting hardware, DFSMSdss also provides a concurrent copy capability. This capability enables you to copy or back up data while that data is being used.

If a data set failure occurs, you can use a backup of the data set and a forward recovery utility, such as CICS VSAM Recovery (CICSVR), to recover the VSAM file.

CICS 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.

Before a change is made to a resource, the recovery information for backout, in the form of a before-image, is recorded on the CICS system log. A before-image is a record of what the resource was like before the change. These before-images are used by CICS to perform backout in two situations:

- In the event of failure of an individual in-flight transaction, which CICS backs out dynamically at the time of failure (dynamic transaction backout)

- In the event of an emergency restart, when CICS backs out all those transactions that were in-flight at the time of the CICS failure (emergency restart backout).

Although these occur in different situations, CICS uses the same backout process in each case. CICS does not distinguish between dynamic backout and emergency restart backout. See Chapter 6, "CICS emergency restart," on page 61 for an explanation of how CICS reattaches failed in-flight units of work in order to perform transaction backout following an emergency restart.

Each CICS region has only one system log, which cannot be shared with any other CICS region. The system log is written to a unique MVS system logger log stream. The CICS system log is intended for use only for recovery purposes, for example during dynamic transaction backout, or during emergency restart. It is not meant to be used for any other purpose.

CICS supports two physical log streams - a primary and a secondary log stream. CICS uses the secondary log stream for storing log records of failed units of work, and also some long running tasks that have not caused any data to be written to the log for two complete activity key points. Failed units of work are moved from the primary to the secondary log stream at the next activity keypoint. Logically, both the primary and secondary log stream form one log, and as a general rule are referred to as the system log.

Dynamic transaction backout

In the event of a transaction failure, or if an application explicitly requests a syncpoint rollback, the CICS recovery manager uses the system log data to drive the resource managers to back out any updates made by the current unit of work.

This process, known as dynamic transaction backout, takes place while the rest of CICS continues normally.

For example, when any updates made to a recoverable data set are to be backed out, file control uses the system log records to reverse the updates. When all the updates made in the unit of work have been backed out, the unit of work is completed. The locks held on the updated records are freed if the backout is successful.

For data sets open in RLS mode, CICS requests VSAM RLS to release the locks; for data sets open in non-RLS mode, the CICS enqueue domain releases the locks automatically.

See "Units of work" on page 13 for a description of units of work.

Emergency restart backout

If a CICS region fails, you restart CICS with an emergency restart to back out any transactions that were in-flight at the time of failure.

During emergency restart, the recovery manager uses the system log data to drive backout processing for any units of work that were in-flight at the time of the failure. The backout of units of work during emergency restart is the same as a dynamic backout; there is no distinction between the backout that takes place at emergency restart and that which takes place at any other time. At this point, while recovery processing continues, CICS is ready to accept new work for normal processing.

The recovery manager also drives:

- The backout processing for any units of work that were in a backout-failed state at the time of the CICS failure
- The commit processing for any units of work that had not finished commit processing at the time of failure (for example, for resource definitions that were being installed when CICS failed)
- The commit processing for any units of work that were in a commit-failed state at the time of the CICS failure

See "Unit of work recovery" on page 73 for an explanation of the terms *commit-failed* and *backout-failed*.

The recovery manager drives these backout and commit processes because the condition that caused them to fail might be resolved by the time CICS restarts. If the condition that caused a failure has not been resolved, the unit of work remains in backout- or commit-failed state. See "Backout-failed recovery" on page 79 and "Commit-failed recovery" on page 83 for more information.

CICS 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 forward recovery log (a general log stream managed by the MVS system logger).
3. After the failure, restore the most recent backup copy of the failed data set, and use the information recorded on the forward recovery log to update the data set with all the changes that have occurred since the backup copy was taken.

These operations are known as *forward recovery*. On completion of the forward recovery, as a fourth step, CICS also performs backout of units of work that failed in-flight as a result of the data set failure.

Forward recovery of CICS data sets

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

CICS writes the after-images of changes made to a data set to a forward recovery log, which is a general log stream managed by the MVS system logger.

CICS obtains the log stream name of a VSAM forward recovery log in one of two ways:

1. For files opened in VSAM record level sharing (RLS) mode, the explicit log stream name is obtained directly from the VSAM ICF catalog entry for the data set.
2. For files in non-RLS mode, the log stream name is derived from:
   - The VSAM ICF catalog entry for the data set if it is defined there, and if `RLS=YES` is specified as a system initialization parameter. In this case, CICS file control manages writes to the log stream directly.
   - A journal model definition referenced by a forward recovery journal name specified in the file resource definition.

Forward recovery journal names are of the form DFHJ*nn* where *nn* is a number in the range 1–99 and is obtained from the forward recovery log id (FWDRECOVLOG) in the FILE resource definition.

In this case, CICS creates a journal entry for the forward recovery log, which can be mapped by a JOURNALMODEL resource definition. Although this method enables user application programs to reference the log, and write user journal records to it, you are recommended not to do so. You should ensure that forward recovery log streams are reserved for forward recovery data only.

Note: You cannot use a CICS system log stream as a forward recovery log.

The VSAM recovery options or the CICS file control recovery options that you require to implement forward recovery are explained further in "Defining files as recoverable resources" on page 126.

For details of procedures for performing forward recovery, see Chapter 17, "Forward recovery procedures," on page 187.

Forward recovery for non-VSAM resources

CICS does not provide forward recovery logging for non-VSAM resources, such as BDAM files. However, you can provide this support yourself by ensuring that the necessary information is logged to a suitable log stream. In the case of BDAM files, you can use the CICS autojournaling facility to write the necessary after-images to a log stream.

Failures that require CICS recovery processing

The following section briefly describes CICS recovery processing after a communication failure, transaction failure, and 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.

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
- Network path failure
- Loss of an MVS image that is a member of a sysplex

There are two aspects to processing following a communications failure:

1. If the failure occurs during a conversation that is not engaged in syncpoint protocol, CICS must terminate the conversation and allow customized handling of the error, if required. An example of when such customization is helpful is for 3270 device types. This is described below.

2. If the failure occurs during the execution of a CICS syncpoint, where the conversation is with another resource manager (perhaps in another CICS region), CICS handles the resynchronization. This is described in the *CICS Intercommunication Guide*.

If the link fails and is later reestablished, CICS and its partners 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 Intercommunication Guide*.

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 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 10).
- Reducing the amount of error information printed.

For more information about NEPs and TEPs, see Chapter 9, "Communication error processing," on page 97.

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

Transactions can fail for a variety of reasons, including a program check in an application program, an invalid request from an application that causes an abend, a task issuing an ABEND request, or I/O errors on a data set that is being accessed by a transaction.

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

After dynamic transaction backout 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 93.

If dynamic transaction backout fails, perhaps because of an I/O error on a VSAM data set, CICS backout failure processing shunts the unit of work and converts the locks that are held on the backout-failed records into retained locks. The data set remains open for use, allowing the shunted unit of work to be retried. If backout keeps failing because the data set is damaged, you can create a new data set using a backup copy and then perform forward recovery, using a utility such as CICSVR. When the data set is recovered, retry the shunted unit of work to complete the failed backout and release the locks.

Chapter 8, "Unit of work recovery and abend processing," on page 73 gives more details about CICS processing of a transaction failure.

CICS recovery processing following a system failure

Causes of a system failure include a processor failure, the loss of a electrical power supply, an operating system failure, or a CICS failure.

During normal execution, CICS stores recovery information on its **system log stream**, which is managed by the MVS system logger. If you specify START=AUTO, CICS automatically performs an **emergency restart** when it restarts after a system failure.

During an emergency restart, the CICS log manager reads the system log backward and passes information to the CICS recovery manager.

The CICS recovery manager then uses the information retrieved from the system log to:

- Back out recoverable resources.
- Recover changes to terminal resource definitions. (All resource definitions installed at the time of the CICS failure are initially restored from the CICS global catalog.)

A special case of CICS processing following a system failure is covered in Chapter 6, “CICS emergency restart,” on page 61.

Chapter 2. Resource recovery in CICS

Before you begin to plan and implement resource recovery in CICS, you should understand the concepts involved, including units of work, logging and journaling.

Units of work

When resources are being changed, there comes a point when the changes are complete and do not need backout if a failure occurs later. The period between the start of a particular set of changes and the point at which they are complete is called a *unit of work* (UOW). The unit of work is a fundamental concept of all CICS backout mechanisms.

From the application designer's point of view, a UOW is a sequence of actions that needs to be complete before any of the individual actions can be regarded as complete. To ensure data integrity, a unit of work must be atomic, consistent, isolated, and durable.

The CICS recovery manager operates with units of work. If a transaction that consists of multiple UOWs fails, or the CICS region fails, committed UOWs are not backed out.

A unit of work can be in one of the following states:

- Active (in-flight)
- Shunted following a failure of some kind
- Indoubt pending the decision of the unit of work coordinator.
- Completed and no longer of interest to the recovery manager

Shunted units of work

A shunted unit of work is one awaiting resolution of an indoubt failure, a commit failure, or a backout failure. The CICS recovery manager attempts to complete a shunted unit of work when the failure that caused it to be shunted has been resolved.

A unit of work can be unshunted and then shunted again (in theory, any number of times). For example, a unit of work could go through the following stages:

1. A unit of work fails indoubt and is shunted.
2. After resynchronization, CICS finds that the decision is to back out the indoubt unit of work.
3. Recovery manager unshunts the unit of work to perform backout.
4. If backout fails, it is shunted again.
5. Recovery manager unshunts the unit of work to retry the backout.
6. Steps 4 and 5 can occur several times until the backout succeeds.

These situations can persist for some time, depending on how long it takes to resolve the cause of the failure. Because it is undesirable for transaction resources to be held up for too long, CICS attempts to release as many resources as possible while a unit of work is shunted. This is generally achieved by abending the user task to which the unit of work belongs, resulting in the release of the following:

- Terminals
- User programs

- Working storage
- Any LU6.2 sessions
- Any LU6.1 links
- Any MRO links

The resources CICS retains include:

- Locks on recoverable data. If the unit of work is shunted indoubt, all locks are retained. If it is shunted because of a commit- or backout-failure, only the locks on the failed resources are retained.
- System log records, which include:
  - Records written by the resource managers, which they need to perform recovery in the event of transaction or CICS failures. Generally, these records are used to support transaction backout, but the RDO resource manager also writes records for rebuilding the CICS state in the event of a CICS failure.
  - CICS recovery manager records, which include identifiers relating to the original transaction such as:
    - The transaction ID
    - The task ID
    - The CICS terminal ID
    - The VTAM LUNAME
    - The user ID
    - The operator ID.

Locks

For files opened in RLS mode, VSAM maintains a single central lock structure using the lock-assist mechanism of the MVS coupling facility. This central lock structure provides sysplex-wide locking at a record level. Control interval (CI) locking is not used.

The locks for files accessed in non-RLS mode, the scope of which is limited to a single CICS region, are file-control managed locks. Initially, when CICS processes a read-for-update request, CICS obtains a CI lock. File control then issues an ENQ request to the enqueue domain to acquire a CICS lock on the specific record. This enables file control to notify VSAM to release the CI lock before returning control to the application program. Releasing the CI lock minimizes the potential for deadlocks to occur.

For coupling facility data tables updated under the locking model, the coupling facility data table server stores the lock with its record in the CFDT. As in the case of RLS locks, storing the lock with its record in the coupling facility list structure that holds the coupling facility data table ensures sysplex-wide locking at record level.

For both RLS and non-RLS recoverable files, CICS releases all locks on completion of a unit of work. For recoverable coupling facility data tables, the locks are released on completion of a unit of work by the CFDT server.

Active and retained states for locks

CICS supports active and retained states for locks.

When a lock is first acquired, it is an active lock. It remains an active lock until successful completion of the unit of work, when it is released, or is converted into a retained lock if the unit of work fails, or for a CICS or SMSVSAM failure:

- If a unit of work fails, RLS VSAM or the CICS enqueue domain continues to hold the record locks that were owned by the failed unit of work for recoverable data sets, but converted into retained locks. Retaining locks ensures that data integrity for those records is maintained until the unit of work is completed.
- If a CICS region fails, locks are converted into retained locks to ensure that data integrity is maintained while CICS is being restarted.
- If an SMSVSAM server fails, locks are converted into retained locks (with the conversion being carried out by the other servers in the sysplex, or by the first server to restart if all servers have failed). This means that a UOW that held active RLS locks will hold retained RLS locks following the failure of an SMSVSAM server.

Converting active locks into retained locks not only protects data integrity. It also ensures that new requests for locks owned by the failed unit of work do not wait, but instead are rejected with the LOCKED response.

Synchronization points

The end of a UOW 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 as a result of an **EXEC CICS RETURN** command at the highest logical level. This means that a UOW cannot span tasks.
- Explicitly by **EXEC CICS SYNCPOINT** commands issued by an application program 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 UOW.

  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 through one of the following CICS commands:
  - **EXEC CICS CREATE TERMINAL**
  - **EXEC CICS CREATE CONNECTION COMPLETE**
  - **EXEC CICS DISCARD CONNECTION**
  - **EXEC CICS DISCARD TERMINAL**
- Implicitly by a program called by a distributed program link (DPL) command if the SYNCONRETURN option is specified. When the DPL program terminates with an **EXEC CICS RETURN** command, the CICS mirror transaction takes a syncpoint.

It follows from this that a unit of work starts:

- At the beginning of a transaction
- Whenever an explicit syncpoint is issued and the transaction does not end
- Whenever a DL/I PSB termination call causes an implicit syncpoint and the transaction does not end
- Whenever one of the following CICS commands causes an implicit syncpoint and the transaction does not end:
  - **EXEC CICS CREATE TERMINAL**

- **EXEC CICS CREATE CONNECTION COMPLETE**
- **EXEC CICS DISCARD CONNECTION**
- **EXEC CICS DISCARD TERMINAL**

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

A unit of work can also be ended by backout, which causes a syncpoint in one of the following ways:

- Implicitly when a transaction terminates abnormally, and CICS performs dynamic transaction backout
- Explicitly by **EXEC CICS SYNCPOINT ROLLBACK** commands issued by an application program to backout changes made by the UOW.

Examples of synchronization points

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

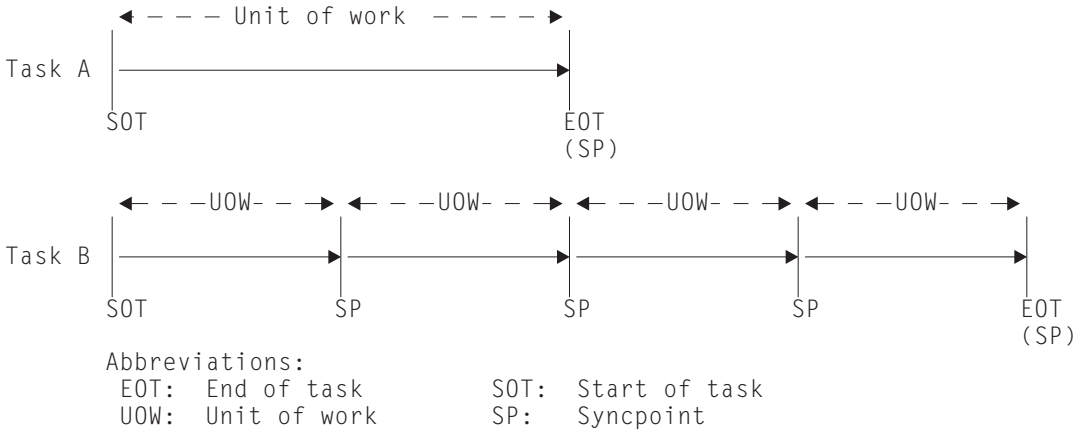


Figure 1. Units of work and syncpoints

Figure 2 on page 17 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 UOW 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.

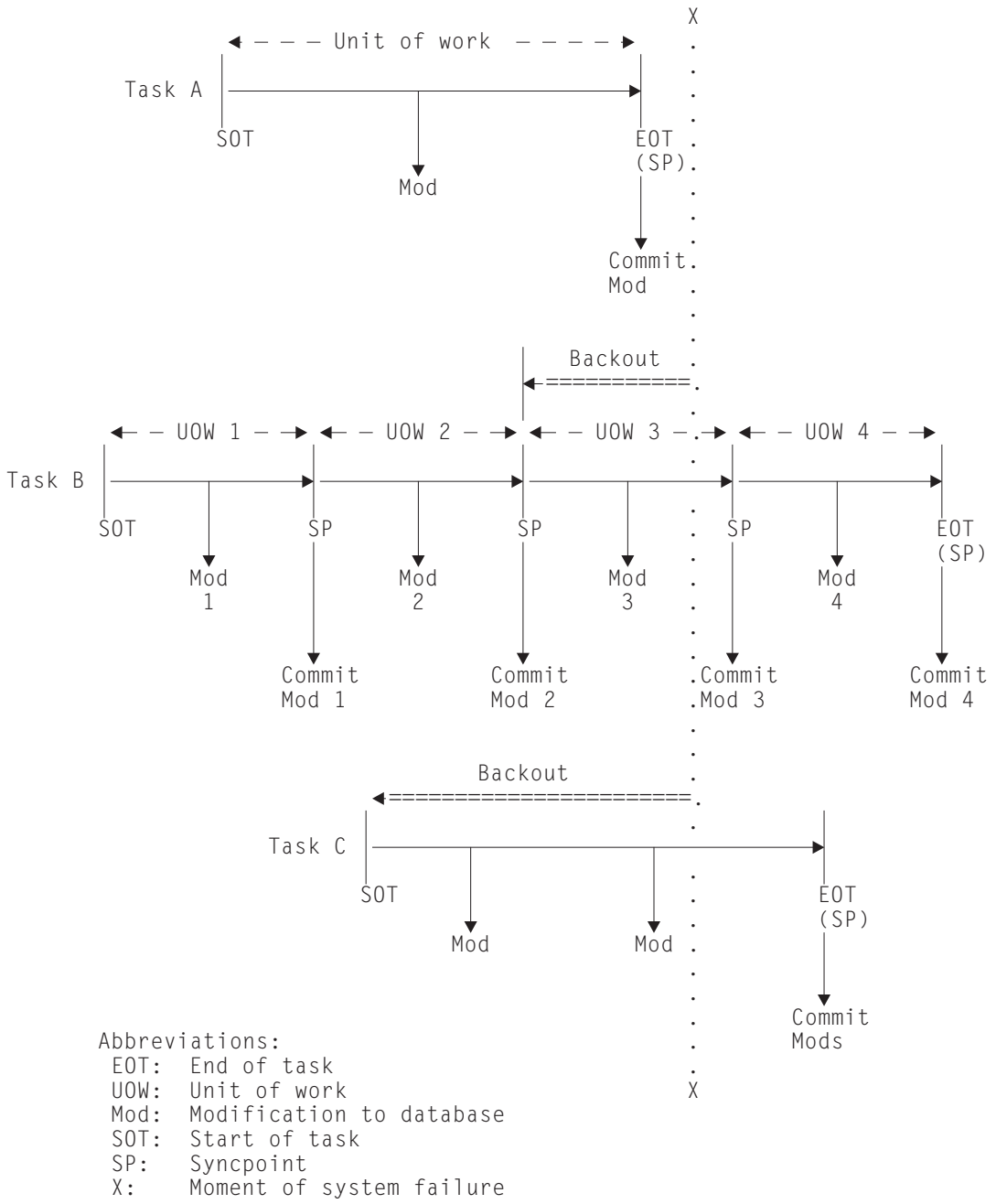


Figure 2. Backout of units of work

CICS recovery manager

The recovery manager ensures the integrity and consistency of resources (such as files and databases) both within a single CICS region and distributed over interconnected systems in a network.

Figure 3 on page 18 shows the resource managers and their resources with which the CICS recovery manager works.

The main functions of the CICS recovery manager are:

- Managing the state, and controlling the execution, of each UOW
- Coordinating UOW-related changes during syncpoint processing for recoverable resources
- Coordinating UOW-related changes during restart processing for recoverable resources
- Coordinating recoverable conversations to remote nodes
- Temporarily suspending completion (**shunting**), and later resuming completion (**unshunting**), of UOWs that cannot immediately complete commit or backout processing because the required resources are unavailable, because of system, communication, or media failure

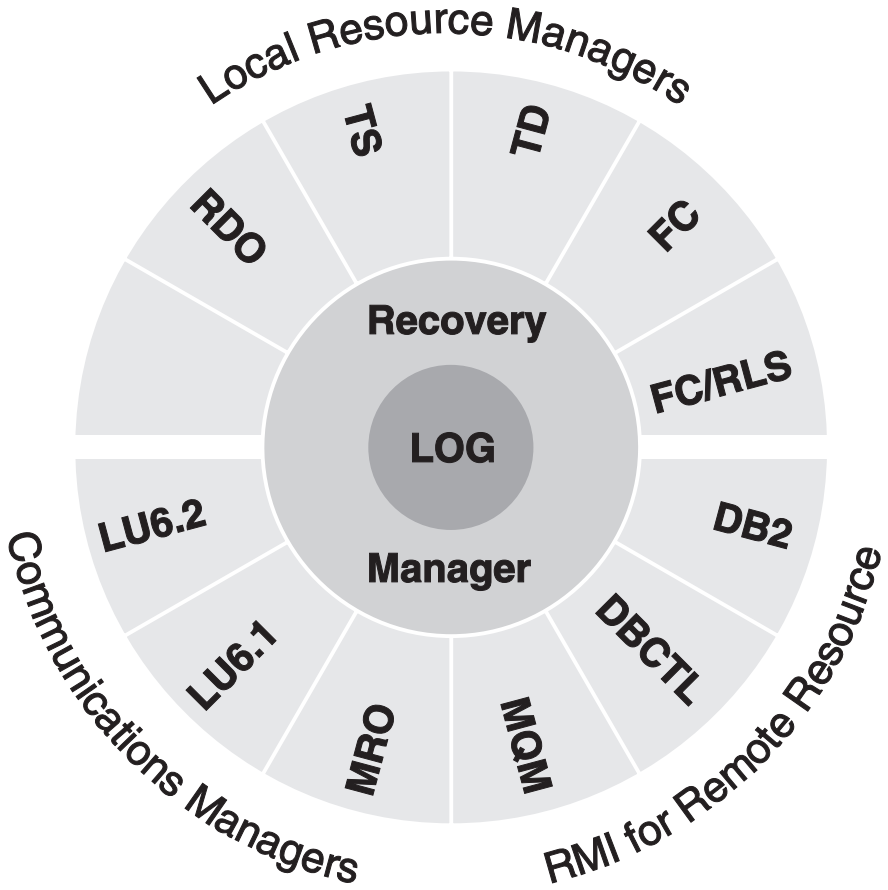


Figure 3. CICS recovery manager and resources it works with

Managing the state of each unit of work

The CICS recovery manager maintains, for each UOW in a CICS region, a record of the changes of state that occur during its lifetime.

Typical events that cause state changes include:

- Creation of the UOW, with a unique identifier
- Premature termination of the UOW because of transaction failure
- Receipt of a syncpoint request
- Entry into the indoubt period during **two-phase commit** processing (see the *CICS Transaction Server for z/OS Glossary* for a definition of two-phase commit)

- Notification that the resource is not available, requiring temporary suspension (shunting) of the UOW
- Notification that the resource is available, enabling retry of shunted UOWs
- Notification that a connection is reestablished, and can deliver a commit or rollback (backout) decision
- Syncpoint rollback
- Normal termination of the UOW

The identity of a UOW and its state are owned by the CICS recovery manager, and are recorded in storage and on the system log. The system log records are used by the CICS recovery manager during emergency restart to reconstruct the state of the UOWs in progress at the time of the earlier system failure.

The execution of a UOW can be distributed over more than one CICS system in a network of communicating systems.

The CICS recovery manager supports SPI commands that provide information about UOWs.

Coordinating updates to local resources

The recoverable local resources managed by a CICS region are files, temporary storage, and transient data, plus resource definitions for terminals, typeterms, connections, and sessions.

Each local resource manager can write UOW-related log records to the local system log, which the CICS recovery manager may subsequently be required to re-present to the resource manager during recovery from failure.

To enable the CICS recovery manager to deliver log records to each resource manager as required, the CICS recovery manager adds additional information when the log records are created. Therefore, all logging by resource managers to the system log is performed through the CICS recovery manager.

During syncpoint processing, the CICS recovery manager invokes each local resource manager that has updated recoverable resources within the UOW. The local resource managers then perform the required action. This provides the means of coordinating the actions performed by individual resource managers.

If the commit or backout of a file resource fails (for example, because of an I/O error or the inability of a resource manager to free a lock), the CICS recovery manager takes appropriate action with regard to the failed resource:

- If the failure occurs during commit processing, the UOW is marked as commit-failed and is shunted awaiting resolution of whatever caused the commit failure.
- If the failure occurs during backout processing, the UOW is marked as backout-failed, and is shunted awaiting resolution of whatever caused the backout to fail.

Note that a commit failure can occur during the commit phase of a completed UOW, or the commit phase that takes place after successfully completing backout. (These two phases (or 'directions') of commit processing—commit after normal completion and commit after backout—are sometimes referred to as 'forward commit' and 'backward commit' respectively.) Note also that a UOW can be backout-failed with respect to some resources and commit-failed with respect to

others. This can happen, for example, if two data sets are updated and the UOW has to be backed out, and the following happens:

- One resource backs out successfully
- While committing this successful backout, the commit fails
- The other resource fails to back out

These events leave one data set commit-failed, and the other backout-failed. In this situation, the overall status of the UOW is logged as backout-failed.

During emergency restart following a CICS failure, each UOW and its state is reconstructed from the system log. If any UOW is in the backout-failed or commit-failed state, CICS automatically retries the UOW to complete the backout or commit.

Coordinating updates in distributed units of work

If the execution of a UOW is distributed across more than one system, the CICS recovery manager (or their non-CICS equivalents) in each pair of connected systems ensure that the effects of the distributed UOW are atomic.

Each CICS recovery manager (or its non-CICS equivalent) issues the requests necessary to effect two-phase syncpoint processing to each of the connected systems with which a UOW may be in conversation.

Note: In this context, the non-CICS equivalent of a CICS recovery manager could be the recovery component of a database manager, such as DBCTL or DB2®, or any equivalent function where one of a pair of connected systems is not CICS.

In each connected system in a network, the CICS recovery manager uses interfaces to its local recovery manager connectors (RMCs) to communicate with partner recovery managers. The RMCs are the communication resource managers (IPIC, LU6.2, LU6.1, MRO, and RMI) which have the function of understanding the transport protocols and constructing the flows between the connected systems.

As remote resources are accessed during UOW execution, the CICS recovery manager keeps track of data describing the status of its end of the conversation with that RMC. The CICS recovery manager also assumes responsibility for the coordination of two-phase syncpoint processing for the RMC.

Managing indoubt units of work

During the syncpoint phases, for each RMC, the CICS recovery manager records the changes in the status of the conversation, and also writes, on behalf of the RMC, equivalent information to the system log.

If a session fails at any time during the running of a UOW, it is the RMC responsibility to notify the CICS recovery manager, which takes appropriate action with regard to the unit of work as a whole. If the failure occurs during syncpoint processing, the CICS recovery manager may be in doubt and unable to determine immediately how to complete the UOW. In this case, the CICS recovery manager causes the UOW to be shunted awaiting UOW resolution, which follows notification from its RMC of successful resynchronization on the failed session.

During emergency restart following a CICS failure, each UOW and its state is reconstructed from the system log. If any UOW is in the indoubt state, it remains shunted awaiting resolution.

Resynchronization after system or connection failure

Units of work that fail while in an indoubt state remain shunted until the indoubt state can be resolved following successful resynchronization with the coordinator.

Resynchronization takes place automatically when communications are next established between subordinate and coordinator. Any decisions held by the coordinator are passed to the subordinate, and indoubt units of work complete normally. If a subordinate has meanwhile taken a unilateral decision following the loss of communication, this decision is compared with that taken by the coordinator, and messages report any discrepancy.

For an explanation and illustration of the roles played by subordinate and coordinator CICS regions, and for information about recovery and resynchronization of distributed units of work generally, see the *CICS Intercommunication Guide*.

CICS system log

CICS system log data is written to two MVS system logger log streams, the primary log stream and secondary log stream, which together form a single logical log stream.

The system log is the only place where CICS records information for use when backing out transactions, either dynamically or during emergency restart processing. CICS automatically connects to its system log stream during initialization, unless you have specified a journal model definition that defines the system log as DUMMY (in which case CICS can perform only an initial start).

The integrity of the system log is critical in enabling CICS to perform recovery. If any of the components involved with the system log—the CICS recovery manager, the CICS log manager, or the MVS system logger—experience problems with the system log, it might be impossible for CICS to perform successfully recovery processing. For more information about errors affecting the system log, see "Effect of problems with the system log" on page 33.

The *CICS System Definition Guide* tells you more about CICS system log streams, and how you can use journal model definitions to map the CICS journal names for the primary system log stream (DFHLOG) and the secondary system log stream (DFHSHUNT) to specific log stream names. If you don't specify journal model definitions, CICS uses default log stream names.

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. You cannot use the system log for forward recovery information, or for terminal control or file control autojournaling.

Your application programs can write user-defined recovery records to the system log using EXEC CICS WRITE JOURNALNAME commands. Any user-written log records to support your own recovery processes are made available to global user exit programs enabled at the XRCINPT exit point.

CICS also writes "backout-failed" records to the system log if a failure occurs in backout processing of a VSAM data set during dynamic backout or emergency restart backout.

Records on the system log are used for cold, warm, and emergency restarts of a CICS region. The only type of start for which the system log records are *not* used is an initial start.

System activity keypoints

The recovery manager controls the recording of keypoint information, and the delivery of the information to the various resource managers at emergency restart.

The recovery manager provides the support that enables activity keypoint information to be recorded at frequent intervals on the system log. You specify the activity keypoint frequency on the **AKPFREQ** system initialization parameter. See the *CICS System Definition Guide* for details. Activity keypoint information is of two types:

1. A list of all the UOWs currently active in the system
2. Image-copy type information allowing the current contents of a particular resource to be rebuilt

During an initial phase of CICS restart, recovery manager uses this information, together with UOW-related log records, to restore the CICS system to its state at the time of the previous shutdown. This is done on a single backward scan of the system log.

Frequency of taking activity keypoints: You are strongly recommended to specify a nonzero activity keypoint frequency. Choose an activity keypoint frequency that is suitable for the size of your system log stream. Note that writing activity keypoints at short intervals improves restart times, but at the expense of extra processing during normal running.

The following additional actions are taken for files accessed in non-RLS mode that use backup while open (BWO):

- Tie-up records are recorded on the forward recovery log stream. A tie-up record 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 does not need to be used.

Forward recovery logs

CICS writes VSAM forward recovery logs to a general log stream defined to the MVS system logger. You can merge forward recovery log data for more than one VSAM data set to the same log stream, or you can dedicate a forward recovery log stream to a single data set.

See "Defining forward recovery log streams" on page 116 for information about the use of forward recovery log streams.

User journals and automatic journaling

User journals and autojournals are written to a general log stream defined to the MVS system logger.

- User journaling is entirely under your application programs' control. You write records for your own purpose using EXEC CICS WRITE JOURNALNAME commands. See "Flushing journal buffers" on page 28 for information about CICS shutdown considerations.
- Automatic journaling means that CICS automatically writes records to a log stream, referenced by the journal name specified in a journal model definition, as a result of:
  - Records read from or written to files. These records represent data that has been read, or data that has been read for update, or data that has been written, or records to indicate the completion of a write, and so on, depending on what types of request you selected for autojournaling.

    You specify that you want autojournaling for VSAM files using the autojournaling options on the file resource definition in the CSD. For BDAM files, you specify the options on a file entry in the file control table.
  - Input or output messages from terminals accessed through VTAM.

    You specify that you want terminal control autojournaling on the JOURNAL option of the profile resource definition referenced by your transaction definitions. These messages could be used to create audit trails.

  Automatic journaling is used for user-defined purposes; for example, for an audit trail. Automatic journaling is not used for CICS recovery purposes.

Chapter 3. Shutdown and restart recovery

CICS can shut down normally or abnormally and this affects the way that CICS restarts after it shuts down.

CICS can stop executing as a result of:

- A normal (warm) shutdown initiated by a CEMT, or EXEC CICS, PERFORM SHUT command
- An immediate shutdown initiated by a CEMT, or EXEC CICS, PERFORM SHUT IMMEDIATE command
- An abnormal shutdown caused by a CICS system module encountering an irrecoverable error
- An abnormal shutdown initiated by a request from the operating system (arising, for example, from a program check or system abend)
- A machine check or power failure

Normal shutdown processing

Normal shutdown is initiated by issuing a CEMT PERFORM SHUTDOWN command, or by an application program issuing an EXEC CICS PERFORM SHUTDOWN command. It takes place in three quiesce stages, as follows:

First quiesce stage

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

The following activities are performed:

- CICS invokes the **shutdown assist transaction** specified on the SDTRAN system initialization parameter or on the shutdown command.

  Because all user tasks must terminate during the first quiesce stage, it is possible that shutdown could be unacceptably delayed by long-running tasks (such as conversational transactions). The purpose of the shutdown assist transaction is to allow as many tasks as possible to commit or back out cleanly, while ensuring that shutdown completes within a reasonable time.

  CICS obtains the name of the shutdown assist transaction as follows:

  1. If SDTRAN(*tranid*) is specified on the PERFORM SHUTDOWN command, or as a system initialization parameter, CICS invokes that *tranid*.
  2. If NOSDTRAN is specified on the PERFORM SHUTDOWN command, or as a system initialization parameter, CICS does not start a shutdown transaction. Without a shutdown assist transaction, all tasks that are already running are allowed to complete.
  3. If the SDTRAN (or NOSDTRAN) options are omitted from the PERFORM SHUTDOWN command, and omitted from the system initialization parameters, CICS invokes the default shutdown assist transaction, CESD, which runs the CICS-supplied program DFHCESD.

  The SDTRAN option specified on the PERFORM SHUT command overrides any SDTRAN option specified as a system initialization parameter.

- The DFHCESD program started by the CICS-supplied transaction, CESD, attempts to purge and back out long-running tasks using increasingly stronger methods (see "The shutdown assist transaction" on page 30).
- Tasks that are automatically initiated are run—if they start before the second quiesce stage.
- Any programs listed in the first part of the shutdown program list table (PLT) are run sequentially. (The shutdown PLT suffix is specified in the PLTSD system initialization parameter, which can 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 resource definition. 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 can 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, CLR1, CLR2, CLQ2, CLS1, CLS2, CSAC, CSTE, and CSNE.
- Finally, at the end of this stage and before the second stage of shutdown, CICS unbinds all the VTAM terminals and devices.

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. If the CICS-supplied shutdown transaction CESD is used, this stage does not wait indefinitely for all user tasks to complete.

Second quiesce stage

During the second quiesce stage of shutdown:

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

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

Third quiesce stage

During the third quiesce stage of shutdown:

- CICS closes all files that are defined to CICS file control. However, CICS does not catalog the files as UNENABLED; they can then be opened implicitly by the first reference after a subsequent CICS restart.

  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.
- All extrapartition TD queues are closed.
- CICS writes statistics to the system management facility (SMF) data set.
- CICS recovery manager sets the type-of-restart indicator in its domain state record in the global catalog to "warm-start-possible". If you specify START=AUTO when you next initialize the CICS region, CICS uses the status of

this indicator to determine the type of startup it is to perform. See "How the state of the CICS region is reconstructed" on page 34.

- CICS writes warm keypoint records to:
  - The global catalog for terminal control and profiles
  - The CICS system log for all other resources.

  See "Warm keypoints."
- CICS deletes all completed units of work (log tail deletion), leaving only shunted units of work and the warm keypoint.

  Note: Specifying no activity keypointing (AKPFREQ=0) only suppresses log tail deletion while CICS is running, not at shutdown. CICS always performs log cleanup at shutdown unless you specify RETPD=dddd on the MVS definition of the system log. See "Activity keypointing" on page 112 for more information.
- CICS stops executing.

Warm keypoints

The CICS-provided warm keypoint program (DFHWKP) writes a warm keypoint to the global catalog, for terminal control and profile resources only, during the third quiesce stage of shutdown processing when all system activity is quiesced.

The remainder of the warm keypoint information, for all other resources, is written to the CICS system log stream, under the control of the CICS recovery manager. This system log warm keypoint is written by the activity keypoint program as a special form of activity keypoint that contains information relating to shutdown.

The warm keypoints contain information needed to restore the CICS environment during a subsequent warm or emergency restart. Thus CICS needs both the global catalog and the system log to perform a restart. If you run CICS with a system log that is defined by a journal model specifying TYPE(DUMMY), you cannot restart CICS with START=AUTO following a normal shutdown, or with START=COLD.

Shunted units of work at shutdown

If there are shunted units of work of any kind at shutdown, CICS issues message DFHRM0203.

This message displays the numbers of indoubt, backout-failed, and commit-failed units of work held in the CICS region's system log at the time of the normal shutdown. It is issued only if there is at least one such UOW. If there are no shunted units of work, CICS issues message DFHRM0204.

DFHRM0203 is an important message that should be logged, and should be taken note of when you next restart the CICS region. For example, if you receive DFHRM0203 indicating that there is outstanding work waiting to be completed, you should not perform a cold or initial start of the CICS region. You are recommended to always restart CICS with START=AUTO, and especially after message DFHRM0203, otherwise recovery data is lost.

See Chapter 4, "CICS cold start," on page 45 for information about a cold start if CICS has issued message DFHRM0203 at the previous shutdown.

Flushing journal buffers

During a successful normal shutdown, CICS calls the log manager domain to flush all journal buffers, ensuring that all journal records are written to their corresponding MVS system logger log streams.

During an immediate shutdown, the call to the log manager domain is bypassed and journal records are not flushed. This also applies to an immediate shutdown that is initiated by the shutdown-assist transaction because a normal shutdown has stalled. Therefore, any user journal records in a log manager buffer at the time of an immediate shutdown are lost. This does not affect CICS system data integrity. The system log and forward recovery logs are always synchronized with regard to I/O and unit of work activity. If user journal data is important, you should take appropriate steps to ensure that journal buffers are flushed at shutdown.

These situations and possible solutions are summarized as follows:

- In a controlled shutdown that completes normally, CICS ensures that user journals are flushed.
- In a controlled shutdown that is forced into an immediate shutdown by a shutdown-assist transaction, CICS does not flush buffers. To avoid the potential loss of journal records in this case, you can provide a PLTSD program that issues a **SET JOURNAL FLUSH** command to ensure that log manager buffers are written to the corresponding log streams. PLTSD programs are invoked before an immediate shutdown is initiated by the shutdown-assist transaction.
- In an uncontrolled shutdown explicitly requested with the **SHUT IMMEDIATE** command, CICS does not flush buffers. To avoid the potential loss of journal records in this case, you can issue an **EXEC CICS WAIT JOURNALNAME** command at appropriate points in the application program, or immediately before returning control to CICS. (Alternatively, you could specify the WAIT option on the **WRITE JOURNALNAME** command.) See the description of the command in the *CICS Application Programming Reference* for information about the journaling WAIT option.

Immediate shutdown processing (PERFORM SHUTDOWN IMMEDIATE)

As a general rule when terminating CICS, you are recommended to use a normal shutdown with a shutdown assist transaction, specifying either your own or the CICS-supplied default, CESD.

PERFORM IMMEDIATE not recommended

You should resort to using an immediate shutdown only if you have a special reason for doing so. For instance, you might need to stop and restart CICS during a particularly busy period, when the slightly faster immediate shutdown may be of benefit. Also, you can use VTAM persistent sessions support with an immediate shutdown.

You initiate an immediate shutdown by a CEMT, or EXEC CICS, PERFORM SHUTDOWN IMMEDIATE command. Immediate shutdown is different from a normal shutdown in a number of important ways:

1. If the shutdown assist transaction is not run (that is, the SDTRAN system initialization parameter specifies NO, or the PERFORM SHUTDOWN command specifies NOSDTRAN), user tasks are not guaranteed to complete. This can lead to an unacceptable number of units of work being shunted, with locks being retained.

2. If the default shutdown assist transaction CESD is run, it allows as many tasks as possible to commit or back out cleanly, but within a shorter time than that allowed on a normal shutdown. See "The shutdown assist transaction" on page 30 for more information about CESD, which runs the CICS-supplied program DFHCESD.
3. None of the programs listed in the shutdown PLT is executed.
4. CICS does **not** write a warm keypoint or a warm-start-possible indicator to the global catalog.
5. CICS does not close files managed by file control. It is left to VSAM to close the files when VSAM is notified by MVS that the address space is terminating. This form of closing files means that a VSAM VERIFY is needed on the next open of the files closed in this way, but this is done automatically.
6. VTAM sessions wait for the restarted region to initialize or until the expiry of the interval specified in the PSDINT system initialization parameter, whichever is earlier.

The next initialization of CICS *must* be an emergency restart, in order to preserve data integrity. An emergency restart is ensured if the next initialization of CICS specifies START=AUTO. This is because the recovery manager's type-of-restart indicator is set to "emergency-restart-needed" during initialization and is not reset in the event of an immediate or uncontrolled shutdown. See "How the state of the CICS region is reconstructed" on page 34.

Note: A PERFORM SHUTDOWN IMMEDIATE command can be issued, by the operator or by the shutdown assist transaction, while a normal or immediate shutdown is already in progress. If this happens, the shutdown assist transaction is not restarted; the effect is to force an immediate shutdown with no shutdown assist transaction.

If the original PERFORM SHUTDOWN request specified a normal shutdown, and the restart manager (ARM) was active, CICS is restarted (because CICS will not de-register from the automatic restart manager until the second quiesce stage of shutdown has completed).

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 operating system abends and program checks" on page 94.)

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 (unless system dumps are suppressed by the DUMP=NO system initialization parameter).
- Does not close any open files. It is left to VSAM to close the files when VSAM is notified by MVS that the address space is terminating. This form of closing files means that a VSAM VERIFY is needed on the next open of the files closed in this way, but this is done automatically.

The next initialization of CICS *must* be an emergency restart, in order to preserve data integrity. An emergency restart is ensured if the next initialization of CICS specifies START=AUTO. This is because the recovery manager's type-of-restart indicator is set to "emergency-restart-needed" during initialization, and is not reset in the event of an immediate or uncontrolled shutdown.

Uncontrolled termination

An uncontrolled shutdown of CICS can be caused by a power failure, machine check, or 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 ensured if the next initialization of CICS specifies START=AUTO. This is because the recovery manager's type-of-restart indicator is set to "emergency-restart-needed" during initialization, and is not reset in the event of an immediate or uncontrolled shutdown.

The shutdown assist transaction

On an immediate shutdown, CICS does not allow running tasks to finish. A backout is not performed until an emergency restart.

This can cause an unacceptable number of units of work to be shunted, with locks being retained. On the other hand, on a normal shutdown, CICS waits indefinitely for running transactions to finish, which can delay shutdown to a degree that is unacceptable. The CICS shutdown assist transaction improves both these forms of shutdown and, to a large degree, removes the need for an immediate shutdown.

The operation of CESD, for both normal and immediate shutdowns, takes place over a number of stages. CESD controls these stages by sampling the number of tasks present in the system, and proceeds to the next stage if the number of in-flight tasks is not reducing quickly enough.

The stages of a normal shutdown CESD are as follows:

- In the initial stage of assisted shutdown, CESD attempts to complete a normal shutdown in a reasonable time.
- After a time allowed for transactions to finish normally (that is, after the number of tasks has not reduced over a period of eight samples), CESD proceeds to issue a normal purge for each remaining task. The transaction dump data set is closed in this stage.
- If there are still transactions running after a further eight samples (except when persistent sessions support is being used), VTAM is force-closed and IRC is closed immediately.
- Finally, if there are still transactions running, CICS shuts down abnormally, leaving details of the remaining in-flight transactions on the system log to be dealt with during an emergency restart.

The operation of CESD is quicker for an immediate shutdown, with the number of tasks in the system being sampled only four times instead of eight.

You are recommended always to use the CESD shutdown-assist transaction when shutting down your CICS regions. You can use the DFHCESD program "as is", or use the supplied source code as the basis for your own customized version (CICS supplies versions in assembler, COBOL, and PL/I). For more information about the operation of the CICS-supplied shutdown assist program, see the *CICS Operations and Utilities Guide*.

Cataloging CICS resources

CICS uses a global catalog data set (DFHGCD) and a local catalog data set (DFHLCD) to store information that is passed from one execution of CICS, through a shutdown, to the next execution of CICS.

This information is used for warm and emergency restarts, and to a lesser extent for cold starts. If the global catalog fails (for reasons other than filling the available space), the recovery manager control record is lost. Without this, it is impossible to perform a warm, emergency, or cold start, and the only possibility is then an initial start. For example, if the failure is due to an I/O error, you cannot restart CICS.

Usually, if the global catalog fills, CICS abnormally terminates, in which case you could define more space and attempt an emergency restart.

Consider putting the catalog data sets on the most reliable storage available—RAID or dual-copy devices—to ensure maximum protection of the data. Taking ordinary copies is not recommended because of the risk of getting out of step with the system log.

From a restart point of view, the system log and the CICS catalog (both data sets) form one logical set of data, and all of them are required for a restart.

The *CICS System Definition Guide* tells you how to create and initialize these CICS catalog data sets.

Global catalog

The global catalog contains information that CICS requires on a restart.

CICS uses the global catalog to store the following information:

- The names of the system log streams.
- Copies of tables of installed resource definitions, and related information, for the following:
  - Transactions and transaction classes
  - DB2 resource definitions
  - Programs, mapsets, and partitionsets (including autoinstalled programs, subject to the operand you specify on the PGAICTLG system initialization parameter)
  - Terminals and typeterms (for predefined and autoinstalled resources)
  - Autoinstall terminal models
  - Profiles
  - Connections, sessions, and partners
  - BDAM and VSAM files (including data tables) and
    - VSAM LSR pool share control blocks
    - Data set names and data set name blocks

- File control recovery blocks (only if a SHCDS NONRLSUPDATEPERMITTED command has been used).

– Transient data queue definitions
– Dump table information
– Interval control elements and automatic initiate descriptors at shutdown
– APPC connection information so that relevant values can be restored during a persistent sessions restart
– Logname information used for communications resynchronization
– Monitoring options in force at shutdown
– Statistics interval collection options in force at shutdown
– Journal model and journal name definitions
– Enqueue model definitions
– Temporary storage model definitions
– URIMAP definitions and virtual hosts for CICS Web support.

Most resource managers update the catalog whenever they make a change to their table entries. Terminal and profile resource definitions are exceptions (see the next list item about the catalog warm keypoint). Because of the typical volume of changes, terminal control does not update the catalog, except when:

– Running a VTAM query against a terminal
– A generic connection has bound to a remote system
– Installing a terminal
– Deleting a terminal.

- A partial warm keypoint at normal shutdown. This keypoint contains an image copy of the TCT and profile resource definitions at shutdown for use during a warm restart.

  Note: The image copy of the TCT includes all the permanent devices installed by explicit resource definitions. Except for some autoinstalled APPC connections, it does *not* include autoinstalled devices. Autoinstalled terminal resources are cataloged initially, in case they need to be recovered during an emergency restart, but only if the AIRDELAY system initialization parameter specifies a nonzero value. Therefore, apart from the APPC exceptions mentioned above, autoinstalled devices are excluded from the warm keypoint, and are thus not recovered on a warm start.
- Statistics options.
- Monitoring options.
- The recovery manager's control record, which includes the type-of-restart indicator (see "How the state of the CICS region is reconstructed" on page 34).

All this information is essential for a successful restart following any kind of shutdown.

Local catalog

The CICS local catalog data set represents just one part of the CICS catalog, which is implemented as two physical data sets.

The two data sets are logically one set of cataloged data managed by the CICS catalog domain. Although minor in terms of the volume of information recorded on it, the local catalog is of equal importance with the global catalog, and the data should be equally protected when restarts are performed.

If you ever need to redefine and reinitialize the CICS local catalog, you should also reinitialize the global catalog. After reinitializing both catalog data sets, you must perform an initial start.

Shutdown initiated by CICS log manager

The CICS log manager initiates a shutdown of the region if it encounters an error in the system log that indicates previously logged data has been lost.

In addition to initiating the shutdown, the log manager informs the recovery manager of the failure, which causes the recovery manager to set the type-of-restart indicator to "no-restart-possible" and to issue message DFHRM0144. The result is that recovery during a subsequent restart is not possible and you can perform only an initial start of the region. To do this you are recommended to run the recovery manager utility program (DFHRMUTL) to force an initial start, using the SET_AUTO_START=AUTOINIT option.

During shutdown processing, existing transactions are given the chance to complete their processing. However, no further data is written to the system log. This strategy ensures that the minimum number of units of work are impacted by the failure of the system log. This is because:

- If a unit of work does not attempt to backout its resource updates, and completes successfully, it is unaffected by the failure.
- If a unit of work does attempt to backout, it cannot rely on the necessary log records being available, and therefore it is permanently suspended.

Therefore, when the system has completed the log manager-initiated shutdown all (or most) units of work will have completed normally during this period and if there are no backout attempts, data integrity is not compromised.

Effect of problems with the system log

A key value of CICS is its ability to implement its transactional recovery commitments and thus safeguard the integrity of recoverable data updated by CICS applications.

This ability relies upon logging before-images and other information to the system log. However, the system log itself might suffer software or hardware related problems, including failures in the CICS recovery manager, the CICS logger domain, or the MVS system logger. Although problems with these components are unlikely, you must understand the actions to take to minimize the impact of such problems.

If the CICS log manager detects an error in the system log that indicates previously logged data has been lost, it initiates a shutdown of the region. This action minimizes the number of transactions that fail after a problem with the log is detected and therefore minimizes the data integrity exposure.

Any problem with the system log that indicates that it might not be able to access all the data previously logged invalidates the log. In this case, you can perform only a diagnostic run or an initial start of the region to which the system log belongs.

The reason that a system log is completely invalidated by these kinds of error is that CICS can no longer rely on the data it previously logged being available for recovery processing. For example, the last records logged might be unavailable,

and therefore recovery of the most recent units of work cannot be carried out. However, data might be missing from any part of the system log and CICS cannot identify what is missing. CICS cannot examine the log and determine exactly what data is missing, because the log data might appear consistent in itself even when CICS has detected that some data is missing.

These are the messages that CICS issues as it reads the log during a warm or emergency start and that can help you identify which units of work were recovered:

DFHRM0402
This message is issued for each unit of work when it is first encountered on the log.

DFHRM0403 and DFHRM0404
One of these messages is issued for each unit of work when its context is found. The message reports the state of the unit of work.

DFHRM0405
This message is issued when a complete keypoint has been recovered from the log.

If you see that message DFHRM0402 is issued for a unit of work, and it is matched by message DFHRM0403 or DFHRM0404, you can be sure of the state of the unit of work. If you see message DFHRM0405, you can use the preceding messages to determine which units of work are incomplete, and you can also be sure that none of the units of work is completely missing.

Another class of problem with the system log is one that does not indicate any loss of previously logged data; for example, access to the logstream was lost due to termination of the MVS system logger address space. This class of problem causes an immediate termination of CICS because a subsequent emergency restart will probably succeed when the cause of the problem has been resolved.

For information about how to deal with system log problems, see the *CICS Problem Determination Guide*.

How the state of the CICS region is reconstructed

CICS recovery manager uses the type-of-restart indicator in its domain state record from the global catalog to determine which type of restart it is to perform.

This indicator operates as follows:

- Before the end of initialization, on all types of startup, CICS sets the indicator in the control record to “emergency restart needed”.
- If CICS terminates normally, this indicator is changed to “warm start possible”.
- If CICS terminates abnormally because the system log has been corrupted and is no longer usable, this indicator is changed to “no restart”. After fixing the system log, perform an initial start of the failed CICS region.
- 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.

Overriding the type of start indicator

The operation of the recovery manager's control record can be modified by running the recovery manager utility program, DFHRMUTL.

About this task

This can set an autostart record that determines the type of start CICS is to perform, effectively overriding the type of start indicator in the control record. See the *CICS Operations and Utilities Guide* for information about using DFHRMUTL to modify the type of start performed by START=AUTO.

Warm restart

If you shut down a CICS region normally, CICS restarts with a warm restart if you specify START=AUTO. For a warm start to succeed, CICS needs the information stored in the CICS catalogs at the previous shutdown, and the information stored in the system log.

In a warm restart, CICS:

1. Restores the state of the CICS region to the state it was in at completion of the normal shutdown. All CICS resource definitions are restored from the global catalog, and the **GRPLIST**, **FCT**, and **CSD** system initialization parameters are ignored.

   CICS also uses information from the warm keypoint in the system log.
2. Reconnects to the system log.
3. Retries any backout-failed and commit-failed units of work.
4. Rebuilds indoubt-failed units of work.

For more information about the warm restart process, see Chapter 5, “CICS warm restart,” on page 53.

Emergency restart

If a CICS region fails, CICS restarts with an emergency restart if you specify START=AUTO. An emergency restart is similar to a warm start but with additional recovery processing for example, to back out any transactions that were in-flight at the time of failure, and thus free any locks protecting resources.

If the failed CICS region was running with VSAM record-level sharing, SMSVSAM converts into retained locks any active exclusive locks held by the failed system, pending the CICS restart. This means that the records are protected from being updated by any other CICS region in the sysplex. Retained locks also ensure that other regions trying to access the protected records do not wait on the locks until the failed region restarts. See the *CICS Application Programming Guide* for information about active and retained locks.

For non-RLS data sets (including BDAM data sets), any locks (ENQUEUES) that were held before the CICS failure are reacquired.

Initialization during emergency restart

Most of CICS initialization following an emergency restart is the same as for a warm restart, and CICS uses the catalogs and the system log to restore the state of the CICS region. Then, after the normal initialization process, emergency restart

performs the recovery process for work that was in-flight when the previous run of CICS was abnormally terminated.

Recovery of data during an emergency restart

During the final stage of emergency restart, the recovery manager uses the system log data to drive backout processing for any units of work that were in-flight at the time of the failure. The backout of units of work during emergency restart is the same as a dynamic backout; there is no distinction between the backout that takes place at emergency restart and that which takes place at any other time.

The recovery manager also drives:

- The backout processing for any units of work that were in a backout-failed state at the time of the CICS failure.
- The commit processing for any units of work that were in a commit-failed state at the time of the CICS failure.
- The commit processing for units of work that had not completed commit at the time of failure (resource definition recovery, for example).

The recovery manager drives these backout and commit processes because the condition that caused them to fail may be resolved by the time CICS restarts. If the condition that caused a failure has not been resolved, the unit of work remains in backout- or commit-failed state. See "Backout-failed recovery" on page 79 and "Commit-failed recovery" on page 83 for more information.

For more information about the emergency restart process, see Chapter 6, "CICS emergency restart," on page 61.

Cold start

On a cold start, CICS reconstructs the state of the region from the previous run for remote resources only. For all resources, the region is built from resource definitions specified on the **GRPLIST** system initialization parameter and those resources defined in control tables.

The following is a summary of how CICS uses information stored in the global catalog and the system log on a cold start:

- CICS preserves, in both the global catalog and the system log, all the information relating to distributed units of work for partners linked by:
  - APPC
  - MRO connections to regions running under CICS Transaction Server
  - The resource manager interface (RMI); for example, to DB2 and DBCTL.
- CICS does not preserve any information in the global catalog or the system log that relates to local units of work.

Generally, to perform a cold start you specify START=COLD, but CICS can also force a cold start in some circumstances when START=AUTO is specified. See the *CICS System Definition Guide* for details of the effect of the **START** parameter in conjunction with various states of the global catalog and the system log.

An initial start of CICS

If you want to initialize a CICS region without reference to the global catalog from a previous run, perform an initial start.

You can do this by specifying START=INITIAL as a system initialization parameter, or by running the recovery manager's utility program (DFHRMUTL) to override the type of start indicator to force an initial start.

See the *CICS Operations and Utilities Guide* for information about the DFHRMUTL utility program.

Dynamic RLS restart

If a CICS region is connected to an SMSVSAM server when the server fails, CICS continues running, and recovers using a process known as dynamic RLS restart. An SMSVSAM server failure does not cause CICS to fail, and does not affect any resource other than data sets opened in RLS mode.

When an SMSVSAM server fails, any locks for which it was responsible are converted to retained locks by another SMSVSAM server within the sysplex, thus preventing access to the records until the situation has been recovered. CICS detects that the SMSVSAM server has failed the next time it tries to perform an RLS access after the failure, and issues message DFHFC0153. The CICS regions that were using the failed SMSVSAM server defer in-flight transactions by abending units of work that attempt to access RLS, and shunt them when the backouts fail with "RLS is disabled" responses. If a unit of work is attempting to commit its changes and release RLS locks, commit failure processing is invoked when CICS first detects that the SMSVSAM server is not available (see "Commit-failed recovery" on page 83).

RLS mode open requests and RLS mode record access requests issued by new units of work receive error responses from VSAM when the server has failed. The SMSVSAM server normally restarts itself without any manual intervention. After the SMSVSAM server has restarted, it uses the MVS event notification facility (ENF) to notify all the CICS regions within its MVS image that the SMSVSAM server is available again.

CICS performs a dynamic equivalent of emergency restart for the RLS component, and drives backout of the deferred work.

Recovery after the failure of an SMSVSAM server is usually performed automatically by CICS. CICS retries any backout-failed and commit-failed units of work. In addition to retrying those failed as a result of the SMSVSAM server failure, this also provides an opportunity to retry any backout failures for which the cause has now been resolved. Manual intervention is required only if there are units of work which, due to the timing of their failure, were not retried when CICS received the ENF signal. This situation is extremely unlikely, and such units of work can be detected using the **INQUIRE UOWDSNFAIL** command.

Note that an SMSVSAM server failure causes commit-failed or backout-failed units of work only in the CICS regions registered with the server in the same MVS image. Transactions running in CICS regions in other MVS images within the sysplex are affected only to the extent that they receive LOCKED responses if they try to access records protected by retained locks owned by any CICS regions that were using the failed SMSVSAM server.

Recovery with VTAM persistent sessions

With VTAM persistent sessions support, if CICS fails or undergoes immediate shutdown (by means of a **PERFORM SHUTDOWN IMMEDIATE** command), VTAM holds the CICS LU-LU sessions in recovery-pending state, and they can be recovered during startup by a newly starting CICS region. With multinode persistent sessions support, sessions can also be recovered if VTAM or z/OS® fails in a sysplex.

The CICS system initialization parameter **PSTYPE** specifies the type of persistent sessions support for a CICS region:

SNPS, single-node persistent sessions
Persistent sessions support is available, so that VTAM sessions can be recovered after a CICS failure and restart. This setting is the default.

MNPS, multinode persistent sessions
In addition to the SNPS support, VTAM sessions can also be recovered after a VTAM or z/OS failure in a sysplex.

NOPS, no persistent sessions
Persistent sessions support is not required for the CICS region. For example, a CICS region that is used only for development or testing might not require persistent sessions.

For single-node persistent sessions support, you require VTAM V3.4.1 or later, which supports persistent LU-LU sessions. CICS Transaction Server for z/OS, Version 4 Release 1 functions with releases of VTAM earlier than V3.4.1, but in the earlier releases sessions are not retained in a bound state if CICS fails. For multinode persistent sessions support, you require VTAM V4.R4 or later, and VTAM must be in a Parallel Sysplex® with a coupling facility. The *VTAM Network Implementation Guide* explains the exact VTAM configuration requirements for multinode persistent sessions support.

CICS support of persistent sessions includes the support of all LU-LU sessions, except LU0 pipeline and LU6.1 sessions. With multinode persistent sessions support, if VTAM or z/OS fails, LU62 synclevel 1 sessions are restored, but LU62 synclevel 2 sessions are not restored.

Running with persistent sessions support

When you specify SNPS or MNPS for the **PSTYPE** system initialization parameter so that VTAM persistent sessions support is in use for a CICS region, the time specified by the **PSDINT** system initialization parameter for the region determines how long the sessions are retained.

If a CICS, VTAM, or z/OS failure occurs, if a connection to VTAM is reestablished within this time, CICS can use the retained sessions immediately; there is no need for network flows to rebind them.

Make sure that you set a nonzero value for the persistent sessions delay interval, so that sessions are retained. The default is zero, which means that persistent sessions support is available if you have specified SNPS or MNPS for **PSTYPE**, but it is not being exploited.

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

During an emergency restart of CICS, CICS restores those sessions pending recovery from the CICS global catalog and the CICS system log to an in-session state. This process of persistent sessions recovery takes place when CICS opens its VTAM ACB. With multinode persistent sessions support, if VTAM or z/OS fails, sessions are restored when CICS reopens its VTAM ACB, either automatically by the COVR transaction, or by a CEMT or **EXEC CICS SET VTAM OPEN** command. Although sessions are recovered, any transactions inflight at the time of the failure are abended and not recovered.

When a terminal user enters data during persistent sessions recovery, CICS appears to hang. The screen that was displayed at the time of the failure remains on display until persistent sessions recovery is complete. You can use options on the TYPETERM and SESSIONS resource definitions for the CICS region to customize CICS so that either a successful recovery can be transparent to terminal users, or terminal users can be notified of the recovery, allowing them to take the appropriate actions.

If APPC sessions are active at the time of the CICS, VTAM or z/OS failure, persistent sessions recovery appears to APPC partners as CICS hanging. VTAM saves requests issued by the APPC partner, and passes them to CICS when recovery is complete. When CICS reestablishes a connection with VTAM, recovery of terminal sessions is determined by the settings for the PSRECOVERY option of the CONNECTION resource definition and the RECOVOPTION option of the SESSIONS resource definition. You must set the PSRECOVERY option of the CONNECTION resource definition to the default value SYSDEFAULT for sessions to be recovered. The alternative, NONE, means that no sessions are recovered. If you have selected the appropriate recovery options and the APPC sessions are in the correct state, CICS performs an **ISSUE ABEND** to inform the partner that the current conversation has been abnormally ended.

If CICS has persistent verification defined, the sign-on is not active under persistent sessions until the first input is received by CICS from the terminal.

The *CICS Resource Definition Guide* describes the steps required to define persistent sessions support for a CICS region.

Situations in which sessions are not reestablished

When VTAM persistent sessions support is in use for a CICS region, CICS does not always reestablish sessions that are being held by VTAM in a recovery pending state. In the situations listed here, CICS or VTAM unbinds and does not rebind recovery pending sessions.

- If CICS does not restart within the persistent sessions delay interval, as specified by the **PSDINT** system initialization parameter.
- 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 (RECOVOPTION) of the TYPETERM or SESSIONS resource definition set to RELEASESESS, UNCONDREL or NONE.
- If a connection is defined with the persistent sessions recovery option (PSRECOVERY) of the CONNECTION resource definition set to NONE.

- If CICS determines that it cannot recover the session without unbinding and rebinding it.

The result in each case is as if CICS has restarted following a failure without VTAM persistent sessions support.

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

With multinode persistent sessions support, if a VTAM or z/OS 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 reestablished and the terminal is not reused within the AIRDELAY interval, CICS deletes the TCTTE when the AIRDELAY interval expires after the ACB is reopened successfully.

Situations in which VTAM does not retain sessions

When VTAM persistent sessions support is in use for a CICS region, in some circumstances VTAM does not retain LU-LU sessions.

- If you close VTAM with any of the following CICS commands:
  - **SET VTAM FORCECLOSE**
  - **SET VTAM IMMCLOSE**
  - **SET VTAM CLOSED**
- If you close the CICS node with the VTAM command **VARY NET INACT ID**=*applid*.
- If your CICS system performs a normal shutdown, with a **PERFORM SHUTDOWN** command.

If single-node persistent sessions support (SNPS), which is the default, is specified for a CICS region, sessions are not retained after a VTAM or z/OS failure. If multinode persistent sessions support (MNPS) is specified, sessions are retained after a VTAM or z/OS failure.

Running without persistent sessions support

VTAM persistent sessions support is the default for a CICS region, but you might choose to run a CICS region without this support if it is used only for development or testing. Specify NOPS for the **PSTYPE** system initialization parameter to start a CICS region without persistent sessions support. Running without persistent sessions support can enable you to increase the number of CICS regions in an LPAR.

If you have a large number of CICS regions in the same LPAR (around 500), with persistent sessions support available for all the regions, you might reach a z/OS limit on the maximum number of data spaces and be unable to add any more CICS regions. In this situation, when you attempt to start further CICS regions, you see messages IST967I and DFHSI1572, stating that the ALESERV ADD request has failed and the VTAM ACB cannot be opened. However, a region without persistent sessions support does not use a data space and so does not count towards the limit. To obtain a greater number of CICS regions in the LPAR:

1. Identify existing regions that can run without persistent sessions support.
2. Change the **PSTYPE** system initialization parameter for those regions to specify NOPS, and specify a zero value for the **PSDINT** system initialization parameter.
3. Cold start the regions to implement the change.

You can then start further CICS regions with or without persistent sessions support as appropriate, provided that you do not exceed the limit for the number of regions that do have persistent sessions support.

If you specify NOPS (no persistent session support) for the **PSTYPE** system initialization parameter, a zero value is required for the **PSDINT** (persistent session delay interval) system initialization parameter.

When persistent sessions support is not in use, all sessions existing on a CICS system are lost if that CICS system, VTAM, or z/OS fails. In any subsequent restart of CICS, the rebinding of sessions that existed before the failure depends on the AUTOCONNECT option for the terminal. If AUTOCONNECT is specified for a terminal, the user of that terminal waits until the GMTRAN transaction has run before being able to continue working. The user sees the VTAM logon panel followed by the "good morning" message. If AUTOCONNECT is not specified for a terminal, the user of that terminal has no way of knowing (unless told by support staff) when CICS is operational again unless the user tries to log on. In either case, users are disconnected from CICS and need to reestablish a session, or sessions, to regain their working environment.

Part 2. Recovery and restart processes

You can add your own processing to the CICS recovery and restart processes.

This part contains the following sections:

Chapter 4. CICS cold start

This section describes the CICS startup processing specific to a cold start.

It covers the two forms of cold start:

- "Starting CICS with the START=COLD parameter"
- "Starting CICS with the START=INITIAL parameter" on page 50

Starting CICS with the START=COLD parameter

START=COLD performs a dual type of startup, performing a cold start for all local resources while preserving recovery information that relates to remote systems or resource managers connected through the resource manager interface (RMI).

About this task

This ensures the integrity of the CICS region with its partners in a network that manages a distributed workload. You can use a cold start to install resource definitions from the CSD (and from macro control tables). It is normally safe to cold start a CICS region that does not own any local resources (such as a terminal-owning region that performs only transaction routing). For more information about performing a cold start, and when it is safe to do so, see the *CICS Intercommunication Guide*.

If you specify `START=COLD`, CICS either discards or preserves information in the system log and global catalog data set, as follows:

- CICS deletes all cataloged resource definitions in the CICS catalogs and installs definitions either from the CSD or from macro control tables. CICS writes a record of each definition in the global catalog data set as each resource definition is installed.
- Any program LIBRARY definitions that had been dynamically defined will be lost. Only the static DFHRPL concatenation will remain, together with any LIBRARY definitions in the grouplist specified at startup or installed via BAS at startup.
- CICS preserves the recovery manager control record, which contains the CICS logname token used in the previous run. CICS also preserves the log stream name of the system log.
- CICS discards any information from the system log that relates to *local* resources, and resets the system log to begin writing at the start of the primary log stream.

  Note: If CICS detects that there were shunted units of work at the previous shutdown (that is, it had issued message DFHRM0203) CICS issues a warning message, DFHRM0154, to let you know that local recovery data has been lost, and initialization continues. The only way to avoid this loss of data from the system log is *not* to perform a cold start after CICS has issued DFHRM0203.

  If the cold start is being performed following a shutdown that issued message DFHRM0204, CICS issues message DFHRM0156 to confirm that the cold start has not caused any loss of local recovery data.
- CICS releases *all* retained locks:

  - CICS requests the SMSVSAM server, if connected, to release all RLS retained locks.
  - CICS does not rebuild the non-RLS retained locks.
- CICS requests the SMSVSAM server to clear the RLS sharing control status for the region.
- CICS does not restore the dump table, which may contain entries controlling system and transaction dumps.
- CICS preserves resynchronization information about distributed units of work—information regarding unit of work obligations to remote systems, or to non-CICS resource managers (such as DB2) connected through the RMI. For example, the preserved information includes data about the outcome of distributed UOWs that is needed to allow remote systems (or RMI resource managers) to resynchronize their resources.

  Note: The system log information preserved does *not* include before-images of any file control data updated by a distributed unit of work. Any changes made to local file resources are not backed out, and by freeing all locks they are effectively committed. To preserve data integrity, perform a warm or emergency restart using START=AUTO.
- CICS retrieves its logname token from the recovery manager control record for use in the "exchange lognames" process during reconnection to partner systems. Thus, by using the logname token from the previous execution, CICS ensures a warm start of those connections for which there is outstanding resynchronization work.

To perform these actions on a cold start, CICS needs the contents of the catalog data sets and the system log from a previous run.

See the *CICS System Definition Guide* for details of the actions that CICS takes for START=COLD in conjunction with various states of the global catalog and the system log.

The DFHRMUTL utility returns information about the type of previous CICS shutdown which is of use in determining whether a cold restart is possible or not. For further details, see the *CICS Operations and Utilities Guide*.

Files

All previous file control state data, including file resource definitions, is lost.

If RLS support is specified, CICS connects to the SMSVSAM, and when connected requests the server to:

- Release all RLS retained locks
- Clear any "lost locks" status
- Clear any data sets in "non-RLS update permitted" status

For non-RLS files, the CICS enqueue domain does not rebuild the retained locks relating to shunted units of work.

File resource definitions are installed as follows:

VSAM
: Except for the CSD itself, all VSAM file definitions are installed from the CSD. You specify these in groups named in the CSD group list, which you

specify on the GRPLIST system initialization parameter. The CSD file definition is built and installed from the CSD*xxxx* system initialization parameters.

Data tables
As for VSAM file definitions.

BDAM
File definitions are installed from file control table entries, specified by the FCT system initialization parameter.

Attention: If you use the **SHCDS REMOVESUBSYS** command for a CICS region that uses RLS access mode, ensure that you perform a cold start the next time you start the CICS region. The **SHCDS REMOVESUBSYS** command causes SMSVSAM to release all locks held for the region that is the subject of the command, allowing other CICS regions and batch jobs to update records released in this way. If you restart a CICS region with either a warm or emergency restart, after specifying it on a **REMOVESUBSYS** command, you risk losing data integrity.

You are recommended to use the REMOVESUBSYS command only for those CICS regions that you do not intend to run again, and therefore you need to free any retained locks that SMSVSAM might be holding.

Temporary storage

All temporary storage queues from a previous run are lost, including CICS-generated queues (for example, for data passed on START requests).

If the auxiliary temporary storage data set was used on a previous run, CICS opens the data set for update. If CICS finds that the data set is newly initialized, CICS closes it, reopens it in output mode, and formats all the control intervals (CIs) in the primary extent. When formatting is complete, CICS closes the data set and reopens it in update mode. The time taken for this formatting operation depends on the size of the primary extent, but it can add significantly to the time taken to perform a cold start.

Temporary storage data sharing server

Any queues written to a shared temporary storage pool normally persist across a cold start.

Shared TS pools are managed by a temporary storage server, and stored in the coupling facility. Stopping and restarting a TS data sharing server does not affect the contents of the TS pool, unless you clear the coupling facility structure in which the pool resides.

If you want to cause a server to reinitialize its pool, use the MVS SETXCF FORCE command to clean up the structure:

```
SETXCF FORCE,STRUCTURE,STRNAME(DFHXQLS_poolname)
```

The next time you start up the TS server following a SETXCF FORCE command, the server initializes its TS pool in the structure using the server startup parameters specified in the DFHXQMN job.

Transient data

All transient data queues from a previous run are lost.

Transient data resource definitions are installed from Resource groups defined in the CSD, as specified in the CSD group list (named on the GRPLIST system initialization parameter). Any extrapartition TD queues that require opening are opened; that is, any that specify OPEN(INITIAL). All the newly-installed TD queue definitions are written to the global catalog. All TD queues are installed as enabled.CSD definitions are installed later than the macro-defined entries because of the position of CSD group list processing in the initialization process. Any extrapartition TD queues that need to be opened are opened; that is, any that specify OPEN=INITIAL. The TDINTRA system initialization parameter has no effect in a cold start.

Note: If, during the period when CICS is installing the TD queues, an attempt is made to write a record to a CICS-defined queue that has not yet been installed (for example, CSSL), CICS writes the record to the CICS-defined queue CXRF.

Transactions

All transaction and transaction class resource definitions are installed from the CSD, and are cataloged in the global catalog.

Journal names and journal models

All journal model definitions are installed from the CSD, and are cataloged in the global catalog. Journal name definitions (including the system logs DFHLOG and DFHSHUNT) are created using the installed journal models and cataloged in the global catalog.

Note: The CICS log manager retrieves the system log stream name from the global catalog, ensuring that, even on a cold start, CICS uses the same log stream as on a previous run.

LIBRARY resources

All LIBRARY resources from a previous run are lost.

LIBRARY resource definitions are installed from resource groups defined in the CSD, as specified in the CSD group list (named on the **GRPLIST** system initialization parameter).

Programs

All programs, mapsets, and partitionsets are installed from the CSD, and are cataloged in the global catalog.

Start requests (with and without a terminal)

All forms of start request recorded in a warm keypoint (if the previous shutdown was normal) are lost. This applies both to START requests issued by a user application program and to START commands issued internally by CICS in support of basic mapping support (BMS) paging.

Any data associated with START requests is also lost, even if it was stored in a recoverable TS queue.

Resource definitions dynamically installed

Any resource definitions dynamically added to a previous run of CICS are lost in a cold start, unless they are included in the group list specified on the GRPLIST system initialization parameter.

If you define new resource definitions and install them dynamically, ensure the group containing the resources is added to the appropriate group list.

Monitoring and statistics

The initial status of CICS monitoring is determined by the monitoring system initialization parameters (MN and MN*xxxx*).

The initial recording status for CICS statistics is determined by the statistics system initialization parameter (STATRCD). If STATRCD=ON is specified, interval statistics are recorded at the default interval of every three hours.

Terminal control resources

All previous terminal control information stored in the global catalog warm keypoint is lost.

Terminal control resource definitions are installed as follows:

VTAM devices
: All VTAM terminal resource definitions are installed from the CSD. The definitions to be installed are specified in groups named in the CSD group list, which is specified by the GRPLIST system initialization parameter. The resource definitions, of type TERMINAL and TYPETERM, include autoinstall model definitions as well as explicitly defined devices.

Connection, sessions, and profiles
: All connection and sessions definitions are installed from the CSD. The definitions to be installed are specified in groups named in the CSD group list, which is specified by the GRPLIST system initialization parameter. The connections and sessions resource definitions include those used for APPC autoinstall of parallel and single sessions, as well as explicitly defined connections.

TCAM and sequential devices
: All TCAM and sequential (BSAM) device terminal resource definitions are installed from the terminal control table specified by the TCT system initialization parameter. CICS loads the table from the load library defined in the DFHRPL library concatenation. CICS TS for z/OS, Version 4.1

  Note: supports only *remote* TCAM terminals—that is, the only TCAM terminals you can define are those attached to a remote, pre-CICS TS 3.1, terminal-owning region by TCAM/DCB.

  Resource definitions for TCAM and BSAM terminals are not cataloged at install time. They are cataloged only in the terminal control warm keypoint during a normal shutdown.

Committing and cataloging resources installed from the CSD

CICS has two ways of installing and committing terminal resource definitions. Some resource definitions can be installed in groups or individually and are committed at the individual resource level, whereas some VTAM terminal control resource definitions must be installed in groups and are committed in "installable sets".

Single resource install
: All except the resources that are installed in installable sets are committed individually. CICS writes each single resource definition to the global catalog as the resource is installed. If a definition fails, it is not written to the catalog (and therefore is not recovered at a restart).

Installable set install

The following VTAM terminal control resources are committed in installable sets:

- Connections and their associated sessions
- Pipeline terminals—all the terminal definitions sharing the same POOL name

If one definition in an installable set fails, the set fails. However, each installable set is treated independently within its CSD group. If an installable set fails as CICS installs the CSD group, it is removed from the set of successful installs. Logical sets that are not successfully installed do not have catalog records written and are not recovered.

If the install of a resource or of an installable set is successful, CICS writes the resource definitions to the global catalog during commit processing.

Distributed transaction resources

Unlike all other resources in a cold start, CICS preserves any information (units of work) about *distributed* transactions.

This has no effect on units of work that relate only to the local CICS - it applies only to distributed units of work. The CICS recovery manager deals with these preserved units of work when resynchronization with the partner system takes place, just as in a warm or emergency restart.

This is effective only if both the system log stream and the global catalog from the previous run of CICS are available at restart.

See the *CICS Transaction Server for z/OS Installation Guide* for information about recovery of distributed units of work.

Dump table

The dump table that you use for controlling system and transaction dumps is not preserved in a cold start.

If you have built up over a period of time a number of entries in a dump table, which is recorded in the CICS catalog, you have to re-create these entries following a cold start.

Starting CICS with the START=INITIAL parameter

If you specify START=INITIAL, CICS performs an initial start as if you are starting a new region for the first time.

About this task

This initial start of a CICS region is different from a CICS region that initializes with a START=COLD parameter, as follows:

- The state of the global catalog is ignored. It can contain either data from a previous run of CICS, or it can be newly initialized. Any previous data is purged.
- The state of the system log is ignored. It can contain either data from a previous run of CICS, or it can reference new log streams. CICS does not keep any

information saved in the system log from a previous run. The primary and secondary system log streams are purged and CICS begins writing a new system log.

- Because CICS is starting a new catalog, it uses a new logname token in the "exchange lognames" process when connecting to partner systems. Thus, remote systems are notified that CICS has performed a cold start and cannot resynchronize.
- User journals are not affected by starting CICS with the START=INITIAL parameter.

Note: An initial start can also result from a START=COLD parameter if the global catalog is newly initialized and does not contain a recovery manager control record. If the recovery manager finds that there is no control record on the catalog, it issues a message to the console prompting the operator to reply with a GO or CANCEL response. If the response is GO, CICS performs an initial start as if START=INITIAL was specified.

For more information about the effect of the state of the global catalog and the system log on the type of start CICS performs, see the *CICS System Definition Guide*.

Chapter 5. CICS warm restart

This section describes the CICS startup processing specific to a warm restart.

If you specify START=AUTO, which is the recommended method, CICS determines which type of start to perform using information retrieved from the recovery manager's control record in the global catalog. If the type-of-restart indicator in the control record indicates “warm start possible”, CICS performs a warm restart.

You should not attempt to compress a library after a warm start, without subsequently performing a CEMT SET PROGRAM(PRGMID) NEWCOPY for each program in the library. This is because on a warm start, CICS obtains the directory information for all programs which were installed on the previous execution. Compressing a library could alter its contents and subsequently invalidate the directory information known to CICS.

See Chapter 6, “CICS emergency restart,” on page 61 for the restart processing performed if the type-of-restart indicates “emergency restart needed”.

Rebuilding the CICS state after a normal shutdown

During a warm restart, CICS initializes using information from the catalogs and system log to restore the region to its state at the previous normal shutdown.

Note: CICS needs both the catalogs and the system log from the previous run of CICS to perform a warm restart—the catalogs alone are not sufficient. If you run CICS with the system log defined as TYPE(DUMMY), CICS appears to shut down normally, but only the global catalog portion of the warm keypoint is written. Therefore, without the warm keypoint information from the system log, CICS cannot perform a warm restart. CICS startup fails unless you specify an initial start with START=INITIAL.

Recovering their own state is the responsibility of the individual resource managers (such as file control) and the CICS domains. This topic discusses the process of rebuilding their state from the catalogs and system log, in terms of the following resources:

- Files
- Temporary storage queues
- Transient data queues
- Transactions
- LIBRARY resources
- Programs, including mapsets and partitionsets
- Start requests
- Monitoring and statistics
- Journals and journal models
- Terminal control resources
- Distributed transaction resources
- URIMAP definitions and virtual hosts

Files

File control information from the previous run is recovered from information recorded in the CICS catalog only.

File resource definitions for VSAM and BDAM files, data tables, and LSR pools are installed from the global catalog, including any definitions that were added dynamically during the previous run. The information recovered and reinstalled in this way reflects the state of all file resources at the previous shutdown. For example:

- If you manually set a file closed (which changes its status to UNENABLED) and perform a normal shutdown, it remains UNENABLED after the warm restart.
- Similarly, if you set a file DISABLED, it remains DISABLED after the warm restart.

Note: An exception to the above rule occurs when there are updates to a file to be backed out during restarts, in which case the file is opened regardless of the OPENTIME option. At a warm start, there cannot be any in-flight units of work to back out, so this backout can only occur when retrying backout-failed units of work against the file.

CICS closes all files at shutdown, and, as a general rule, you should expect your files to be re-installed on restart as either:

- OPEN and ENABLED if the OPENTIME option is STARTUP
- CLOSED and ENABLED if the OPENTIME option is FIRSTREF.

The FCT and the CSD*xxxx* system initialization parameters are ignored.

File control uses the system log to reconstruct the internal structures, which it uses for recovery.

Data set name blocks

Data set name blocks (DSNBs), one for each data set opened by CICS file control, are recovered during a warm restart.

If you have an application that creates many temporary data sets, with a different name for every data set created, it is important that your application removes these after use. If applications fail to get rid of unwanted name blocks they can, over time, use up a considerable amount of CICS dynamic storage. See the *CICS System Programming Reference* for information about using the SET DSNAME REMOVE command to remove unwanted data set name blocks.

Reconnecting to SMSVSAM for RLS access

CICS connects to the SMSVSAM server, if present, and exchanges RLS recovery information.

In this exchange, CICS finds out whether SMSVSAM has lost any retained locks while CICS has been shut down. This could happen, for example, if SMSVSAM could not recover from a coupling facility failure that caused the loss of the lock structure. If this has happened, CICS is notified by SMSVSAM to perform **lost locks recovery**. See “Lost locks recovery” on page 89 for information about this process.

Recreating non-RLS retained locks

For non-RLS files, the CICS enqueue domain rebuilds the retained locks relating to shunted units of work.

Temporary storage

Auxiliary temporary storage queue information (for both recoverable and non-recoverable queues) is retrieved from the warm keypoint. Note that TS READ pointers are recovered on a warm restart (which is not the case on an emergency restart).

CICS opens the auxiliary temporary storage data set for update.

Temporary storage data sharing server

Any queues written to a shared temporary storage pool, even though non-recoverable, persist across a warm restart.

Transient data

Transient data initialization on a warm restart depends on the TDINTRA system initialization parameter, which specifies whether or not TD is to initialize with empty intrapartition queues. The different options are discussed as follows:

TDINTRA=NOEMPTY (the default)

All transient data resource definitions are installed from the global catalog, including any definitions that were added dynamically during the previous run. TD queues are always installed as enabled.

CICS opens any extrapartition TD queues that need to be opened—that is, any that specify OPEN=INITIAL.

Note: If, during the period when CICS is installing the TD queues, an attempt is made to write a record to a CICS-defined queue that has not yet been installed (for example, CSSL), CICS writes the record to the CICS-defined queue CXRF.

The recovery manager returns log records and keypoint data associated with TD queues. CICS applies this data to the installed queue definitions to return the TD queues to the state they were in at normal shutdown. Logically recoverable, physically recoverable, and non-recoverable intrapartition TD queues are recovered from the warm keypoint data.

Trigger levels (for TERMINAL and SYSTEM only):

After the queues have been recovered, CICS checks the trigger level status of each intrapartition TD queue that is defined with FACILITY(TERMINAL|SYSTEM) to determine whether a start request needs to be rescheduled for the trigger transaction.

If a trigger transaction failed to complete during the previous run (that is, did not reach the empty queue (QZERO) condition) or the number of items on the queue is greater than the trigger level, CICS schedules a start request for the trigger transaction.

This does not apply to trigger transactions defined for queues that are associated with files (FACILITY(FILE)).

TDINTRA=EMPTY

If you specify this option, the transient data queues are cold started, but the resource definitions are warm started.

The following processing takes place:

- All intrapartition TD queues are initialized empty.
- The queue resource definitions are installed from the global catalog, but they are not updated by any log records or keypoint data. They are always installed enabled.

This option is intended for use when initiating remote site recovery (see Chapter 6, "CICS emergency restart," on page 61), but you can also use it for a normal warm restart. For example, you might want to 'cold start' the intrapartition queues when switching to a new data set if the old one is corrupted, while preserving all the resource definitions from the catalog.

You cannot specify a general cold start of transient data while the rest of CICS performs a warm restart, as you can for temporary storage.

Transactions

All transaction and transaction class resource definitions are installed from the CSD, and updated with information from the warm keypoint in the system log. The resource definitions installed from the catalog include any that were added dynamically during the previous run.

LIBRARY resources

On WARM or EMERGENCY start, all LIBRARY definitions will be restored from the catalog, and the actual search order through the list of LIBRARY resources that was active at the time of the preceding shutdown will be preserved.

The latter will ensure that the search order of two LIBRARY resources of equal RANKING will remain the same. An equal RANKING implies that the relative search order of the LIBRARY resources is unimportant, but unexpected behavior might result if this order changed after a warm or emergency restart.

If a LIBRARY with an option of CRITICAL(YES) is restored from the catalog, and one of the data sets in its concatenation is no longer available, a message will be issued to allow the operator to choose whether to continue the CICS startup, or to cancel it. This Go or Cancel message will be preceded by a set of messages providing information on any data sets which are not available. For LIBRARY resources, with an option of CRITICAL(NO), this condition will not cause CICS startup to fail, but a warning message will be issued and the LIBRARY will not be reinstalled. This warning message will be preceded by a set of messages providing information on any data sets which are not available

Programs

The recovery of program, mapset, and partitionset resource definitions depends on whether you are using program autoinstall and, if you are, whether you have requested autoinstall cataloging (specified by the system initialization parameter PGAICTLG=ALL|MODIFY).

No autoinstall for programs

If program autoinstall is disabled (PGAIPGM=INACTIVE), all program, mapset, and partitionset resource definitions are installed from the CSD, and updated with information from the warm keypoint in the system log.

The resource definitions installed from the catalog include any that were added dynamically during the previous run.

Autoinstall for programs

If program autoinstall is enabled (PGAIPGM=ACTIVE), program, mapset, and partitionset resource definitions are installed from the CSD only if they were cataloged; otherwise they are installed at first reference by the autoinstall process.

All definitions installed from the CSD are updated with information from the warm keypoint in the system log.

CICS catalogs program, mapset, and partitionset resource definitions as follows:

- If they are installed from predefined definitions in the CSD, either during a cold start or by an explicit INSTALL command, CICS catalogs the definitions.
- If the PGAICTLG system initialization parameter specifies ALL, CICS catalogs all the autoinstalled program-type definitions, and these are reinstalled during the warm restart.
- If the PGAICTLG system initialization parameter specifies MODIFY, CICS catalogs only those autoinstalled program-type definitions that are modified by a SET PROGRAM command, and these are reinstalled during the warm restart.

Start requests

In general, start requests are recovered together with any associated start data.

Recovery can, however, be suppressed by specifying explicit cold start system initialization parameters for temporary storage, interval control, or basic mapping support (on the TS, ICP, and BMS system initialization parameters respectively). Any data associated with suppressed starts is discarded.

The rules governing the operation of the explicit cold requests on system initialization parameters are:

- ICP=COLD suppresses all starts that do not have both data and a terminal associated with them. It also suppresses any starts that had not expired at shutdown. This includes BMS starts.
- TS=COLD (or TS main only) suppresses all starts that had data associated with them.
- BMS=COLD suppresses all starts relating to BMS paging.

Start requests that have not been suppressed for any of the above reasons either continue to wait if their start time or interval has not yet expired, or they are processed immediately. For start requests with terminals, consider the effects of the CICS restart on the set of installed terminal definitions. For example, if the terminal specified on a start request is no longer installed after the CICS restart, CICS invokes an XALTENF global user exit program (if enabled), but not the XICTENF exit.

Monitoring and statistics

The CICS monitoring and statistics domains retrieve their status from their control records stored in the global catalog at the previous shutdown.

This is modified by any runtime system initialization parameters.

Journal names and journal models

The CICS log manager restores the journal name and journal model definitions from the global catalog. Journal name entries contain the names of the log streams used in the previous run, and the log manager reconnects to these during the warm restart.

Terminal control resources

Terminal control information is installed from the warm keypoint in the global catalog, or installed from the terminal control table (TCT), depending on whether the resources are CSD-defined or TCT-defined.

CSD-defined resource definitions

When resources are defined in the CICS System Definition data set (CSD), terminal control information is installed from the warm keypoint in the global catalog.

CICS installs the following terminal control resource definitions from the global catalog:

- All permanent terminal devices, originally installed from explicit resource definitions, and profiles.
- The following autoinstalled APPC connections:
  - Synclevel-2-capable connections (for example, CICS-to-CICS connections)
  - Synclevel-1-capable, limited resource connections installed on a CICS that is a member of a VTAM generic resource.

  Other autoinstalled terminals are not recovered, because they are removed from the warm keypoint during normal shutdown. This ensures that their definitions are installed only when terminal users next log on after a CICS restart that follows a normal shutdown.

When a multiregion operation (MRO) connection is restored, it has the same status that was defined in the CSD. Any changes of status, for example the service status, are not saved on the global catalog, so are not recovered during a warm or emergency restart.

Only the global catalog is referenced for terminals defined in the CSD.

To add a terminal after initialization, use the CEDA INSTALL or EXEC CICS CREATE command, or the autoinstall facility. To delete a terminal definition, use the DISCARD command or, if autoinstalled, allow it to be deleted by the autoinstall facility after the interval specified by the AILDELAY system initialization parameter.

TCAM and sequential (BSAM) devices

Terminal control information for TCAM and sequential terminal devices is installed from the terminal control table (TCT).

CICS installs TCAM and sequential terminal resource definitions as follows:

- **Same TCT as last run**. CICS installs the TCT and then modifies the terminal entries in the table by applying the cataloged data from the terminal control warm keypoint from the previous shutdown. This means that, if you reassemble the TCT and keep the same suffix, any changes you make could be undone by the warm keypoint taken from the catalog.

- **Different TCT from last run**. CICS installs the TCT only, and does not apply the warm keypoint information, effectively making this a cold start for these devices.

Note: CICS TS for z/OS, Version 4.1 supports only *remote* TCAM terminals—that is, the only TCAM terminals you can define are those attached to a remote, pre-CICS TS 3.1, terminal-owning region by TCAM/DCB.

Distributed transaction resources

CICS retrieves its logname from the recovery manager control record in the global catalog for use in the "exchange lognames" process with remote systems. Resynchronization of indoubt units of work takes place after CICS completes reconnection to remote systems.

See the *CICS Recovery and Restart Guide* for information about recovery of distributed units of work.

URIMAP definitions and virtual hosts

Installed URIMAP definitions for CICS Web support are restored from the global catalog, including their enable status. Virtual hosts, which are created by CICS using the host names specified in installed URIMAP definitions, are also restored to their former enabled or disabled state.

Chapter 6. CICS emergency restart

This section describes the CICS startup processing specific to an emergency restart.

If you specify START=AUTO, CICS determines what type of start to perform using information retrieved from the recovery manager's control record in the global catalog. If the type-of-restart indicator in the control record indicates "emergency restart needed", CICS performs an emergency restart.

See Chapter 5, "CICS warm restart," on page 53 for the restart processing performed if the type-of-restart indicates "warm start possible".

Recovering after a CICS failure

CICS initialization for an emergency restart after a CICS failure is the same as initialization for a warm restart, with some additional processing.

The additional processing performed for an emergency restart is mainly related to the recovery of in-flight transactions. There are two aspects to the recovery operation:

1. Recovering information from the system log
2. Driving backout processing for in-flight units of work

Recovering information from the system log

At some point during initialization (and before CICS performs program list table post-initialization (PLTPI) processing), the recovery manager scans the system log backwards. CICS uses the information retrieved to restore the region to its state at the time of the abnormal termination.

For non-RLS data sets and other recoverable resources, any locks (ENQUEUES) that were held before the CICS failure are re-acquired during this initial phase.

For data sets accessed in RLS mode, the locks that were held by SMSVSAM for in-flight tasks are converted into retained locks at the point of abnormal termination.

Driving backout processing for in-flight units of work

When initialization is almost complete, and after the completion of PLTPI processing, the recovery manager starts backout processing for any units of work that were in-flight at the time of the failure of the previous run.

Starting recovery processing at the end of initialization means that it occurs concurrently with new work.

Concurrent processing of new work and backout

The backout of units of work that occurs after an emergency restart is the same process as dynamic backout of a failed transaction. Backing out in-flight transactions continues after "control is given to CICS", which means that the process takes place concurrently with new work arriving in the region.

Any non-RLS locks associated with in-flight (and other failed) transactions are acquired as active locks for the tasks attached to perform the backouts. This means that, if any new transaction attempts to access non-RLS data that is locked by a backout task, it waits normally rather than receiving the LOCKED condition.

Retained RLS locks are held by SMSVSAM, and these do not change while backout is being performed. Any new transactions that attempt to access RLS resources locked by a backout task receive a LOCKED condition.

For both RLS and non-RLS resources, the backout of in-flight transactions after an emergency restart is indistinguishable from dynamic transaction backout.

Effect of delayed recovery on PLTPI processing

Because recovery processing does not take place until PLTPI processing is complete, PLT programs may fail during an emergency restart if they attempt to access resources protected by retained locks. If PLT programs are not written to handle the LOCKED exception condition they abend with an AEX8 abend code.

If successful completion of PLTPI processing is essential before your CICS applications are allowed to start, consider alternative methods of completing necessary PLT processing. You may have to allow emergency restart recovery processing to finish, and then complete the failed PLTPI processing when the locks have been released.

Other backout processing

The recovery manager also drives the backout processing for any units of work that were in a backout-failed state at the time of a CICS failure and the commit processing for any units of work that were in a commit-failed state at the time of a CICS failure.

The recovery manager drives these backout and commit processes because the condition that caused them to fail may be resolved by the time CICS restarts. If the condition that caused a failure has not been resolved, the unit of work remains in backout- or commit-failed state. See “Backout-failed recovery” on page 79 and “Commit-failed recovery” on page 83 for more information.

Rebuilding the CICS state after an abnormal termination

The individual resource managers, such as file control, and the CICS domains are responsible for recovering their state as it was at an abnormal termination.

The process of rebuilding the state for the following resources is the same as for a warm restart:

- Transactions
- Programs
- Monitoring and statistics
- Journal names and journal models
- URIMAP definitions and virtual hosts

The processing for other resources is different from a warm restart.

Files

All file control state data and resource definitions are recovered in the same way as on a warm start.

Reconnecting to SMSVSAM for RLS access

As on a warm restart, CICS connects to the SMSVSAM server. In addition to notifying CICS about lost locks, VSAM also informs CICS of the units of work belonging to the CICS region for which it holds retained locks. See "Lost locks recovery" on page 89 for information about the lost locks recovery process for CICS.

CICS uses the information it receives from SMSVSAM to eliminate orphan locks.

RLS restart processing and orphan locks

CICS emergency restart performs CICS-RLS restart processing during which **orphan locks** are eliminated. An orphan lock is one that is held by VSAM RLS on behalf of a specific CICS but unknown to the CICS region, and a VSAM interface enables CICS to detect units of work that are associated with such locks.

Orphan locks can occur if a CICS region acquires an RLS lock, but then fails before logging it. Records associated with orphan locks that have not been logged cannot have been updated, and CICS can safely release them.

Note: Locks that fail to be released during UOW commit processing cause the UOW to become a commit-failed UOW. CICS automatically retries commit processing for these UOWs, but if the locks are still not released before the CICS region terminates, these also are treated as orphan locks during the next restart.

Recreating non-RLS retained locks

Recovery is the same as for a warm restart. See "Recreating non-RLS retained locks" on page 54 for details.

Temporary storage

Auxiliary temporary storage queue information for recoverable queues only is retrieved from the warm keypoint. The TS READ pointers are not recovered and are set to zero.

If a nonzero TSAGE parameter is specified in the temporary storage table (TST), all queues that have not been referenced for this interval are deleted.

Transient data

Recovery of transient data is the same as for a warm start, with the following exceptions:

- Non-recoverable queues are not recovered.
- Physically recoverable queues are recovered, using log records and keypoint data. Generally, backing out units of work that were in-flight at the time of the CICS failure does not affect a physically recoverable TD intrapartition data set. Changes to physically recoverable TD queues are committed immediately, with the result that backing out a unit of work does not affect the physical data set. An exception to this is the last read request from a TD queue by a unit of work that fails in-flight because of a CICS failure. In this case, CICS backs out the last read, ensuring that the queue item is not deleted by the read. A further exception occurs when the read is followed by a "delete queue" command. In this case, the read is not backed out, because the whole queue is deleted.

Start requests

In general, start requests are recovered only when they are associated with recoverable data or are protected and the issuing unit of work is indoubt.

However, recovery can be further limited by the use of the specific COLD option on the system initialization parameter for TS, ICP, or BMS. If you suppress start requests by means of the COLD option on the appropriate system initialization parameter, any data associated with the suppressed starts is discarded. The rules are:

- ICP=COLD suppresses all starts including BMS starts.
- TS=COLD (or TS main only) suppresses all starts that had data associated with them.
- BMS=COLD suppresses all starts relating to BMS paging.

Start requests that have not been suppressed for any of the above reasons either continue to wait if their start time or interval has not yet expired, or are processed immediately.

For start requests with terminals, consider the effects of the CICS restart on the set of installed terminal definitions. For example, if the terminal specified on a start request is no longer installed after the CICS restart, CICS invokes an XALTENF global user exit program (if enabled), but not the XICTENF exit.

Terminal control resources

Terminal control information is installed from the warm keypoint in the global catalog, or installed from the TCT, depending on whether the definitions are CSD-defined or TCT-defined.

CSD-defined resource definitions

CICS retrieves the state of the CSD-eligible terminal control resources from the catalog entries that were written:

- During a previous cold start
- When resources were added with EXEC CICS CREATE or CEDA INSTALL
- When resources were added with autoinstall (subject to the AIRDELAY system initialization parameter)
- Rewritten to the catalog at an intervening warm shutdown

The state of the catalog may have been modified for some of the above resources by their removal with a CEMT, or an EXEC CICS DISCARD, command.

CICS uses records from the system log, written when any terminal resources were being updated, to perform any necessary recovery on the cataloged data. This may be needed if terminal resources are installed or deleted while CICS is running, and CICS fails before the operation is completed.

Some terminal control resources are installed or deleted in "installable sets" as described under "Committing and cataloging resources installed from the CSD" on page 49. If modifications are made to terminal resource definitions while CICS is running, CICS writes the changes in the form of forward recovery records to the system log. If the installation or deletion of installable sets or individual resources

is successful, but CICS abnormally terminates before the catalog can be updated, CICS recovers the information from the forward recovery records on the system log.

If the installation or deletion of installable sets or individual resources is unsuccessful, or has not reached commit point when CICS abnormally terminates, CICS does not recover the changes.

In this way, CICS ensures that the terminal entries recovered at emergency restart consist of complete logical sets of resources (for connections, sessions, and pipelines), and complete terminal resources and autoinstall models, and that the catalog reflects the real state of the system accurately.

TCAM and sequential (BSAM) devices

CICS installs TCAM and sequential terminal resource definitions from the TCT. Because there is no warm keypoint if the previous run terminated abnormally, the TCT cannot be modified as on a warm start. Whatever is defined in the TCT is installed, and the effect is the same whether or not it is a different TCT from the last run.

Note: CICS TS for z/OS, Version 4.1 supports only *remote* TCAM terminals. That is, the only TCAM terminals you can define are those attached to a remote, pre-CICS TS 3.1, terminal-owning region by TCAM/DCB.

Distributed transaction resources

CICS retrieves its logname from the recovery manager control record in the global catalog for use in the "exchange lognames" process with remote systems. Resynchronization of indoubt units of work takes place when CICS completes reconnection to remote systems.

See the CICS Installation Guide for information about recovery of distributed units of work.

Chapter 7. Automatic restart management

CICS uses the automatic restart manager (ARM) component of MVS to increase the availability of your systems.

MVS automatic restart management is a sysplex-wide integrated automatic restart mechanism that performs the following tasks:

- Restarts an MVS subsystem in place if it abends (or if a monitor program notifies ARM of a stall condition)
- Restarts all the elements of a workload (for example, CICS TORs, AORs, FORs, DB2, and so on) on another MVS image after an MVS failure
- Restarts CICS data sharing servers in the event of a server failure.
- Restarts a failed MVS image

CICS reconnects to DBCTL and VTAM automatically if either of these subsystems restart after a failure. CICS is not dependent on using ARM to reconnect in the event of failure.

The MVS automatic restart manager provides the following benefits:

- Enables CICS to preserve data integrity automatically in the event of any system failure.
- Eliminates the need for operator-initiated restarts, or restarts by other automatic packages, thereby:
  - Improving emergency restart times
  - Reducing errors
  - Reducing complexity.
- Provides cross-system restart capability. It ensures that the workload is restarted on MVS images with spare capacity, by working with the MVS workload manager.
- Allows all elements within a restart group to be restarted in parallel. Restart levels (using the ARM WAITPRED protocol) ensure the correct starting sequence of dependent or related subsystems.

Restrictions

You cannot use MVS automatic restart for CICS regions running with XRF. If you specify XRF=YES, CICS deregisters from ARM and continues initialization with XRF support.

MVS automatic restart management is available only to those MVS subsystems that register with ARM. CICS regions register with ARM automatically as part of CICS system initialization. If a CICS region fails before it has registered for the first time with ARM, it will not be restarted. After a CICS region has registered, it is restarted by ARM according to a predefined policy for the workload.

CICS ARM processing

A prime objective of CICS support for the MVS automatic restart manager (ARM) is to preserve data integrity automatically in the event of any system failure.

If CICS is restarted by ARM with the same persistent JCL, CICS forces START=AUTO to ensure data integrity.

Registering with ARM

To register with ARM, you must implement automatic restart management on the MVS images that the CICS workload is to run on. You must also ensure that the CICS startup JCL used to restart a CICS region is suitable for ARM.

Before you begin

The implementation of ARM is part of setting up your MVS environment to support CICS. See the *CICS Transaction Server for z/OS Installation Guide* for details.

About this task

During initialization CICS registers with ARM automatically.

CICS always registers with ARM because CICS needs to know whether it is being restarted by ARM and, if it is, whether or not the restart is with persistent JCL. (The ARM registration response to CICS indicates whether or not the same JCL that started the failed region is being used for the ARM restart.) You indicate whether MVS is to use the same JCL or command text that previously started CICS by specifying PERSIST as the *restart_type* operand on the **RESTART_METHOD** parameter in your automatic restart management policy.

When it registers with ARM, CICS passes the value 'SYSCICS' as the element type, and the string 'SYSCICS_*aaaaaaaa*' as the element name, where *aaaaaaaa* is the CICS applid. Using the applid in the element name means that only one CICS region can successfully register with ARM for a given applid. If two CICS regions try to register with the same applid, the second region is rejected by ARM.

Waiting for predecessor subsystems

During initialization CICS issues an ARM WAITPRED (**wait predecessor**) request to wait, if necessary, for predecessor subsystems (such as DB2 and DBCTL) to become available.

This is indicated by message DFHKE0406. One reason for this wait is to ensure that CICS can resynchronize with its partner resource managers for recovery purposes before accepting new work from the network.

De-registering from ARM

During normal shutdown, CICS de-registers from ARM to ensure that it is not automatically restarted. Also, if you want to perform an immediate shutdown and do not want ARM to cause an automatic restart, you can specify the NORESTART option on the PERFORM SHUT IMMEDIATE command.

About this task

CICS also de-registers during initialization if it detects XRF=YES is specified as a system initialization parameter—XRF takes precedence over ARM.

Some error situations that occur during CICS initialization cause CICS to issue a message, with an operator prompt to reply GO or CANCEL. If you reply

CANCEL, CICS de-registers from ARM before terminating, because if CICS remained registered, an automatic restart would probably encounter the same error condition.

For other error situations, CICS does not de-register, and automatic restarts follow. To control the number of restarts, specify in your ARM policy the number of times ARM is to restart a failed CICS region.

Failing to register

If ARM support is present but the register fails, CICS issues message DFHKE0401. In this case, CICS does not know if it is being restarted by ARM, and therefore it doesn't know whether to override the START parameter to force an emergency restart to preserve data integrity.

If START=COLD or START=INITIAL is specified as a system initialization parameter and CICS fails to register, CICS also issues message DFHKE0408. When CICS is restarting with START=COLD or START=INITIAL, CICS relies on ARM to determine whether to override the start type and change it to AUTO. Because the REGISTER has failed, CICS cannot determine whether the region is being restarted by ARM, and so does not know whether to override the start type. Message DFHKE0408 prompts the operator to reply ASIS or AUTO, to indicate the type of start CICS is to perform:

- A reply of ASIS means that CICS is to perform the start specified on the START parameter.
- A reply of AUTO means that CICS is being restarted by ARM, and the type of start is to be resolved by CICS. If the previous run terminated abnormally, CICS will perform an emergency restart.

  Note: A CICS restart can have been initiated by ARM, even though CICS registration with ARM has failed in the restarted CICS.

ARM couple data sets

You must ensure that you define the couple data sets required for ARM and that they are online and active before you start any CICS region for which you want ARM support.

- CICS automatic ARM registration fails if the couple data sets are not active at CICS startup. When CICS is notified by ARM that registration has failed for this reason, CICS assumes this means that you do not want ARM support, and CICS initialization continues.
- If ARM loses access to the couple data sets, the CICS registration is lost. In this event, ARM cannot restart a CICS region that fails.

See *z/OS MVS Setting Up a Sysplex* for information about ARM couple data sets and ARM policies.

CICS restart JCL and parameters

Each CICS restart can use the previous startup JCL and system initialization parameters, or can use a new job and parameters.

You cannot specify XRF=YES if you want to use ARM support. If the **XRF** system initialization parameter is changed to XRF=YES for a CICS region being restarted by ARM, CICS issues message DFHKE0407 to the console, then terminates.

CICS START options

You are recommended to specify START=AUTO, which causes a warm start after a normal shutdown and an emergency restart after failure.

You are also recommended always to use the same JCL, even if it specifies START=COLD or START=INITIAL, to ensure that CICS restarts correctly when restarted by the MVS automatic restart manager after a failure.

If you specify START=COLD (or INITIAL) and your ARM policy specifies that the automatic restart manager is to use the same JCL for a restart following a CICS failure, CICS overrides the start parameter when restarted by ARM and enforces START=AUTO. CICS issues message DFHPA1934 and ensures the resultant emergency restart handles recoverable data correctly.

If the ARM policy specifies different JCL for an automatic restart and that JCL specifies START=COLD, CICS uses this parameter value but risks losing data integrity. Therefore, if you need to specify different JCL to ARM, specify START=AUTO to ensure data integrity.

Workload policies

Workloads are started initially by scheduling or automation products.

The components of the workload, and the MVS images capable of running them, are specified as part of the policies for MVS workload manager and ARM. The MVS images must have access to the databases, logs, and program libraries required for the workload.

Administrative policies provide ARM with the necessary information to perform appropriate restart processing. You can define one or more administrative policies, but can have only one active policy for all MVS images in a sysplex. You can modify administrative policies by using an MVS-supplied utility, and can activate a policy with the MVS SETXCF command.

Connecting to VTAM

VTAM is at restart level 1, the same as DB2 and DBCTL.

However, VTAM is not restarted when failed subsystems are being restarted on another MVS, because ARM expects VTAM to be running on all MVS images in the sysplex. For this reason, CICS and VTAM are not generally part of the same restart group.

In a VTAM network, the session between CICS and VTAM is started automatically if VTAM is started before CICS. If VTAM is not active when you start (or restart) CICS, you receive the following messages:

```
+DFHSI1589D 'applid' VTAM is not currently active.
+DFHSI1572 'applid'  Unable to OPEN VTAM ACB - RC=xxxxxxxx, ACB CODE=yy.
```

CICS provides a new transaction, COVR, to open the VTAM ACB automatically when VTAM becomes available. See “The COVR transaction” on page 71 for more information about this.

The COVR transaction

To ensure that CICS reconnects to VTAM in the event of a VTAM abend, CICS keeps retrying the OPEN VTAM ACB using a time-delay mechanism via the non-terminal transaction COVR.

After CICS has completed clean-up following the VTAM failure, it invokes the CICS open VTAM retry (COVR) transaction. The COVR transaction invokes the terminal control open VTAM retry program, DFHZCOVR, which performs an OPEN VTAM retry loop with a 5-second wait. CICS issues a DFHZC0200 message every minute, while the open is unsuccessful, and each attempt is recorded on the CSNE transient data queue. After ten minutes, CICS issues a DFHZC0201 message and terminates the transaction. If CICS shutdown is initiated while the transaction is running, CICS issues a DFHZC0201 message and terminates the transaction.

You cannot run the COVR transaction from a terminal. If you invoke COVR from a terminal, it abends with an AZCU transaction abend.

Messages associated with automatic restart

There are some CICS messages for ARM support, which CICS can issue during startup if problems are encountered when CICS tries to connect to ARM.

The message numbers are:

DFHKE0401
DFHKE0402
DFHKE0403
DFHKE0404
DFHKE0405
DFHKE0406
DFHKE0407
DFHKE0408
DFHKE0410
DFHKE0411
DFHZC0200
DFHZC0201

For the text of these messages, see *CICS Messages and Codes*.

Automatic restart of CICS data-sharing servers

All three types of CICS data-sharing server—temporary storage, coupling facility data tables, and named counters—support automatic restart using the services of automatic restart manager.

The servers also have the ability to wait during start-up, using an event notification facility (ENF) exit, for the coupling facility structure to become available if the initial connection attempt fails.

Server ARM processing

During initialization, a data-sharing server unconditionally registers with ARM, except when starting up for unload or reload. A server does not start if registration fails, with return code 8 or above.

If a server encounters an unrecoverable problem with the coupling facility connection, consisting either of lost connectivity or a structure failure, it cancels itself using the server command `CANCEL RESTART=YES`. This terminates the existing connection, closes the server and its old job, and starts a new instance of the server job.

You can also restart a server explicitly using either the server command CANCEL RESTART=YES, or the MVS command CANCEL *jobname*,ARMRESTART

By default, the server uses an ARM element type of SYSCICSS, and an ARM element identifier of the form DFH*xxnn_poolname* where *xx* is the server type (XQ, CF or NC) and *nn* is the one- or two-character &SYSCLONE identifier of the MVS image. You can use these parameters to identify the servers for the purpose of overriding automatic restart options in the ARM policy.

Waiting on events during initialization

If a server is unable to connect to its coupling facility structure during server initialization because of an environmental error, the server uses an ENF event exit to wait for cross-system extended services (XES) to indicate that it is worth trying again.

The event exit listens for either:

- A specific XES event indicating that the structure has become available, or
- A general XES event indicating that some change has occurred in the status of coupling facility resources (for example, when a new CFRM policy has been activated).

When a relevant event occurs, the server retries the original connection request, and continues to wait and retry until the connection succeeds. A server can be canceled at this stage using an MVS CANCEL command if necessary.

Server initialization parameters for ARM support

The server startup parameters for ARM support are:

ARMELEMENTNAME=*elementname*
: specifies the automatic restart manager element name, up to 16 characters, to identify the server to ARM for automatic restart purposes.

ARMELEMENTTYPE=*elementtype*
: specifies the automatic restart manager element type, up to 8 characters for use in ARM policies as a means of classifying similar elements.

These parameters are the same for all the data sharing servers. For more details, see the automatic restart manager (ARM) parameters in the *CICS System Definition Guide* .

Server commands for ARM support

The following are the ARM options you can use on server commands:

CANCEL RESTART={NO|YES}
: terminates the server immediately, specifying whether or not automatic restart should be requested. The default is RESTART=NO.

  You can also enter RESTART on its own for RESTART=YES, NORESTART for RESTART=NO.

ARMREGISTERED
: shows whether ARM registration was successful (YES or NO).

ARM
: This keyword, in the category of display keywords that represent combined options, can be used to display all ARM-related parameter values. It can also be coded as **ARMSTATUS**.

These commands are the same for all the data sharing servers.

Chapter 8. Unit of work recovery and abend processing

A number of different events can cause the abnormal termination of transactions in CICS.

These events include:

- A transaction ABEND request issued by a CICS management module.
- A program check or operating system abend (this is trapped by CICS and converted into an ASRA or ASRB transaction abend).
- An ABEND request issued by a user application program.
- A CEMT, or EXEC CICS, command such as **SET TASK PURGE** or FORCEPURGE.

  Note: Unlike the **EXEC CICS ABEND** command above, these EXEC CICS commands cause other tasks to abend, not the one issuing the command.
- A transaction abend request issued by DFHZNEP or DFHTEP following a communication error. This includes the abnormal termination of a remote CICS during processing of in-flight distributed UOWs on the local CICS.
- An abnormal termination of CICS, in which all in-flight transactions are effectively abended as a result of the CICS region failing.

  In-flight transactions are recovered during a subsequent emergency restart to enable CICS to complete the necessary backout of recoverable resources, which is performed in the same way as if the task abended while CICS was running.

Unit of work recovery

A unit of work in CICS is also the unit of recovery - that is, it is the atomic component of the transaction in which any changes made either must all be committed, or must all be backed out.

A transaction can be composed of a single unit of work or multiple units of work. In CICS, recovery is managed at the unit of work level.

For recovery purposes, CICS recovery manager is concerned only with the units of work that have not yet completed a syncpoint because of some failure. This topic discusses how CICS handles these failed units of work.

The CICS recovery manager has to manage the recovery of the following types of unit of work failure:

In-flight-failed
: The transaction fails before the current unit of work reaches a syncpoint, as a result either of a task abend, or the abnormal termination of CICS. The transaction is abnormally terminated, and recovery manager initiates backout of any changes made by the unit of work.

  See “Transaction backout” on page 74.

Commit-failed
: A unit of work fails during commit processing while taking a syncpoint. A partial copy of the unit of work is shunted to await retry of the commit process when the problem is resolved.

  This does *not* cause the transaction to terminate abnormally.

See “Commit-failed recovery” on page 83.

Backout-failed
A unit of work fails while backing out updates to file control recoverable resources. (The concept of backout-failed applies in principle to any resource that performs backout recovery, but CICS file control is the only resource manager to provide backout failure support.) A partial copy of the unit of work is shunted to await retry of the backout process when the problem is resolved.

Note: Although the failed backout may have been attempted as a result of the abnormal termination of a transaction, the backout failure itself does *not* cause the transaction to terminate abnormally.

For example, if a transaction initiates backout through an EXEC CICS SYNCPOINT ROLLBACK command, CICS returns a normal response (not an exception condition) and the transaction continues executing. It is up to recovery manager to ensure that locks are preserved until backout is eventually completed.

If some resources involved in a unit of work are backout-failed, while others are commit-failed, the UOW as a whole is flagged as backout-failed.

See “Backout-failed recovery” on page 79.

Indoubt-failed
A distributed unit of work fails while in the indoubt state of the two-phase commit process. The transaction is abnormally terminated. If there are normally more units of work that follow the one that failed indoubt, these will not be executed as a result of the abend.

A partial copy of the unit of work is shunted to await resynchronization when CICS re-establishes communication with its coordinator. This action happens only when the transaction resource definition specifies that units of work are to wait in the event of failure while indoubt. If they are defined with WAIT(NO), CICS takes the action specified on the ACTION parameter, and the unit of work cannot become failed indoubt.

See “Indoubt failure recovery” on page 84.

Transaction backout

If the resources updated by a failed unit of work are defined as recoverable, CICS automatically performs transaction backout of all uncommitted changes to the recoverable resources.

Transaction backout is mandatory and automatic - there is not an option on the transaction resource definition allowing you to control this. You can, however, control backout of the resources on which your transactions operate by defining whether or not they are recoverable.

In transaction backout, CICS restores the resources specified as recoverable to the state they were in at the beginning of the interrupted unit of work (that is, at start of task or completion of the most recent synchronization point). The resources are thus restored to a consistent state.

In general, the same process of transaction backout is used for individual units of work that abend while CICS is running and for in-flight tasks recovered during emergency restart. One difference is that dynamic backout of a **single** abnormally

terminating transaction takes place immediately. Therefore, it does not cause any active locks to be converted into retained locks. In the case of a CICS region abend, in-flight tasks have to wait to be backed out when CICS is restarted, during which time the locks are retained to protect uncommitted resources.

To restore the resources to the state they were in at the beginning of the unit of work, CICS preserves a description of their state at that time:

- For tables maintained by CICS, information is held in the tables themselves.
- For recoverable auxiliary temporary storage, CICS maintains information on the system log about all new items written to TS queues. CICS maintains information about TS queues for backout purposes in main storage.
- For transient data, CICS maintains cursors that indicate how much has been read and written to the queue, and these cursors are logged. CICS does not log before- or after-images for transient data.
- For CICS files, the before-images of deleted or changed records are recorded in the system log. Although they are not strictly "before-images", CICS also logs newly added records, because CICS needs information about them if they have to be removed during backout.

Files

CICS file control is presented with the log records of all the recoverable files that have to be backed out.

File control performs the following processing:

- Restores the before-images of updated records
- Restores deleted records
- Removes new records added by the unit of work

If backout fails for any file-control-managed resources, file control invokes backout failure support before the unit of work is marked as backout-failed. See "Backout-failed recovery" on page 79.

BDAM files and VSAM ESDS files:

In the special case of the file access methods that do not support delete requests (VSAM ESDS and BDAM) CICS cannot remove new records added by the unit of work.

In this case, CICS invokes the global user exit program enabled at the XFCLDEL exit point whenever a WRITE to a VSAM ESDS, or to a BDAM data set, is being backed out. This enables your exit program to perform a logical delete by amending the record in some way that flags it as deleted.

If you do not have an XFCLDEL exit program, CICS handles the unit of work as backout-failed, and shunts the unit of work to be retried later (see "Backout-failed recovery" on page 79). For information about resolving backout failures, see Logical delete not performed.

Such flagged records can be physically deleted when you subsequently reorganize the data set offline with a utility program.

CICS data tables:

For CICS-maintained data tables, the updates made to the source VSAM data set are backed out. For user-maintained data tables, the in-storage data is backed out.

Intrapartition transient data

Intrapartition destinations specified as **logically recoverable** are restored by transaction backout. Read and write pointers are restored to what they were before the transaction failure occurred.

Physically recoverable queues are recovered on warm and emergency restarts.

Transient data does not provide any support for the concept of transaction backout, which means that:

- Any records retrieved by the abending unit of work are not available to be read by another task, and are therefore lost.
- Any records written by the abending unit of work are not backed out. This means that these records **are** available to be read by other tasks, although they might be invalid.

CICS does not support recovery of extrapartition queues.

Auxiliary temporary storage

CICS transaction backout backs out updates to auxiliary temporary storage queues if they are defined as recoverable in a temporary storage table. Read and write pointers are restored to what they were before the transaction failure occurred.

CICS does not back out changes to temporary storage queues held in main storage or in a TS server temporary storage pool.

START requests

Recovery of EXEC CICS START requests during transaction backout depends on some of the options specified on the request. The options that affect recoverability are:

PROTECT
: This option effectively causes the start request to be treated like any other recoverable resource, and the request is committed only when the task issuing the START takes a syncpoint. It ensures that the new task cannot be attached for execution until the START request is committed.

FROM, QUEUE, RTERMID, RTRANSID
: These options pass data to the started task using temporary storage.

When designing your applications, consider the recoverability of data that is being passed to a started transaction.

Recovery of START requests during transaction backout is described below for different combinations of these options.

START with no data (no PROTECT)
: Transaction backout does not affect the START request. The new task will start at its specified time (and could already be executing when the task issuing the START command is backed out). Abending the task that issued the START does not abend the started task.

START with no data (PROTECT)
: Transaction backout of the task issuing the START command causes the START request also to be backed out (canceled). If the abended transaction is restarted, it can safely reissue the START command without risk of duplication.

START with recoverable data (no PROTECT)
: Transaction backout of the task issuing the START also backs out the data

intended for the started task, but does **not** back out the START request itself. Thus the new task will start at its specified time, but the data will not be available to the started task, to which CICS will return a NOTFND condition in response to the RETRIEVE command.

START with recoverable data (PROTECT)
Transaction backout of the task issuing the START command causes the START request and the associated data to be backed out. If the abended transaction is restarted, it can safely reissue the START command without risk of duplication.

START with nonrecoverable data (no PROTECT)
Transaction backout of the task issuing the START does **not** back out either the START request or the data intended for the (canceled) started task. Thus the new task will start at its specified time, and the data will be available, regardless of the abend of the issuing task.

START with nonrecoverable data (PROTECT)
Transaction backout of the task issuing the START command causes the START request to be canceled, but not the associated data, which is left stranded in temporary storage.

Note: Recovery of temporary storage (whether or not PROTECT is specified) does not cause the new task to start immediately. (It may qualify for restart like any other task, if RESTART(YES) is specified on the transaction resource definition.) On emergency restart, a started task is restarted only if it was started with data written to a **recoverable** temporary storage queue.

Restart of started transactions:

Non-terminal START transactions that are defined with RESTART(YES) are eligible for restart in certain circumstances only.

The effect of RESTART(NO) and RESTART(YES) on started transactions is shown in Table 1.

Table 1. Effect of RESTART option on started transactions

Description of non-terminal START command	Events	Effect of RESTART(YES)	Effect of RESTART(NO)
Specifies either recoverable or nonrecoverable data	Started task ends normally, but does *not* retrieve data.	START request and its data (TS queue) are discarded at normal end.	START request and its data (TS queue) are discarded at normal end.
Specifies recoverable data	Started task abends *after* retrieving its data	START request and its data are recovered and restarted, up to n^1 times.	START request and its data are discarded.
Specifies recoverable data	Started task abends *without* retrieving its data	START request and its data are recovered and restarted, up to n^1 times.	START request and its data are discarded.
Specifies nonrecoverable data	Started task abends *after* retrieving its data	START request is discarded and not restarted.	Not restarted.

Table 1. Effect of RESTART option on started transactions (continued)

Description of non-terminal START command	Events	Effect of RESTART(YES)	Effect of RESTART(NO)
Specifies nonrecoverable data	Started task abends *without* retrieving its data	Transaction is restarted with its data still available, up to n[1] times.	START request and its data are discarded.
Without data	Started task abends	Transaction is restarted up to n[1] times.	—

[1] n is defined in the transaction restart program, DFHREST, where the CICS-supplied default is 20.

Note: Channel data is not recoverable, therefore channels are not available on restart. Transactions with large channels, for example, greater than 32 KB, cannot be restarted.

EXEC CICS CANCEL requests

Recovery from CANCEL requests during transaction backout depends on whether the data is being passed to the started task and if the temporary storage queue used to pass the data is defined as recoverable.

During transaction backout of a failed task that has canceled a START request that has recoverable data associated with it, CICS recovers both the temporary storage queue and the start request. 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, neither the canceled started task nor its data is recovered, and it stays canceled.

Basic mapping support (BMS) messages

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 making the BMS temporary storage queues recoverable, by defining their DATAIDs in the temporary storage table. For more information about the temporary storage table, see the *CICS 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 backout fails, CICS does not try to restart regardless of the setting of the restart program.

Backout-failed recovery

Backout failure support is currently provided only by CICS file control.

If backout to a VSAM data set fails for any reason, CICS performs the following processing:

- Invokes the backout failure global user exit program at XFCBFAIL, if this exit is enabled. If the user exit program chooses to bypass backout failure processing, the remaining actions below are not taken.
- Issues message DFHFC4701, giving details of the update that has failed backout, and the type of backout failure that has occurred.
- Converts the active exclusive locks into retained locks. This ensures that no other task in any CICS region (including the region that owns the locks) waits for a lock that cannot be granted until the failure is resolved. (In this situation, CICS returns the LOCKED condition to other tasks that request a lock.) Preserving locks in this way also prevents other tasks from updating the records until the failure is resolved.
  - For data sets open in RLS mode, CICS requests SMSVSAM to retain the locks.
  - For VSAM data sets open in non-RLS mode, the CICS enqueue domain provides an equivalent function.

  Creating retained locks also ensures that other requests do not have to wait on the locks until the backout completes successfully.
- Keeps the log records that failed to be backed out (by shunting the unit of work) so that the failed records can be presented to file control again when backout is retried. (See "Shunted units of work" on page 13 for more information about shunted units of work.

If a unit of work updates more than one data set, the backout might fail for only one, or some, of the data sets. When this occurs, CICS converts to retained locks only those locks held by the unit of work for the data sets for which backout has failed. When the unit of work is shunted, CICS releases the locks for records in data sets that are backed out successfully. The log records for the updates made to the data sets that fail backout are kept for the subsequent backout retry. CICS does not keep the log records that are successfully backed out.

For a given data set, it is not possible for some of the records updated by a unit of work to fail backout and for other records not to fail. For example, if a unit of work updates several records in the same data set, and backout of one record fails, they are all deemed to have failed backout. The backout failure exit is invoked once only within a unit of work, and the backout failure message is issued once only, for each data set that fails backout. However, if the backout is retried and fails again, the exit is reinvoked and the message is issued again.

For BDAM data sets, there is only limited backout failure support: the backout failure exit, XFCBFAIL, is invoked (if enabled) to take installation-defined action, and message DFHFC4702 is issued.

Auxiliary temporary storage

All updates to recoverable auxiliary temporary storage queues are managed in main storage until syncpoint. TS always commits forwards; therefore TS can never suffer a backout failure.

Transient data

All updates to logically recoverable intrapartition queues are managed in main storage until syncpoint, or until a buffer must be flushed because all buffers are in use. TD always commits forwards; therefore, TD can never suffer a backout failure on DFHINTRA.

Retrying backout-failed units of work

Backout retry for a backout-failed data set either can be driven manually (using the **SET DSNAME RETRY** command) or in many situations occurs automatically when the cause of the failure has been resolved.

When CICS performs backout retry for a data set, any backout-failed units of work that are shunted because of backout failures on that data set are unshunted, and the recovery manager passes the log records for that data set to file control. File control attempts to back out the updates represented by the log records and, if the original cause of the backout failure is now resolved, the backout retry succeeds. If the cause of a backout failure is not resolved, the backout fails again, and backout failure support is reinvoked.

Disposition of data sets after backout failures

Because individual records are locked when a backout failure occurs, CICS need not set the entire data set into a **backout-failed** condition.

CICS may be able to continue using the data set, with only the locked records being unavailable. Some kinds of backout failure can be corrected without any need to take the data set offline (that is, without needing to stop all current use of the data set and prevent further access). Even for those failures that cannot be corrected with the data set online, it may still be preferable to schedule the repair at some future time and to continue to use the data set in the meantime, if this is possible.

Possible reasons for VSAM backout failure

There are many reasons why back out can fail, and these are described in this topic. In general, each of these descriptions corresponds with a REASON returned on an INQUIRE UOWDSNFAIL command.

I/O error

You must take the data set offline to repair it, but there may be occasions when the problem is localized and use of the data set can continue until it is convenient to carry out the repair.

Message DFHFC4701 with a failure code of X'24' indicates that an I/O error (a physical media error) has occurred while backing out a VSAM data set. This indicates that there is some problem with the data set, but it may be that the problem is localized. A better indication of the state of a data set is given by message DFHFC0157 (followed by DFHFC0158), which CICS issues whenever an I/O error occurs (not just during backout). Depending on the data set concerned, and other factors, your policy may be to repair the data set:

- After a few I/O errors
- After the first backout failure

- After a number of I/O errors that you consider to be significant
- After a significant number of backout failures
- Not at all

It might be worth initially deciding to leave a data set online for some time after a backout failure, to evaluate the level of impact the failures have on users.

To recover from a media failure, re-create the data set by applying forward recovery logs to the latest backup. The steps you take depend on whether the data set is opened in RLS or non-RLS mode:

- For data sets opened in non-RLS mode, set the data set offline to all CICS applications by closing all open files against the data set.

  Perform forward recovery using a forward recovery utility.

  When the new data set is ready, use the CEMT (or EXEC CICS) SET DSNAME RETRY command to drive backout retry against the data set for all the units of work in backout-failed state.
- For data sets opened in RLS mode, use the CEMT (or EXEC CICS) SET DSNAME QUIESCED command to quiesce the data set.

  Perform forward recovery using CICSVR as your forward recovery utility.

  CICS regions are notified through the quiesce protocols when CICSVR has completed the forward recovery. This causes backout to be automatically retried. The backout retry fails at this attempt because the data set is still quiesced, and the UOWs are again shunted as backout-failed.

  Unquiesce the data set as soon as you know that forward recovery is complete. Completion of the unquiesce is notified to the CICS regions, which causes backout to be automatically retried again, and this time it should succeed.

  This mechanism, in which the backout retry is performed within CICS, supersedes the batch backout facility supported by releases of CICSVR earlier than CICSVR 2.3. You do not need a batch backout utility.

Logical delete not performed

This error occurs if, during backout of a write to an ESDS, the XFCLDEL logical delete exit was either not enabled, or requested that the backout be handled as a backout failure.

You can correct this by enabling a suitable exit program and manually retrying the backout. There is no need to take the data set offline.

Open error

Investigate the cause of any error that occurs in a file open operation. A data set is normally already open during dynamic backout, so an open error should occur only during backout processing if the backout is being retried, or is being carried out following an emergency restart. Some possible causes are:

- The data set has been quiesced, in which case the backout is automatically retried when the data set is unquiesced.
- It is not possible to open the data set in RLS mode because the SMSVSAM server is not available, in which case the backout is automatically retried when the SMSVSAM server becomes available.

For other cases, manually retry the backout after the cause of the problem has been resolved. There is no need to take the data set offline.

SMSVSAM server failure

This error can occur only for VSAM data sets opened in RLS access mode. The

failure of the SMSVSAM server might be detected by the backout request, in which case CICS file control starts to close the failed SMSVSAM control ACB and issues a console message. If the failure has already been detected by some other (earlier) request, CICS has already started to close the SMSVSAM control ACB when the backout request fails.

The backout is normally retried automatically when the SMSVSAM server becomes available. (See "Dynamic RLS restart" on page 37.) There is no need to take the data set offline.

SMSVSAM server recycle during backout
This error can occur only for VSAM data sets opened in RLS access mode.

This is an extremely unlikely cause of a backout failure. CICS issues message DFHFC4701 with failure code X'C2'. Retry the backout manually: there is no need to take the data set offline.

Coupling facility cache structure failure
This error can occur only for VSAM data sets opened in RLS access mode. The cache structure to which the data set is bound has failed, and VSAM has been unable to rebuild the cache, or to re-bind the data set to an alternative cache.

The backout is retried automatically when a cache becomes available again. (See "Cache failure support" on page 88.) There is no need to take the data set offline.

DFSMSdss non-BWO backup in progress
This error can occur only for VSAM data sets opened in RLS access mode.

DFSMSdss makes use of the VSAM quiesce protocols when taking non-BWO backups of data sets that are open in RLS mode. While a non-BWO backup is in progress, the data set does not need to be closed, but updates to the data set are not allowed. This error means that the backout request was rejected because it was issued while a non-BWO backup was in progress.

The backout is retried automatically when the non-BWO backup completes.

Data set full
The data set ran out of storage during backout processing.

Take the data set offline to reallocate it with more space. (See Chapter 16, "Moving recoverable data sets that have retained locks," on page 183 for information about preserving retained locks in this situation.) You can then retry the backout manually, using the CEMT, or EXEC CICS, SET DSNAME(...) RETRY command.

Non-unique alternate index full
Take the data set offline to rebuild the data set with a larger record size for the alternate index. (See Chapter 16, "Moving recoverable data sets that have retained locks," on page 183 for information about preserving retained locks in this situation.) You can then retry the backout manually, using the CEMT, or EXEC CICS, SET DSNAME(...) RETRY command.

Deadlock detected
This error can occur only for VSAM data sets opened in non-RLS access mode.

This is a transient condition, and a manual retry should enable backout to complete successfully. There is no need to take the data set offline.

Duplicate key error
The backout involved adding a duplicate key value to a unique alternate index. This error can occur only for VSAM data sets opened in non-RLS access mode.

This situation can be resolved only by deleting the rival record with the duplicate key value.

Lock structure full error
The backout required VSAM to acquire a lock for internal processing, but it was unable to do so because the RLS lock structure was full. This error can occur only for VSAM data sets opened in RLS access mode.

To resolve the situation, you must allocate a larger lock structure in an available coupling facility, and rebuild the existing lock structure into the new one. The failed backout can then be retried using SET DSNAME RETRY.

None of the above
If any other error occurs, it indicates a possible error in CICS or VSAM code, or a storage overwrite in the CICS region. Diagnostic information is given in message DFHFC4700, and a system dump is provided.

If the problem is only transient, a manual retry of the backout should succeed.

Commit-failed recovery

Commit failure support is provided only by CICS file control, because it is the only CICS component that needs this support.

A commit failure is one that occurs during the commit stage of a unit of work (either following the prepare phase of two-phase commit, or following backout of the unit of work). It means that the unit of work has not yet completed, and the commit must be retried successfully before the recovery manager can forget about the unit of work.

When a failure occurs during file control's commit processing, CICS ensures that all the unit of work log records for updates made to data sets that have suffered the commit failure are kept by the recovery manager. Preserving the log records ensures that the commit processing for the unit of work can be retried later when conditions are favorable.

The most likely cause of a file control commit failure, from which a unit of work can recover, is that the SMSVSAM server is not available when file control is attempting to release the RLS locks. When other SMSVSAM servers in the sysplex detect that a server has failed, they retain all the active exclusive locks held by the failed server on its behalf. Therefore, CICS does not need to retain locks explicitly when a commit failure occurs. When the SMSVSAM server becomes available again, the commit is automatically retried.

However, it is also possible for a file control commit failure to occur as a result of some other error when CICS is attempting to release RLS locks during commit processing, or is attempting to convert some of the locks into retained locks during the commit processing that follows a backout failure. In this case it may be necessary to retry the commit explicitly using the SET DSNAME RETRY command. Such failures should be rare, and may be indicative of a more serious problem.

It is possible for a unit of work that has not performed any recoverable work, but which has performed repeatable reads, to suffer a commit failure. If the SMSVSAM server fails while holding locks for repeatable read requests, it is possible to access the records when the server recovers, because all repeatable read locks are released at the point of failure. If the commit failure is not due to a server failure, the locks are held as active shared locks. The INQUIRE UOWDSNFAIL command

distinguishes between a commit failure where recoverable work was performed, and one for which only repeatable read locks were held.

Indoubt failure recovery

The CICS recovery manager is responsible for maintaining the state of each unit of work in a CICS region.

For example, typical events that cause a change in the state of a unit of work are temporary suspension and resumption, receipt of syncpoint requests, and entry into the indoubt period during two-phase commit processing.

The CICS recovery manager shunts a unit of work if all the following conditions apply:

- The unit of work has entered the indoubt period.
- The recovery manager detects loss of connectivity to its coordinator for the unit of work.
- The indoubt attribute on the transaction resource definition under which the unit of work is running specifies WAIT(YES).
- The conditions exist that allow shunting. See ../dfht1/topics/dfht12z.dita for a complete list of conditions.

Files

When file control shunts its resources for the unit of work, it detects that the shunt is being issued during the first phase of two-phase commit, indicating an **indoubt failure**.

Any active exclusive lock held against a data set updated by the unit of work is converted into a retained lock. The result of this action is as follows:

- No CICS region, including the CICS region that obtained the locks, can update the records that are awaiting indoubt resolution because the locks have not been freed.
- Other units of work do not wait on these locked records, because the locks are not active locks but retained locks, requests for which cause CICS to return the LOCKED response.

For information about types of locks, see “Locks” on page 14.

For data sets opened in RLS mode, interfaces to VSAM RLS are used to retain the locks. For VSAM data sets opened in non-RLS mode, and for BDAM data sets, the CICS enqueue domain provides an equivalent function. It is not possible for some of the data sets updated in a particular unit of work to be failed indoubt and for the others not to be.

It is possible for a unit of work that has not performed any recoverable work, but which has performed repeatable reads, to be shunted when an indoubt failure occurs. In this event, repeatable read locks are released. Therefore, for any data set against which only repeatable reads were issued, it is possible to access the records, and to open the data set in non-RLS mode for batch processing, despite the existence of the indoubt failure. The INQUIRE UOWDSNFAIL command distinguishes between an indoubt failure where recoverable work has been performed, and one for which only repeatable read locks were held. If you want to open the data set in non-RLS mode in CICS, you need to resolve the indoubt failure before you can define the file as having RLSACCESS(NO). If the unit of work has updated any other data sets, or any other resources, you should try to resolve the indoubt correctly, but if the unit of work has only performed repeatable

reads against VSAM data sets and has made no updates to other resources, it is safe to force the unit of work using the SET DSNAME or SET UOW commands.

CICS saves enough information about the unit of work to allow it to be either committed or backed out when the indoubt unit of work is unshunted when the coordinator provides the resolution (or when the transaction wait time expires). This information includes the log records written by the unit of work.

When CICS has re-established communication with the coordinator for the unit of work, it can resynchronize all indoubt units of work. This involves CICS first unshunting the units of work, and then proceeding with the commit or backout. All CICS enqueues and VSAM RLS record locks are released, unless a commit failure or backout failure occurs.

For information about the resynchronization process for units of work that fail indoubt, see the CICS Installation Guide.

Intrapartition transient data

When a UOW that has updated a logically recoverable intrapartition transient data queue fails indoubt, CICS converts the locks held against the TD queue to retained locks.

Until the UOW is unshunted, the default action is to reject with the LOCKED condition further requests of the following types:

- **READQ**, if the indoubt UOW had issued READQ or DELETEQ requests
- **WRITEQ**, if the indoubt UOW had issued WRITEQ or DELETEQ requests
- **DELETEQ**, if the indoubt UOW had issued READQ, WRITEQ, or DELETEQ requests

You can use the WAITACTION option on the TD queue resource definition to control the action that CICS takes when an update request is made against a shunted indoubt UOW that has updated the queue. In addition to the default option, which is WAITACTION(REJECT), you can specify WAITACTION(QUEUE) to queue further requests while the queue is locked by the failed-indoubt UOW.

After resynchronization, the shunted updates to the TD queue are either committed or backed out, and the retained locks are released.

Auxiliary temporary storage

When a UOW that has updated a recoverable temporary storage queue fails indoubt, the locks held against the queue are converted to retained locks. Until the UOW is unshunted, further update requests against the locked queue items are rejected with the LOCKED condition.

After resynchronization, the shunted updates to the TS queue are either committed or backed out, and the retained locks are released.

Investigating an indoubt failure

This example shows how to investigate a unit of work (UOW) that has failed indoubt. For the purposes of the example, the CICS-supplied transaction CIND has been used to create the failure - one of the FILEA sample application transactions, UPDT, has been made to fail indoubt.

For more information about CIND, see the *CICS Supplied Transactions*.

To retrieve information about a unit of work (UOW), you can use either the CEMT, or EXEC CICS, INQUIRE UOW command. For the purposes of this illustration, the CEMT method is used. You can filter the command to show only UOWs that are associated with a particular transaction. For example, Figure 4 shows one UOW (AC0CD65E5D990800) associated with transaction UPDT.

```
INQUIRE UOW TRANS(UPDT)
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Uow(AC0CD65E5D990800) Ind Shu Tra(UPDT) Tas(0003155)
    Age(00000680) Ter(S233) Netn(IGBS233 ) Use(CICSUSER) Con Lin(DFHINDSP)
```

Figure 4. The CEMT INQUIRE UOW command showing UOWs associated with a transaction

Each UOW identifier is unique within the local CICS system. To see more information about the UOW, move the cursor to the UOW row and press ENTER. This display the following screen:

```
INQUIRE UOW TRA(UPDT)
RESULT - OVERTYPE TO MODIFY
  Uow(AC0CD65E5D990800)
  Uowstate( Indoubt )
  Waitstate(Shunted)
  Transid(UPDT)
  Taskid(0003155)
  Age(00000826)
  Termid(S233)
  Netname(IGBS233)
  Userid(CICSUSER)
  Waitcause(Connection)
  Link(DFHINDSP)
  Sysid()
  Netuowid(..GBIBMIYA.IGBS233 .0;)r...)
```

Figure 5. CEMT INQUIRE UOW - details of UOW AC0CD65E5D990800

The UOWSTATE for this UOW is **Indoubt**. The TRANSACTION definition attribute WAIT(YES|NO) controls the action that CICS takes when a UOW fails indoubt. CICS does one of two things:

- Makes the UOW wait, pending recovery from the failure. (In other words, the UOW is shunted.) Updates to recoverable resources are suspended,
- Takes an immediate decision to commit or backout the recoverable resource updates.

The WAITSTATE of **Shunted** shows that this UOW has been suspended.

Figure 5 reveals other information about the UOW:

- The original transaction was UPDT, the taskid was 3155, and the termid was S233. Any of these can be used to tie this particular failure with messages written to CSMT.
- The UOW has been indoubt for **826** seconds (AGE).
- The cause of the indoubt failure was a **Connection** failure. (The connection is the dummy connection used by CIND **DFHINDSP**.)

When a UOW has been shunted indoubt, CICS retains locks on the recoverable resources that the UOW has updated. This prevents further tasks from changing the resource updates while they are indoubt. To display CICS locks held by a UOW that has been shunted indoubt, use the CEMT INQUIRE UOWENQ command. You can filter the command to show only locks that are associated with a particular UOW. (Note that the INQUIRE UOWENQ command operates only on non-RLS resources on which CICS has enqueued, and for RLS-accessed resources you should use the INQUIRE UOWDSNFAIL command.) For example:

```
INQUIRE UOWENQ UOW(*0800)
STATUS:  RESULTS
 Uow(AC0CD65E5D990800) Tra(UPDT) Tas(0003155) Ret Dat Own
    Res(DCXISCG.IYLX.FILEA        ) Rle(018) Enq(00000003)
```

Figure 6. CEMT INQUIRE UOWENQ—used to display locks associated with a UOW

To see more information about this UOWENQ, put the cursor alongside it and press ENTER:

```
INQUIRE UOWENQ UOW(*0800)
RESULT
  Uowenq
  Uow(AC0CD65E5D990800)
  Transid(UPDT)
  Taskid(0003155)
  State(Retained)
  Type(Dataset)
  Relation(Owner)
  Resource(DCXISCG.IYLX.FILEA)
  Rlen(018)
  Enqfails(00000003)
  Netuowid(..GBIBMIYA.IGBS233 .O;)r...)
  Qualifier(000001)
  Qlen(006)
```

Figure 7. CEMT INQUIRE UOWENQ—details of a lock associated with a UOW

We can now see that:

- This UOW is the **Owner** of a **Retained** lock on a **Dataset**. Retained locks differ from active locks in that a further task requiring this lock is not suspended; instead, the transaction receives the LOCKED condition—if the condition is not handled by the application, this results in an AEX8 abend.
- The data set is **DCXISCG.IYLX.FILEA**, and the **Qualifier** (in this case, the key of the record which is indoubt) is **000001**.
- Three other tasks have attempted to update the indoubt record (ENQFAILS).

Because CIND was used to create this indoubt failure, it can also be used to resolve the indoubt UOW. For an example of how to resolve a real indoubt failure, see the *CICS Intercommunication Guide*.

Recovery from failures associated with the coupling facility

This topic deals with recovery from failures arising from the use of the coupling facility, and which affect CICS units of work.

It covers:

- SMSVSAM cache structure failures
- SMSVSAM lock structure failures (lost locks)
- Connection failure to a coupling facility cache structure
- Connection failure to a coupling facility lock structure
- MVS system recovery and sysplex recovery

Cache failure support

This type of failure affects only data sets opened in RLS mode.

SMSVSAM supports **cache set** definitions that allow you to define multiple cache structures within a cache set across one or more coupling facilities. To ensure against a cache structure failure, use at least two coupling facilities and define each cache structure, within the cache set, on a different coupling facility.

In the event of a cache structure failure, SMSVSAM attempts to rebuild the structure. If the rebuild fails, SMSVSAM switches data sets that were using the failed structure to use another cache structure in the cache set. If SMSVSAM is successful in either rebuilding or switching to another cache structure, processing continues normally, and the failure is transparent to CICS regions. Because the cache is used as a store-through cache, no committed data has been lost.

The support for rebuilding cache structures enables coupling facility storage to be used effectively. It is not necessary to reserve space for a rebuild to recover from a cache structure failure—SMSVSAM uses any available space.

If RLS is unable to recover from the cache failure for any reason, the error is reported to CICS when it tries to access a data set that is bound to the failed cache, and CICS issues message DFHFC0162 followed by DFHFC0158. CICS defers any activity on data sets bound to the failed cache by abending units of work that attempt to access the data sets. When "cache failed" responses are encountered during dynamic backout of the abended units of work, CICS invokes backout failure support (see "Backout-failed recovery" on page 79). RLS open requests for data sets that must bind to the failed cache, and RLS record access requests for open data sets that are already bound to the failed cache, receive error responses from SMSVSAM.

When either the failed cache becomes available again, or SMSVSAM is able to connect to another cache in a data set's cache set, CICS is notified by the SMSVSAM quiesce protocols. CICS then retries all backouts that were deferred because of cache failures.

Whenever CICS is notified that a cache is available, it also drives backout retries for other types of backout failure, because this notification provides an opportunity to complete backouts that may have failed for some transient condition.

1. Cache structure. One of three types of coupling facility data structure supported by MVS. SMSVSAM uses its cache structure to perform buffer pool management across the sysplex. This enables SMSVSAM to ensure that the data in the VSAM buffer pools in each MVS image remains valid.

CICS recovers after a cache failure automatically. There is no need for manual intervention (other than the prerequisite action of resolving the underlying cause of the cache failure).

Lost locks recovery

The failure of a coupling facility lock structure that cannot be rebuilt by VSAM creates the lost locks condition.

The lost locks condition can occur only for data sets opened in RLS mode. A data set can be made available for general use only after all the CICS regions that were accessing the data set at the time of the lock structure failure have completed the process known as lost locks recovery.

Issue the IDCAMS **SHCDS LISTSUBSYS(ALL)** command to display the lost locks and the owners of those locks.

Rebuilding the lock structure

When a coupling facility lock structure fails, the SMSVSAM servers attempt to rebuild their locks in the lock structure from their own locally-held copies of the locks. If this is successful, the failure is transparent to CICS.

About this task

If the rebuild fails, all SMSVSAM servers abend and restart, but they are not available for service until they can successfully connect to a new coupling facility lock structure. Thus a lock structure failure is initially detected by CICS as an SMSVSAM server failure, and CICS issues message DFHFC0153.

When the SMSVSAM servers abend because of this failure, the sharing control data set is updated to reflect the lost locks condition. The sharing control data set records:

- The data sets whose locks have been lost.
- The CICS regions that must complete lost locks recovery for each data set. These are the CICS regions that had a data set open for update (in RLS-mode) for which the locks have been lost.

Notifying CICS of SMSVSAM restart

When the SMSVSAM servers are able to connect to a new lock structure, they use the MVS ENF to notify the CICS regions that their SMSVSAM server is available again.

CICS is informed during dynamic RLS restart about those data sets for which it must perform lost locks recovery. CICS issues a message (DFHFC0555) to inform you that lost locks recovery is to be performed for one or more data sets.

If a lost-locks condition occurs and is not resolved when a CICS restart (warm or emergency) occurs, CICS is notified during file control restart about any data sets for which it must perform lost locks recovery. On a cold start, CICS does not perform any lost locks recovery, and the information in the sharing control data set, which records that CICS must complete lost locks recovery, is cleared for each data set. This does not affect the information recorded for other CICS regions.

Only units of work performing lost locks recovery can use data sets affected by lost locks. Error responses are returned on open requests issued by any CICS

region that was not sharing the data set at the time the lost locks condition occurred, and on RLS access requests issued by any new units of work in CICS regions that were sharing the data set.

Performing lost locks recovery for failed units of work

Lost locks recovery requires that any units of work that had been updating the data set at the time of the failure must complete before the data set can be made available for general use.

This is because their updates are no longer protected by the record locks, so access by other units of work and other CICS regions cannot be permitted until the updates have been either committed or backed out. Therefore, each CICS region performing lost locks recovery must complete all units of work before notifying VSAM that it has completed all affected units of work.

CICS takes the following actions during dynamic RLS restart to expedite lost locks recovery:

- It drives backout-failed units of work for backout retry. If backout retry for a data set in the lost locks condition fails, lost locks recovery cannot complete until either:
  - The backout is completed successfully (by resolving the condition that caused it), or
  - The SET DSNAME RESETLOCKS command is used to abandon the backout, with consequent loss of data integrity.
- CICS drives commit-failed units of work for commit retry. Because the failure of an SMSVSAM server is the only cause of commit-failed units of work, retrying them succeeds when the server becomes available.
- CICS purges in-flight transactions that have updated a data set that is in a lost locks condition.

  Following any failure of the SMSVSAM server, CICS abends and backs out any units of work that had made RLS requests before the failure, and which then attempt further RLS requests when the restarted SMSVSAM server is available, They are not backed out until they make a further request to RLS.

  In the case of an SMSVSAM server failure that is caused by a lock structure failure, this would mean that in-flight units of work could delay the recovery from the lost locks condition until the units of work make further RLS updates. To avoid this potential delay, CICS purges the transactions to expedite lost locks recovery. CICS issues message DFHFC0171 if any in-flight transactions cannot be purged, warning that lost locks recovery could potentially be delayed.
- For indoubt units of work that have updated a data set that is in a lost locks condition, CICS must wait for indoubt resolution before it can complete lost locks recovery, unless the unit of work is forced to complete using the SET DSNAME or SET UOW commands with the BACKOUT, COMMIT, or FORCE options. (See “Choosing data availability over data integrity” on page 177 for the reason why the BACKOUT option has advantages for UOWs that have updated CICS files.)

When a CICS region has completed lost locks recovery for a data set, it informs SMSVSAM. This is done once for each data set. When all CICS regions have informed SMSVSAM that they have completed their lost locks recovery for a particular data set, that data set is no longer in a lost locks condition, and is made available for general access. Although the lost locks condition affects

simultaneously all data sets in use when the lock structure fails, each data set can be restored to service individually as soon as all its sharing CICS regions have completed lost locks recovery.

Connection failure to a coupling facility cache structure

If connection to a coupling facility cache structure is lost, DFSMS™ attempts to rebuild the cache in a structure to which all the SMSVSAM servers have connectivity. If the rebuild is successful, the failure is transparent to CICS.

If DFSMS is unable to recover transparently from a connectivity failure to a coupling facility cache structure, CICS issues message DFHFC0163 (followed by DFHFC0158) on detecting the condition. The recovery process from this failure is the same as for a cache structure failure:

- Errors are returned to CICS when it attempts to access data sets bound to the cache, as a result of which:
  - CICS attempts to back out the unit of work, fails, and shunts the unit of work so that backout can be retried later.
  - CICS converts any locks held by the unit of work against records in the data set to retained locks.
  - CICS is informed by the quiesce protocols when the connectivity is restored.
  - CICS retries backout of the shunted units of work.

Connection failure to a coupling facility lock structure

If an SMSVSAM server loses connectivity to the coupling facility lock structure, and it is not possible to rebuild locks in another lock structure to which all the SMSVSAM servers in the sysplex have access, the SMSVSAM server abends and restarts itself.

CICS issues message DFHFC0153 when it detects that the server is not available for service.

The restarted SMSVSAM is not available for service until it is successfully connected to its coupling facility lock structure. When it does become available, recovery follows dynamic RLS restart in the same way as for any other server failure, because no lock information has been lost.

MVS system recovery and sysplex recovery

When an MVS image fails, all CICS regions and the SMSVSAM server in that image also fail. All RLS locks belonging to CICS regions in the image are retained by SMSVSAM servers in the other MVS systems.

When the MVS image restarts, recovery for all resources is through CICS emergency restart. If any CICS region completes emergency restart before the SMSVSAM server becomes available, it performs dynamic RLS restart as soon as the server is available.

The surviving MVS images should be affected by the failure only to the extent that more work is routed to them. Also, tasks that attempt to access records that are locked by CICS regions in the failed MVS image receive the LOCKED response.

If all the MVS images in a sysplex fail, the first SMSVSAM server to restart reconnects to the lock structure in the coupling facility and converts all the locks into retained locks for the whole sysplex.

Recovery from the failure of a sysplex is just the equivalent of multiple MVS failure recoveries.

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.

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 93 describes the processing that might follow the abnormal termination of a task.

Exit code

Exit code can be written either in programs (separate modules defined in the CSD) or in 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 information about **HANDLE ABEND** commands, see the *CICS Application Programming Reference*..

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 151.

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

- 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 93).

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 topic 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.

Note: If a transaction updates recoverable resources and, therefore, requires transaction backout to be performed in the event of a task abend, the exit code **must** end with an ABEND command to preserve data integrity. If your exit code returns to CICS without an ABEND command, CICS gives control to the next higher logical level as described above, transaction backout is not performed, and changes to recoverable resources are committed.

Abnormal termination of a task

If the exit code ends with an ABEND command, or if there is no active exit code, 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
- Whether backout was successful

Transaction restart

The transaction restart user-replaceable module (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 (for instance, 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 write your own code for DFHREST, see the *CICS Customization Guide*.

Note:

1. CICS invokes DFHREST only when RESTART(YES) is specified in a transaction's resource definition.
2. Ensure that resources used by restartable transactions, such as files, temporary storage, and intrapartition transient data queues, are defined as recoverable.
3. 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 unit of work, and DFHREST requested a restart.
4. CICS keeps statistics on the total number of restarts against each transaction.
5. Emergency restart does not restart any user tasks that were in-flight when CICS abnormally terminated. Instead, recovery manager attaches a special task for each in-flight task that had recovery records in the system log at abnormal termination. This task invokes each resource manager to backout recoverable resources.
6. Making a transaction restartable can involve slightly more overhead, because copies of the TCTUA, COMMAREA, and terminal input/output area (TIOA) have to be kept in case the transaction needs to be restarted. For more information about making transactions restartable, see the *CICS Customization Guide*.
7. In some cases, the benefits of transaction restart can be obtained instead by using the **SYNCPOINT ROLLBACK** command. See the *CICS Distributed Transaction Programming Guide* for information about using this command in an MRO environment.
8. A transaction initiated by terminal input is allowed to restart during shutdown, even if the transaction is defined as SHUTDOWN(DISABLED).

Actions taken at transaction failure

The CICS transaction failure program (TFP) is invoked during abnormal transaction termination, unless the transaction is to be restarted.

The TFP is invoked when the following conditions apply:

- There is a failure during execution and before syncpoint processing has begun.
- There is a failure during syncpoint processing.

The principal action of the transaction failure program is to send, if possible, an abend message to the terminal connected to the abending transaction. It also sends a message to the CSMT transient data queue.

Except for transaction failures that occur during syncpoint processing, and before sending the message to the CSMT queue, the transaction failure program links to the user-replaceable program error program, DFHPEP. The CICS transaction failure program gives control to DFHPEP through a LINK. This LINK occurs after all program-level abend exit code has been executed by the task that abnormally terminates, but before dynamic transaction backout (if any) has been performed.

Processing 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 program check or abend is associated with any domain other than the application domain, you have no further part in processing the error. If the program check or abend is in the application domain, one of the following can occur:

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

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, CICS searches the system recovery table, DFHSRT. The system recovery table contains a set of operating system abend codes that you want CICS to recover from. CICS searches the table looking for the system abend code issued by the system:

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

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

CICS supplies a sample system recovery table, DFHSRT1$, that has a default set of abend codes. You can modify the sample table to define abend codes that suit your own requirements. The source of DFHSRT1$ is supplied in the CICSTS41.CICS.SDFHSAMP library. For more information about the system recovery table, see the *CICS Resource Definition Guide*.

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

Chapter 9. Communication error processing

The types of communication error that can occur include terminal error processing and intersystem communication failures.

Terminal error processing

There are two main CICS programs that participate in terminal error processing. These are the node error program, DFHZNEP, and the terminal error program, DFHTEP.

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

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

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

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

For programming information about coding your own node error programs and terminal error programs, see the *CICS Customization Guide*.

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.

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. It is raised for the current or next terminal control request.

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.

Intersystem communication failures

An intersystem communication failure can be caused by the failure of a CICS region, or the remote system to which it is connected. A network failure can also cause the loss of the connection between CICS and a remote system.

If the failure occurs between syncpoints, CICS and the remote system can back out any updates of recoverable resources either dynamically or following an emergency restart.

If a failure occurs during the syncpointing process while a distributed unit of work is indoubt, CICS handles the unit of work according to the indoubt attributes defined on the transaction resource definition. One possible course of action is to shunt the UOW as failed-indoubt to await resynchronization when communications are restored. When designing applications, remember that if a communications failure occurs and units of work are shunted as failed-indoubt, resources remain locked until after resynchronization.

For information about the resolution of indoubt units of work, see the CICS Installation Guide.

Part 3. Implementing recovery and restart

This part describes the way you implement recovery and restart for CICS regions.

Chapter 10. Planning aspects of recovery

When you are planning aspects of recovery, you must consider your applications, system definitions, internal documentation, and test plans.

Application design considerations

Think about recoverability as early as possible during the application design stages. This topic covers a number of aspects of design planning to consider.

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 remaining questions assume that the application does perform updates.

Question 2: 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 (including databases) 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 modify 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 3: 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, the data sets can 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, consider using DL/I or DB2, which ensure both read and write integrity.

Question 4: Does the application access any confidential data? Files that contain confidential data, and the applications having access to those files, must be clearly identified at this stage. You may need to ensure that only authorized users may access confidential data when service is resumed after a failure, by asking for re-identification in a sign-on message.

2. In the context of these questions, the term “data sets” includes databases.

Question 5: 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 6: Which of the files to be updated are to be regarded as vital? Identify any files that are so vital to the business that they must always be recoverable.

Question 7: How important is data integrity, compared to availability? Consider how long the business can afford to wait for a record that is locked, and weigh this against the risks to data integrity if the normal resynchronization process is overridden.

The acceptable waiting time will vary depending on the value of the data, and the number of users whom you expect to be affected. If the data is very valuable or infrequently accessed, the acceptable waiting time will be longer than if the data is of low value or accessed by many business-critical processes.

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 terminal user's restart procedure will depend largely on what is feasible—for example:

- Must the user be able 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), more complex procedures may be required.
- Even if the user still has 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. The best point could be determined with the aid of a progress transaction, or it could be at some point earlier in the application—even at the start of the transaction. Note: a progress transaction here means one that enables users to determine the last actions performed by the application on their behalf.

These considerations should be in the external design statement.

Question 10: During what periods of the day do users expect online applications 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 "The RLS quiesce and unquiesce functions" on page 168.

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.

About this task

Points to consider are:

- The need for users to re-identify themselves to the system in a sign-on message (dictated by security requirements, as discussed under question 4 in "Questions relating to recovery requirements" on page 101).
- 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.
- How much or how little rekeying will be needed when resuming work (dictated by the feasibility of rekeying data, as discussed under question 9 in "Questions relating to recovery requirements" on page 101).

When designing the user's restart procedure (including the progress transaction, if used) 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 manual methods. If you plan such an approach, 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 an intelligent workstation or a terminal attached to a programmable controller, it may be possible to continue gathering data without access to a CICS region on an MVS host system.

Communications between application and user

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

Decide if you will provide special procedures to overcome communication problems; for example:

- Allow the user 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 the 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.

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 sign-on state of a user is not retained after a persistent sessions restart.

System definitions for recovery-related functions

You are recommended to use a number of system definitions to ensure that your CICS regions provide the required recovery and restart support for your CICS applications.

System recovery table (SRT)

You are recommended to specify a system recovery table on the **SRT** system initialization parameter. If you do not specify a table suffix on this system initialization parameter, it defaults to YES, which means that CICS tries to load an unsuffixed table (which probably won't exist in your load libraries). There is a pregenerated sample table, DFHSRT1$, supplied in CICSTS41.CICS.SDFHLOAD, and if this is adequate for your needs, specify `SRT=1$`. This table adds some extra system abend codes to the built-in list that CICS handles automatically, even if you define only the basic DFHSRT TYPE=INITIAL and TYPE=FINAL macros.

If you want to add additional system or user entries of your own, modify the sample table. For information about modifying an SRT, see the *CICS Resource Definition Guide*.

Resource definitions for recovery

Ensure that your CICS region includes DFHLIST as one of the lists specified on the **GRPLIST** system initialization parameter. Included in DFHLIST are the following groups, which provide basic recovery functions:

- DFHRSEND
- DFHSTAND
- DFHVTAM

For information about the contents of these groups, see the *CICS Resource Definition Guide*. For information about individual resource recoverability, see Chapter 12, "Defining recoverability for CICS-managed resources," on page 123.

System log streams and general log streams

Defining system log streams for each CICS region is the most fundamental of all the recovery requirements. A system log is mandatory to maintain data integrity in the event of transaction or system abends, and to enable CICS to perform warm and emergency restarts.

CICS uses the services of the MVS system logger for all its system and general logging requirements. The CICS log manager writes system log

and general log data to log streams defined to the MVS system logger. For more information, see Chapter 11, "Defining system and general log streams," on page 107.

Files For VSAM files defined to be accessed in RLS mode, define the recovery attributes in the ICF catalog, using IDCAMS. For VSAM files defined to be accessed in non-RLS mode, you can define the recovery attributes in the CSD file resource definition, or in the ICF catalog, providing your level of DFSMS supports this. For BDAM files, you define the recovery attributes in the FCT.

Transient data queues
If you want your intrapartition queues to be recoverable, specify the recovery option in the definition for each queue. See the *CICS Resource Definition Guide* for information about defining the recovery option, RECOVSTATUS.

Temporary storage table
If you want your temporary storage queues to be recoverable, define them in a temporary storage table (TST) using the DFHTST TYPE=RECOVERY macro. When you define your temporary storage queues in a TST, note that the TSAGE parameter influences the recovery characteristics of that temporary storage.

Use the **TST** system initialization parameter to specify the suffix of the temporary storage table that you want CICS to load at initialization.

Program list table (PLT)
Use the DFHPLT macro to create program list tables that name each program executed during initialization or controlled shutdown of CICS.

You specify the **PLTPI** system initialization parameter for the PLT to be used at initialization, and the **PLTSD** system initialization parameter for the PLT to be used at shutdown. You can also specify the shutdown PLT on the CICS shutdown command.

If you have global user exit programs that are invoked for recovery purposes, they must be enabled during the second stage of CICS initialization. You can enable these global user exit recovery programs in application programs specified in the first part of the PLTPI.

See the *CICS Resource Definition Guide* for information about defining program list tables.

Transaction list table (XLT)
There are two ways you can specify transactions that can be initiated from a terminal during the first quiesce stage of normal shutdown:

- Use the DFHXLT macro to create a transaction list table that names the transactions.
- Specify the SHUTDOWN(ENABLED) attribute on the transaction resource definition.

Documentation and test plans

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

Recovery and restart programs and procedures usually relate to exceptional conditions, and can therefore be more difficult to test than those that handle

normal conditions. They should, nevertheless, be tested as far as possible, to ensure that they handle the functions for which they are designed.

CICS facilities, such as the execution diagnostic facility (CEDF) and command interpreter (CECI), can help to create exception conditions and to interpret 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 in 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
- Shunted units of work, caused by communication or backout failures

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

Chapter 11. Defining system and general log streams

All CICS system logging and journaling is controlled by the CICS log manager, which uses MVS system logger log streams to store its output.

About this task

CICS logging and journaling can be divided into four broad types of activity:

System logging
CICS maintains a system log to support transaction backout for recoverable resources. CICS implements system logging automatically, but you can define the log stream as DUMMY to inhibit this function. However, if you specify TYPE(DUMMY) for the system log, you are running without any transaction backout facilities and without any restart capability, and you can start CICS with START=INITIAL only.

CICS also supports programming interfaces that enable you to write your own data to the system log, but **user-written records should be for recovery purposes only.**

See "Defining system log streams" on page 108.

Forward recovery logging
CICS supports forward recovery logs for VSAM data sets.

Forward recovery logging is not automatic—you must specify that you want this facility for your files, and also define the forward recovery log streams.

See "Defining forward recovery log streams" on page 116.

Autojournaling
CICS supports autojournaling of file control data and terminal control messages. Autojournaling is generally used for audit trails.

Autojournaling is not automatic—you must specify that you want this facility for your files and terminal messages, and also define the general log streams to be used.

See the JOURNAL attribute described in FILE and PROFILE resource definition attributes in the *CICS Resource Definition Guide* for information about specifying automatic journaling on FILE and PROFILE resource definitions.

User journaling
CICS supports programming interfaces to enable CICS applications to write user-defined records to user journals, which are held on general log streams.

See the *CICS System Definition Guide* for information about defining general log streams for user journals.

Autojournaling and user journals play no part in CICS recovery and therefore are not discussed here.

The CICS log manager writes the data associated with these logging and journaling activities to two types of MVS log stream:

System log streams
These are used by the CICS log manager and the CICS recovery manager exclusively for unit of work recovery purposes. Each system log is unique to a CICS region, and must not be merged with any other system log.

General log streams
These are used by the CICS log manager for all other types of logging and journaling. You can merge forward recovery records, autojournal records, and user journal records onto the same general log stream, from the same, or from different, CICS regions.

For information on how CICS handles the different error conditions detected by the CICS log manager, see the *CICS Problem Determination Guide*.

Defining log streams to MVS

Log streams are MVS resources that reside in the coupling facility (if one is used), and in data sets.

About this task

All log streams required by CICS must be defined to the MVS system logger before CICS can use them. You can either define log streams explicitly, or you can let CICS create them dynamically when they are first used. To enable CICS to create log streams dynamically, you first define model log streams to the MVS system logger. To define explicit log streams and model log streams, use the MVS IXCMIAPU utility.

For information about defining coupling facility log streams and DASD-only log streams, see the *CICS Transaction Server for z/OS Installation Guide*. For more information about the coupling facility and defining log structures generally, see *z/OS MVS Setting Up a Sysplex*.

Defining system log streams

You must define a system log if you want to preserve data integrity in the event of unit of work failures and CICS failures .

About this task

A system log is mandatory for the following processes:

- The backout of recoverable resources changed by failed units of work
- Cold starts, to enable CICS to restore units of work that were shunted at the time of the shutdown
- Warm restarts, to enable CICS to restore the region to its state at the previous shutdown, including units of work that were shunted at the time of the shutdown
- Emergency restarts, to enable CICS to:
  - Restore the region to its state at the previous shutdown, including units of work that were shunted at the time of the shutdown
  - Recover units of work that were in-flight at the time of the CICS failure, and perform backout of recoverable resources that were updated before the failure

CICS log manager connects to its log stream automatically during system initialization, unless it is defined as TYPE(DUMMY) in a CICS JOURNALMODEL resource definition.

Although the CICS system log is logically a single logical log stream, it is written to two physical log streams—a primary and a secondary. In general, it is not necessary to distinguish between these, and most references are to the system log stream. Using a primary and a secondary log stream for its single system log enables CICS to optimize its use of log stream storage through its process of moving long-running UOWs from the primary to secondary logstreams.

Specifying a JOURNALMODEL resource definition

During a cold start, and a warm and emergency restart, CICS retrieves the names of its system log streams from the global catalog, ensuring that it reconnects to the same log streams that were used on the previous run.

During an initial start, CICS uses default log stream names, unless you specify otherwise on a JOURNALMODEL resource definition. The process of selecting and connecting to a system log stream is as follows:

Without a JOURNALMODEL definition
If CICS can't find a JOURNALMODEL resource definition for DFHLOG and DFHSHUNT, it issues calls to the MVS system logger to connect to its system log streams using default names. These are:

region_userid.applid.DFHLOG
region_userid.applid.DFHSHUNT

where *region_userid* is the RACF® user ID under which the CICS address space is running, and *applid* is the region's VTAM APPL name (taken from the APPLID system initialization parameter). The CICS-supplied JOURNALMODEL definitions for default DFHLOG and DFHSHUNT log streams are in the CSD group DFHLGMOD, which is included in DFHLIST.

If you use these default log stream names, ensure that

- The default log streams are defined explicitly to the MVS system logger, or
- Suitable model log streams are defined for dynamic creation.

With a JOURNALMODEL definition
If CICS finds a JOURNALMODEL definition with JOURNALNAME(DFHLOG) and JOURNALNAME(DFHSHUNT), it issues calls to the MVS system logger to connect to the system log streams named in the definitions. Ensure that the system log stream names are unique to the CICS region.

If you define JOURNALMODEL resource definitions for your system logs, ensure that:

- The log streams named in the JOURNALMODEL are defined to the MVS system logger, or
- Suitable model log streams are defined that enable them to be created dynamically.

If you don't want to use a system log (for example, in a CICS test region), specify JOURNALMODEL resource definitions for DFHLOG and DFHSHUNT with TYPE(DUMMY). Note that running CICS with the system log defined as TYPE(DUMMY) forces you to perform an initial start, and CICS does not support dynamic transaction backout.

Model log streams for CICS system logs

If CICS fails to connect to its system log streams because they have not been defined, CICS attempts to have them created dynamically using model log streams.

To create a log stream dynamically, CICS must specify to the MVS system logger all the log stream attributes needed for a new log stream. To determine these otherwise unknown attributes, CICS requests the MVS system logger to create the log stream using attributes of an existing model log stream definition. If you decide to allow CICS to create log streams dynamically, you are responsible for creating the required model log stream definitions to ensure that dynamic creation succeeds.

It is generally worthwhile setting up model log streams only if:

- Each model log stream will be used to create several log streams
- Each log stream created using a model log stream has similar characteristics consistent with the use of the same coupling facility structure
- You don't know in advance how many CICS regions are going to run in a given MVS image (for example, in a development and test environment).

Otherwise, it is probably less work to define the log streams explicitly using IXCMIAPU. Generally, dynamic creation using model log streams is best suited for test and development purposes, and explicit definition for production regions.

The model log stream names that CICS uses for dynamic creation of its system log streams are of the form &*sysname*.*LSN_last_qualifier*.MODEL, where:

- &*sysname* is the MVS symbol that resolves to the system name of the MVS image
- *LSN_last_qualifier* is the last qualifier of the log stream name as specified on the JOURNALMODEL resource definition.

If you do not provide a JOURNALMODEL resource definition for DFHLOG and DFHSHUNT, or if you use the CICS-supplied definitions in group DFHLGMOD, the model names default to &*sysname*.DFHLOG.MODEL and &*sysname*.DFHSHUNT.MODEL.

Example: If a CICS region issues a request to create a log stream for its primary system log, and CICS is running in an MVS image with a sysid of MV10 and using the default JOURNALMODEL definition, the system logger expects to find a model log stream named MV10.DFHLOG.MODEL.

CICS invokes the XLGSTRM global user exit immediately before calling the MVS system logger to create a log stream. If you don't want to use CICS default values for the creation of a log stream, you can write an XLGSTRM global user exit program to modify the request details, including the model log stream name (in parameter UEPMLSN).

Recovery considerations

If you are using coupling facility log streams, sharing structures between MVS images provides some recovery advantages. If an MVS image or logger address space fails, another surviving MVS image using the same log stream structures (not necessarily the same log streams) is notified of the failure and can start immediate log stream recovery for the log streams used by the failing MVS. Otherwise, recovery is delayed until the next time a system connects to a log stream in the affected structures, or until the failed system logger address space restarts.

However, using model log streams defined with the CICS default name are always assigned to the same structure within an MVS image. This may not give you the best allocation in terms of recovery considerations if you are using structures defined across two or more coupling facilities.

For example, consider a two-way sysplex that uses two coupling facilities, each with one log structure defined for use by CICS system logs, structures LOG_DFHLOG_001 and LOG_DFHLOG_002. In this situation, it is better from a recovery point of view for the CICS regions in each MVS to balance log stream allocations across both log structures. This scenario is illustrated in Figure 8, which shows a 2-way sysplex comprising MVSA and MVSB, with four CICS regions in each MVS. CICSA1 through CICSA4 reside in MVSA, and CICSB1 through CICSB4 reside in MVSB. Each MVS is using two coupling facilities, CF1 and CF2, each of which has one log structure referenced by model log streams. The first and second CICS regions in each MVS use log streams defined in the first log structure, and the third and fourth CICS regions in each MVS use log streams defined in the second log structure. In this scenario, each MVS image has a partner to recover its log streams in the event of an MVS failure.

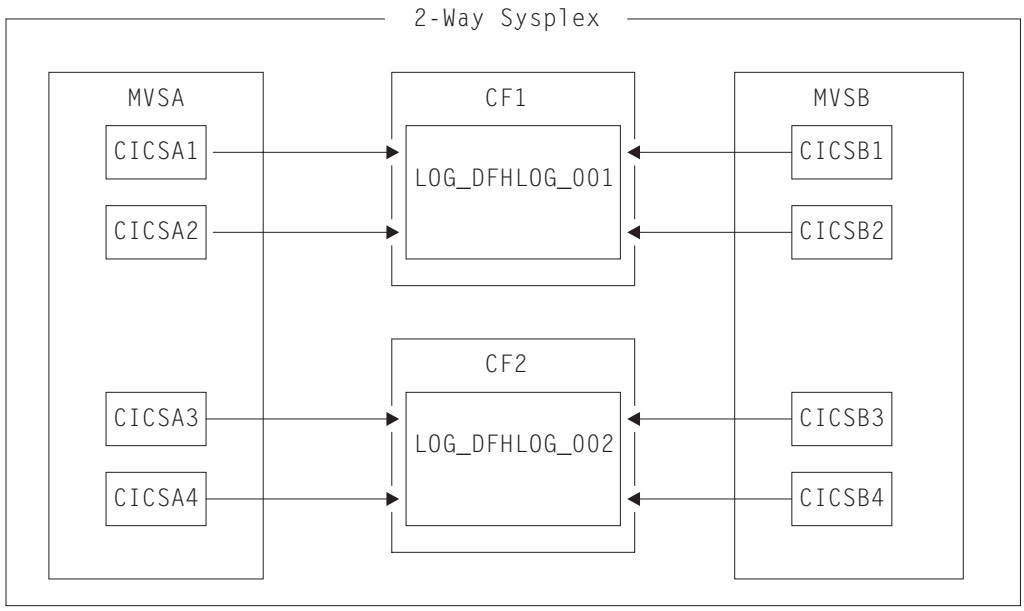


Figure 8. Sharing system logger structures between 2 MVS images

Another example, shown in Figure 9 on page 112, is based on a 4-way sysplex comprising MVSA, MVSB, MVSC, and MVSD. In this case, ensure that the CICS regions that normally run on MVSA and MVSB use structure LOG_DFHLOG_001, and that the regions that run on MVSC and MVSD use structure LOG_DFHLOG_002. Thus each MVS image has a partner to recover its log streams in the event of an MVS failure. If a structure fails, the two MVS images using the other structure can take over the workload.

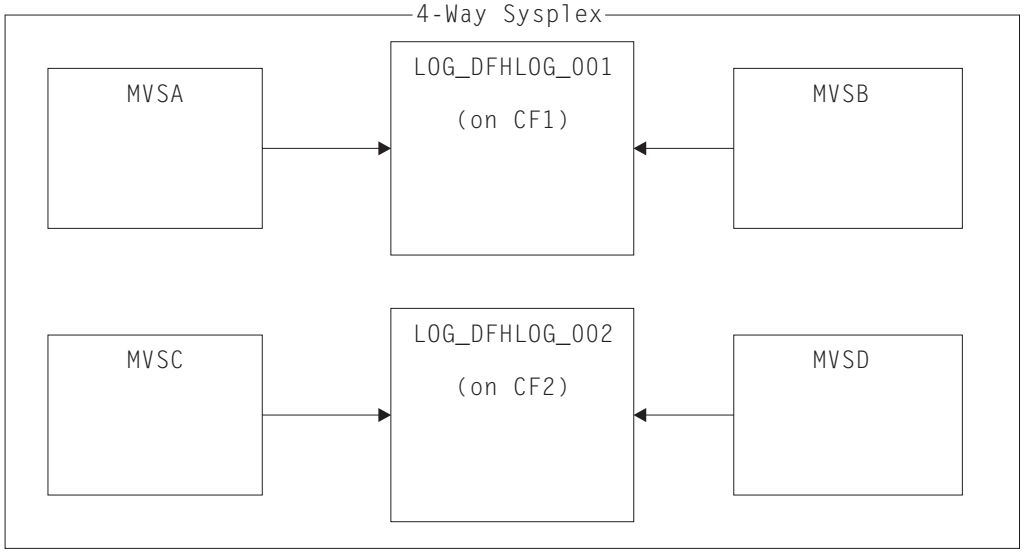


Figure 9. Sharing system logger structures between 4 MVS images

Varying the model log stream name:

To balance log streams across log structures, using model log streams means customizing the model log stream names. You cannot achieve the distribution of log streams shown in this scenario using the CICS default model name.

About this task

You can use an XLGSTRM global user exit program to vary, in a number of ways, the model log stream name to be passed to the system logger. One such way is to store appropriate values in the exit's global work area. For example, you can use the **INITPARM** system initialization parameter to specify a parameter string for use by the exit. This can be retrieved, using the **EXEC CICS ASSIGN INITPARM** command, in the first-phase PLT program that you use to enable the XLGSTRM global user exit program. Having obtained the relevant model log stream information from the INITPARM command, you can store this in the global work area for later use by your XLGSTRM global exit program. Varying the model log stream details in this way enables you to balance log streams across different log structures in a coupling facility.

See the *CICS Customization Guide* for information about writing an XLGSTRM global user exit program, and for information about PLT initialization programs.

Activity keypointing

During a restart, the CICS log manager scans the log backwards to recover unit of work information.

The log is scanned backwards as follows:

1. To begin with, CICS reads all records sequentially as far back as the last complete activity keypoint written before termination. To minimize the time taken for this scan, it is important that you do not specify an activity keypoint frequency zero. For information about setting a suitable activity keypoint frequency for your CICS region, see the *CICS Performance Guide*.
2. When CICS reaches the last-written complete activity keypoint, it extracts all the information relating to in-flight units of work, including indoubt units of

work. With this information, CICS continues reading backwards, but this time reading only the records for units of work that are identified in the activity keypoint. Reading continues until CICS has read all the records for the units of work identified by the activity keypoint.

This process means that completed units of work, including shunted backout-failed and commit-failed units of work, are ignored in this part of the log scan. This is significant for the retrieval of user-written log records. User-written records cannot be presented at the XRCINPT global user exit program (see "Writing user-recovery data" on page 115) if they are part of units of work that CICS skips in this part of the log scan process.

Figure 10 illustrates the way CICS performs log processing during a CICS restart.

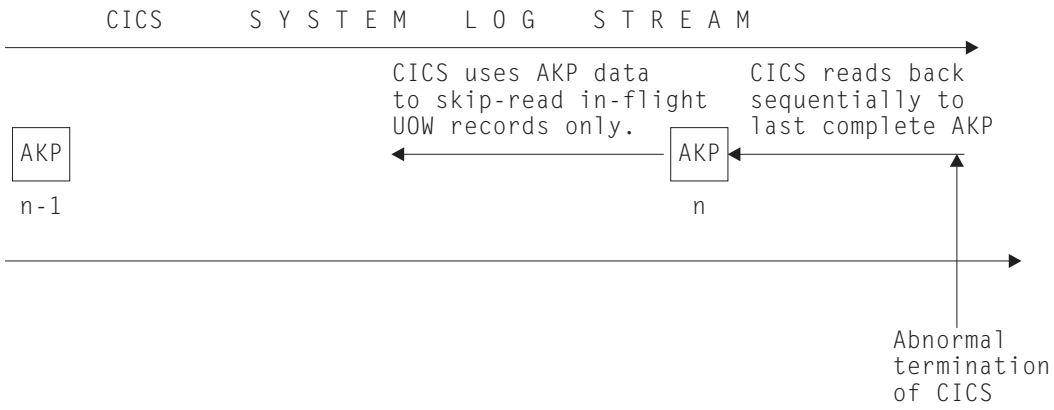


Figure 10. System log scan during restart

Here are some steps you can take to ensure that system log stream sizes, and thus restart times, are kept to a minimum:

- Keep to a minimum the amount of data that has to be read. This means specifying an activity keypoint frequency that is non-zero, and which is:
  - Long enough to avoid excessive keypointing
  - Short enough to avoid large volumes of data between keypoints.

  For information about calculating system log stream structure sizes, see the *CICS Transaction Server for z/OS Installation Guide*.
- Except for your own recovery records that you need during emergency restart, do not write your own data to the system log (for example, audit trail data).
- In particular, do not write information to the system log that needs to be kept. (See "Avoiding retention periods on the system log" on page 115).
- If you write a long-running transaction that updates recoverable resources, ensure that it takes a syncpoint at regular intervals.

Keeping system log data to a minimum

CICS log manager controls the size of the system log stream by regularly deleting the oldest completed unit of work records (**log-tail deletion**).

About this task

This operation is associated with activity keypoints. It is important, therefore, that you choose the correct activity keypoint frequency (AKPFREQ) - that is, one that allows CICS to keep the system log down to a reasonable size and to keep it within the coupling facility (if one is used):

- If a system log stream exceeds the primary storage space allocated, it spills onto secondary storage. (For a definition of primary and secondary storage, see the *CICS Transaction Server for z/OS Installation Guide*.) The resulting I/O can adversely affect system performance.
- If the interval between activity keypoints is long, the volume of data could affect restart times. In general, an activity keypoint interval should be longer than the elapsed time of most transactions (excluding those that are in the category of long-running transactions).

Note: Do not specify AKPFREQ=0, because without activity keypoints CICS cannot perform log tail deletion until shutdown, by which time the system log will have spilled onto secondary storage.

Log-tail deletion

The log tail is the oldest end of the log.

At each activity keypoint, the CICS recovery manager requests the log manager to delete the tail of the system log by establishing a point on the system log before which all older data blocks can be deleted. Thus, if the oldest "live" unit of work is in data block **x**, the CICS log manager requests the system logger to delete all data blocks older than *x* (*x-1* and older).

Note: Long-running units of work that regularly initiate writes to the system log can prevent CICS from deleting completed units of work stretching back over many activity keypoints. See "Long-running transactions" on page 116.

Moving units of work to the secondary log:

In a system with an activity keypoint frequency configured for optimum effect, all units of work execute in a short enough time to enable the log-tail deletion process to keep the primary log stream within its primary storage space allocation.

About this task

However, in most systems, some units of work will last longer. This could be due to application design, or the unit of work suffering an indoubt failure, a commit-failure or backout-failure. To prevent these few units of work affecting the whole region, CICS detects that a unit of work has become long-running and makes copies of its log records on the secondary log stream. This allows more of the primary log stream to be trimmed and hence increases the probability of keeping the system log within its primary space allocation. CICS decides that a unit of work is long-running if the UOW does not write to the system log for two complete activity keypoint intervals.

All log records for units of work are initially written on the primary log stream. From this, they are either deleted by the log-tail deletion mechanism, or copied to the secondary log stream. The copy is made during activity keypointing.

After they have been moved to the secondary logstream, the log records for a unit of work remain there until the unit of work is completed. Note that subsequent writes to the system log by the unit of work are directed to the primary log.

Writing user-recovery data
About this task

You should write only recovery-related records to the system log stream. You can do this using the commands provided by the application programming interface (API) or the exit programming interfaces (XPI). This is important because user recovery records are presented to a global user exit program enabled at the XRCINPT exit point.

User-written recovery records are managed in the same way as recovery manager records, and do not prevent CICS from performing log-tail deletion:

- User-written recovery records written by user transactions are treated as part of the unit of work in which they are written—that is, they form part of the UOW chain. CICS shunts user-written recovery records as part of the unit of work, if the unit of work fails for any reason. These are:
  - Recovered by in-flight unit of work recovery during an emergency restart
  - Recovered by shunted unit of work recovery during both warm and emergency restarts.
- User recovery records written by an XAKUSER global user exit program are written as part of an activity keypoint.

Retrieving user records from the system log:

During warm and emergency restarts, CICS scans the system log backwards. If, during this backwards scan, CICS finds active user records, they are presented to a global user exit program enabled at the XRCINPT exit point.

About this task

Active user log records are those that are:

- Written by an XAKUSER global user exit program during the last completed activity keypoint.
- Written by a user transaction through an API command.

  These user records, which form part of a unit of work chain, are deemed to be active only if:
  - The high-order bit of the JTYPEID field is set to 1.

    These records are presented to your XRCINPT global user exit program, regardless of the state of the unit of work, provided the unit of work lies within the scope of the backward scan of the system during restart. This category includes units of work that are in a backout-failed or commit-failed state.
  - The unit of work in which the command was issued was in-flight or failed indoubt when the previous run of CICS terminated.

    These user records are always presented to your XRCINPT global user exit program, regardless of the state of the high-order bit in the JTYPEID field.

If CICS completes its scan of the system log and finds there are no active user records, it issues message DFHER5731.

Avoiding retention periods on the system log

CICS log tail deletion, which ensures that completed log records are retained for two complete activity key points only, is inhibited if you specify a retention period (RETPD=*dddd*) when you define the system log to MVS.

About this task

The *dddd* value specifies the minimum number of days for which data is to be retained on the log.

You are strongly recommended *not* to use the system log for records that need to be kept. Any log and journal data that needs to be preserved should be written to a general log stream. See the *CICS System Definition Guide* for advice on how to create general log stream data sets.

Long-running transactions

Do not design long-running transactions in such a way that they write frequently to the system log stream, in a single unit of work, across multiple activity keypoints.

If a long-running transaction does not take regular syncpoints while regularly initiating writes to the system log, CICS is prevented from purging units of work that completed after the long-running task started.

In Figure 11, the long-running transaction is updating recoverable resources, causing writes to the system log. All the updates are in the same unit of work because it does not take explicit syncpoints. The oldest deletion point is earlier than activity keypoint number 5, preventing CICS from deleting any completed units of work that lie between activity keypoints 5 and 8. In this example, the long-running transaction will eventually cause the system log stream to spill onto secondary storage.

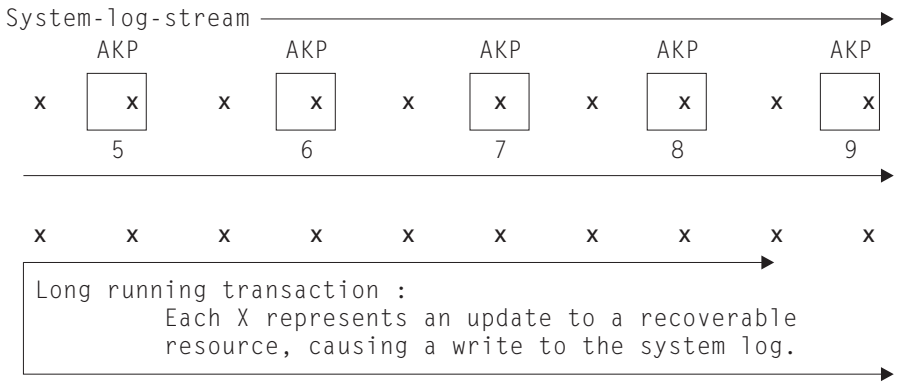


Figure 11. The effect of a 'bad' long-running transaction

Defining forward recovery log streams

You must define forward recovery logs for VSAM data sets that are defined as recoverable files. Neither CICS nor VSAM provides any support for forward recovery logging for a nonrecoverable data set.

About this task

Procedure

1. Define recovery attributes for data sets, including forward recovery, in either the integrated catalog facility (ICF) catalog (if you are using DFSMS 1.3 or later), or in the CICS file resource definition. See "Defining files as recoverable resources" on page 126 for details.

2. Define a general log stream for forward recovery data. If you do not define a general log stream, CICS attempts to create a log stream dynamically. See "Model log streams for CICS general logs" for details.
3. Decide how you want to merge forward recovery data from different CICS regions into one or more log streams. See "Merging data on shared general log streams" on page 118 for details.

What to do next

In the event of physical failure or corruption, restore the most recent backup, and use a forward recovery utility such as CICS VSAM Recovery to reapply updates.

To recover from a storage failure, first restore the most recent backup to a new data set. Then use a forward recovery utility, such as CICS VSAM Recovery (CICSVR), to apply all the updates that were written to a forward recovery log stream after the backup date.

Model log streams for CICS general logs

If CICS fails to connect to a general log stream because it has not been defined, CICS attempts to have it created dynamically.

To create a log stream dynamically, CICS must specify to the MVS system logger all the log stream attributes needed for the new log stream. To determine these otherwise unknown attributes, CICS requests the MVS system logger to create the log stream using attributes of an existing model log stream definition. If you decide to allow CICS to create log streams dynamically, you are responsible for creating the required model log stream definitions to ensure that dynamic creation succeeds.

It is generally worthwhile setting up model log streams only if:

- Each model log stream will be used to create several log streams
- Each log stream created using a model log stream has similar characteristics consistent with the use of the same coupling facility structure
- You don't know in advance the journal names that will be used by CICS applications.

Otherwise, it is probably less work to define the log streams explicitly using IXCMIAPU.

The default model log stream names that CICS uses for dynamic creation of a general log stream are of the form *LSN_qualifier1.LSN_qualifier2*.MODEL where the first and second qualifiers are the CICS region userid and the CICS APPLID, respectively.

Example: If a CICS region, running under userid CICSHA## with APPLID=CICSHAA1, issues a request to create a general log stream that is not defined by a JOURNALMODEL resource definition, CICS specifies CICSHA##.CICSHAA1.MODEL as the model log stream name.

See "Model log streams for CICS system logs" on page 110 for information about using an XLGSTRM global user exit program to modify requests to create a log stream.

Merging data on shared general log streams

Unlike system log streams, which are unique to one CICS region, general log streams can be shared between many CICS regions. This means that you can merge forward recovery data from a number of CICS regions onto the same forward recovery log stream.

About this task

- You can use the same forward recovery log stream for more than one data set. You do not have to define a log stream for each forward-recoverable data set.
- You can share a forward recovery log stream between multiple CICS regions. If the regions are accessing a data set in RLS mode, the use of the same forward recovery log stream is forced.

Your decision on how to define and allocate forward recovery log streams is a trade-off between transaction performance, fast recovery, and having a large number of log streams to manage.

Some things to consider are:

- You can share a coupling facility log stream across multiple CICS regions that run in the same or in different MVS images. You can share a DASD-only log stream only across CICS regions that run in the same MVS image.
- All data sets used by one transaction should use the same log stream (to reduce the number of log streams written to at syncpoint).
- Share a forward recovery log stream between data sets that:
  - Have similar security requirements
  - Have similar backup frequency
  - Are likely to need restoring in their entirety at the same time.
- The last qualifier of the stream name is used as the CICS resource name for dispatcher waits. Therefore, a self-explanatory qualifier can be helpful when you are interpreting monitoring information and CICS trace entries.
- Don't mix frequently updated data sets with infrequently updated data sets on the same forward recovery log, because this causes a disproportionate amount of unwanted log data to be read during recovery of infrequently updated data sets.
- Don't put all frequently updated data sets on a single log stream because you could exceed the throughput capacity of the log stream.
- If you define too many data sets to a single log stream, you could experience frequent structure-full events when the log stream can't keep up with data flow.
- Delete redundant data from log streams periodically, to preserve secondary storage and so that the system logger does not exceed its limit of data sets per log stream. (The limit is several million and in normal circumstances it is not likely to be exceeded.) See the *CICS Transaction Server for z/OS Installation Guide* for information about managing log stream data sets.
- Relate log stream names to the data sets. For example, PAYROLL.data_sets could be mapped to a forward recovery log named PAYROLL.FWDRECOV.PAYLOG.
- Avoid having too many forward recovery log streams, because of coupling facility log structure limits.

Defining the log of logs

Define a log of logs log stream, and define a JOURNALMODEL resource definition that references that log stream.

About this task

The CICS-supplied group, DFHLGMOD, includes a JOURNALMODEL for the log of logs, called DFHLGLOG, which has a log stream name of &USERID..CICSVR.DFHLGLOG. Note that &USERID resolves to the CICS region userid, and if your CICS regions run under different RACF user IDs, the DFHLGLOG definition resolves to a unique log of logs log stream name for each region. However, a separate log of logs is not recommended for recovery products such as CICS VSAM recovery, and you should consider defining a log stream that is shared between CICS regions within the same CICSplex. For example, you might define one log stream for all the CICS regions that are members of the same production CICSplex, and another for test regions, choosing a high-level qualifier that reflects the scope of the log of logs. Using a CICSplex-wide log stream for the log of logs is particularly important if you are using VSAM RLS and multiple CICS application-owning regions have direct access to the same data sets.

The log of logs contains records that are written each time a file is opened or closed. At file-open time, a tie-up record is written that identifies:

- The name of the file
- The name of the underlying VSAM data set
- The name of the forward recovery log stream
- The name of the CICS region that performed the file open.

At file-close time, a tie-up record is written that identifies:

- The name of the file
- The name of the CICS region that performed the file close.

The log of logs assists forward recovery utilities to maintain an index of log data sets.

Log of logs failure

If a log of logs fails, CICS issues a message, but ignores the error.

Further attempts to write to the log of logs are ignored. Forward recovery programs that use the log of logs for performance optimization must not use the log of logs following a failure. The most likely cause of a log of logs failure is a definition error, in which case the failure is unlikely to affect production data sets.

Reading log streams offline

Access to MVS system logger log stream data is provided though the subsystem interface (SSI) with the LOGR subsystem.

About this task

To use the SSI to gain access to the log stream data, specify LOGR on the SUBSYS parameter on the log stream DD statement. Specifying the LOGR subsystem name on the SUBSYS parameter enables LOGR to intercept data set open and read requests at the SSI, and convert them into log stream accesses.

Depending on the options specified on the SUBSYS parameter, general log stream records are presented either:

- In a record format compatible with utility programs written for releases of CICS that use the journal control program, DFHJUP, for logging and journaling

- In a format compatible with utility programs written for versions of CICS that use the log manager for logging and journaling.

See the *CICS Operations and Utilities Guide* for more information about using the LOGR SSI to access log stream data, and for sample JCL.

If you plan to write your own utility program to read log stream data, see the *CICS Customization Guide* for information about log stream record formats.

Effect of daylight 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 daylight saving.

These time changes are also applied to the local time held in computers. Generally, most hardware (TOD) clocks are set to Greenwich Mean Time (GMT), with an offset value to indicate local time. It is this offset value that is adjusted when making daylight saving time changes, while the hardware clock remains unaltered.

Adjusting local time

When setting clocks forward or back an hour to adjust for Summer and Winter time while a CICS region is running, use the **CEMT PERFORM RESET** or **EXEC CICS PERFORM RESETTIME** command to ensure that CICS immediately resynchronizes its local time with that of the MVS TOD clock.

About this task

CICS obtains and stores the *local time offset* at start up, and when the CEMT PERFORM RESET command executes. Use the **CEMT PERFORM RESET** command immediately whenever you change the system date or time-of-day while CICS is running, to ensure that the correct local time is used by all CICS functions, including the API. Whenever an application program issues an **EXEC CICS ASKTIME** command, CICS obtains the current time from the MVS TOD clock, and modifies this by the stored local time difference. CICS then updates the EIBTIME field in the exec interface block with the local time.

Time stamping log and journal records

A local time change, forwards or backwards, has no effect on CICS logging or journaling, or on CICS restarts, but could affect the operation of utility programs such as DFHJUP.

CICS time-stamps the data it writes to the system log as follows:

- System log block headers are time-stamped with both the machine clock (STCK) value and local time
- System log records are time-stamped with the machine clock (STCK) value only.

For general logs, in addition to time-stamping as in system logs, CICS also includes local time in the journal records.

During a restart, for system recovery purposes, CICS reads the youngest—most recently written—record from the primary log stream. Thereafter, CICS uses only direct reads using block ids and does not rely upon time stamps. CICS also uses direct read with block ids to retrieve the logged data for transaction backout purposes, again without any dependence on time stamps.

Operating a recovery process that is independent of time-stamps in the system log data ensures that CICS can restart successfully after an abnormal termination, even if the failure occurs shortly after local time has been put back.

Offline utility program, DFHJUP

Changing the local time forward has no effect on the processing of system log streams or general log streams by the CICS utility program, DFHJUP.

Changing local time backwards will not affect the operation of DFHJUP provided you specify the GMT option on the SUBSYS parameter of the log stream DD statement in the DFHJUP JCL.

However, if you use local time on the SUBSYS parameter to specify the partitioning of a log stream for processing by DFHJUP, you must take steps to ensure the chronological sequence of time-stamps when adjusting clocks backwards. You can do this by stopping CICS regions until the new local time passes the old time at which the change was made.

User- or vendor-written journal utilities and DFHJUP exit programs may also be sensitive to local time changes. These should be checked to ensure that there are no problems posed by backwards time changes.

Forward recovery utilities (but not CICS VSAM Recovery 2.3) may also be sensitive to the time sequence of forward recovery log data. If you are not using CICSVR 2.3, check that your forward recovery utility can handle discontinuities in logged records.

The only circumstances in which forward recovery is jeopardized, while using CICSVR 2.3, are when you:

- Take a backup during the hour following the (local) time change (for example, at 01.30 after setting the time back from 02.00 to 01.00)
- Perform forward recovery using that backup with log records that span the time change
- Specify local time instead of GMT to CICSVR.

If you use a backup taken earlier than the new local time, or if you specify GMT, CICSVR handles forward recovery successfully.

Chapter 12. Defining recoverability for CICS-managed resources

This section describes what to do to ensure that you can recover the resources controlled by CICS on behalf of your application programs.

About this task

It covers the recoverability aspects of the various resources as follows:

- "Recovery for transactions"
- "Recovery for files" on page 125
- "Recovery for intrapartition transient data" on page 131
- "Recovery for extrapartition transient data" on page 134
- "Recovery for temporary storage" on page 135

Recovery for transactions

CICS recoverability options are limited for local and remote transactions.

The following options apply for local transactions:

- Automatic restart when a transaction abends
- Time-out when a transaction has been waiting for resources for longer than a specified time interval, usually because the transaction is deadlocked with another transaction for the same resources
- Allowing you to purge transactions that "hang" for reasons that cannot otherwise be resolved

For remote transactions, in addition to the above, CICS provides indoubt options to specify the recovery action in the event of indoubt failures.

Defining transaction recovery attributes

When defining user transactions that update **local** resources, you can specify the following options on the TRANSACTION resource definition for recovery purposes:

About this task

RESTART({NO|YES})
This option defines whether, in some circumstances, the transaction is eligible to be restarted automatically by CICS under the control of the DFHREST user replaceable module.

The default is RESTART(NO).

DTIMOUT({NO|1–6800})
If the task remains suspended (inactive) for the specified interval (and SPURGE(YES) is also specified), CICS initiates an abnormal termination of the task. (Specifying SPURGE(NO) overrides the DTIMOUT option.)

The default is DTIMOUT(NO).

SPURGE({NO|YES})
This option indicates whether the transaction is initially system-purgeable; that is, whether CICS can purge the transaction as a result of:

- Expiry of a deadlock timeout (DTIMOUT) delay interval
- A CEMT, or EXEC CICS, SET TASK(id) PURGE|FORCEPURGE command.

The default is SPURGE(NO).

TPURGE({NO|YES})
This option indicates whether the transaction is system-purgeable in the event of a (non-VTAM) terminal error.

The default is TPURGE(NO).

Indoubt options for distributed transactions

When defining user transactions that update **remote** resources, you can specify the following options for remote recovery purposes:

ACTION({BACKOUT|COMMIT})
Specifies the action to be taken when a failure occurs during two-phase commit processing, after the unit of work has entered the indoubt period.

If you've specified WAIT(YES), the action is not taken unless the interval specified on WAITTIME expires before recovery occurs.

Whether you specify BACKOUT or COMMIT is likely to depend on the kinds of changes made to resources in the remote system.

WAIT({YES|NO})
Specifies whether a failed indoubt unit of work is to wait, pending recovery from the failure, to resolve its indoubt state, or whether CICS is to take immediately the action specified by the ACTION option.

Note: A WAIT(YES) option can be overridden by WAIT(NO) defined on an intrapartition transient data queue. If a TD queue that is access by a transaction specifies WAIT(NO), and the transaction specifies WAIT(YES), the TD queue definition forces the transaction to either backout or commit, as specified by the ACTION attribute on the transaction resource definition.

WAITTIME({00,00,00|dd,hh,mm})
Specifies, if WAIT(YES), how long the transaction is to wait before taking the action specified by ACTION.

You can use WAIT and WAITTIME to allow an opportunity for normal recovery and resynchronization to take place, while ensuring that a transaction commits or backs out within a reasonable time. See the *CICS Intercommunication Guide*for information about defining wait times for distributed transactions.

For more information about recovery of distributed units of work, see the *CICS Intercommunication Guide*.

For more information about options on the TRANSACTION definition, see the *CICS Resource Definition Guide*.

Recovery for files

A CICS **file** is a logical view of a physical data set, defined to CICS in a file resource definition with an 8-character file name.

A CICS file is associated with a VSAM or BDAM data set by one of the following:

- By dynamic allocation, where the data set name is predefined on the DSNAME parameter in the file definition (or set by a CEMT, or EXEC CICS, SET FILE DSNAME(name) command)
- By allocation by JES at job step initiation, where the data set name is defined on a DD statement

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).

When designing your applications, ensure that application data is protected from transaction failures that could lead to data corruption, and that you can recover from accidental damage to storage devices.

When deciding on the access method, consider the recovery and restart facilities provided by each. These considerations are discussed in the following topics.

VSAM files

CICS file control supports three VSAM access modes—local shared resources (LSR), non-shared resources (NSR), and record-level sharing (RLS).

Sharing data sets with batch jobs

Sharing data sets directly between online CICS update transactions and batch update programs using VSAM share options (where available) or job control sharing is not recommended for non-RLS access modes. Sharing risks the integrity of the data with the result that application programs may appear to function correctly, but with incorrect data. Such loss of data integrity can occur, for example, if a CICS unit of work updates a record that is later updated by a non-CICS job while the CICS unit of work is still running. If the CICS unit of work abends, CICS backs out the record to the value it had at the start of the CICS unit of work, thus destroying the update from the non-CICS job.

File-owning regions and RLS access

To share data sets between more than one CICS region, use the RLS access mode or use a file-owning region (FOR) with function shipping from the application-owning regions (AORs). From a recovery point of view, RLS does not create distributed units of work, as happens between the AOR and FOR, because files accessed in RLS mode are all regarded as local to each CICS region. The SMSVSAM server takes the place of the FOR. However, there are special recovery considerations for data sets accessed in RLS mode, in connection with the role of SMSVSAM and its lock management.

Forward recovery

For VSAM files, you can use a forward recovery utility, such as CICSVR, when online backout processing has failed as a result of some physical damage to the data set. For forward recovery:

- Create backup copies of data sets.
- Record after-images of file changes in a forward recovery stream. CICS does this for you automatically if you specify that you want forward recovery support for the file.
- Write a job to run a forward recovery utility, and keep control of backup data sets and log streams that might be needed as input. CICSVR automatically constructs the forward recovery job for you, using an ISPF dialog interface.

Backward recovery

To ensure that VSAM files can be backward recoverable, you must consider the following points:

- Key-sequenced data sets (KSDS) and both fixed- and variable-length relative record data sets (RRDS):
  - If the files referring to KSDS or RRDS data sets are designated as recoverable with LOG(ALL) specified, CICS can back out any updates, additions, and deletions made by an interrupted unit of work.
  - For information about backout failures, see "Backout-failed recovery" on page 79.
- Entry-sequenced data sets (VSAM-ESDS):
  - New records are added to the end of a VSAM-ESDS. After they have been added, records 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, using an XFCLDEL global user exit program.

    See "Transaction backout" on page 74 for more information.

Basic direct access method (BDAM)

CICS does not support forward recovery for BDAM files. 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.

Defining files as recoverable resources

This section describes how to define the recovery attributes for files managed by CICS file control.

About this task

You specify recovery options, including forward recovery, either in the integrated catalog facility (ICF) catalog (if you are using DFSMS 1.3 or later), or in the CICS FILE resource definition, as follows:

- If your VSAM data sets are accessed by CICS in RLS mode, the recovery attributes must be defined in the ICF catalog.
- If your VSAM data sets are accessed by CICS in non-RLS mode, you can define recovery attributes in either the FILE resource or the ICF catalog. If you use the ICF catalog to define attributes for data sets accessed in non-RLS mode, CICS

uses the ICF catalog entry recovery attributes instead of the FILE resource. To force CICS to use the FILE resource attributes instead of the catalog, set the **NONRLSRECOV** system initialization parameter to FILEDEF.

- You define the recovery attributes for BDAM files in file entries in the file control table (FCT).

VSAM files accessed in non-RLS mode

You can specify support for both forward and backward recovery for VSAM files using the RECOVERY and FWDRECOVLOG options. You define the type of data set backup you want using the **BACKUPTYPE** parameter.

RECOVERY(ALL)

If you specify RECOVERY(ALL), CICS provides both forward and backout recovery for the file. Using the services of the CICS recovery manager and log manager, CICS file control writes:

- Before-images of updated records to the system log stream. These are used to back out file changes following a transaction abend or during an emergency restart after a CICS abnormal termination.
- After-images to the general log stream referenced by the FWDRECOVLOG option. The after-images comprise the following:
  - A write_add_complete record when a new record is added
  - A write_delete record when a record is deleted
  - A write_update record when a record is updated

  File control also writes the following to the forward recovery log:
  - FILE_OPEN tie-up records
  - FILE_CLOSE tie-up records
  - TAKE_KEYPOINT tie-up records
  - Data set BACKUP tie-up records

  See the *CICS Customization Guide* for details of all these forward recovery records.

CICS forward recovery support is totally independent of automatic journaling. However, autojournal records, which are controlled by the JOURNAL and associated JNL*xxx* options on the file resource definition, can be written to the same general log stream as forward recovery data. To do this, specify on the JOURNAL option a journal identifier that maps to the same forward recovery log stream. See the *CICS System Definition Guide* for information about mapping CICS journal identifiers to log streams through journal model resource definitions.

Note: The use of autojournal records for forward recovery purposes is not recommended for VSAM files.

RECOVERY(BACKOUTONLY)

If you specify RECOVERY(BACKOUTONLY), CICS provides backout recovery only, writing only before-images to the system log.

BACKUPTYPE(DYNAMIC)

If you specify BACKUPTYPE(DYNAMIC), CICS supports the DFSMS backup-while-open (BWO) facility, enabling the data set to be backed up while open. If you specify STATIC, all CICS files open against a data set must be closed before it can be backed up. For information about the backup-while-open (BWO) facility, see Chapter 18, “Backup-while-open (BWO),” on page 203.

VSAM files accessed in RLS mode

If you specify file definitions that open a data set in RLS mode, specify the recovery options in the ICF catalog.

The recovery options on the CICS file resource definitions (RECOVERY, FWDRECOVLOG, and BACKUPTYPE) are ignored if the file definition specifies RLS access.

The VSAM parameters LOG and LOGSTREAMID, on the access methods services DEFINE CLUSTER and ALTER commands, determine recoverability for the entire sphere. Locating these recovery parameters in the ICF catalog enforces the same options, for all CICS regions in the sysplex, for all the files opened against a given sphere.

LOG({NONE|UNDO|ALL})
Specifies the type of recovery required for the VSAM sphere. Specify the LOG parameter for data sets that are to be used by CICS in RLS mode.

NONE
The sphere is not recoverable.

UNDO
The sphere is recoverable. CICS must maintain system log records for backout purposes.

ALL
The sphere is recoverable for both backout and forward recovery. CICS must maintain system log records (as for UNDO) and forward recovery log records. If you specify LOG(ALL), also specify LOGSTREAMID to indicate the name of the forward recovery log.

Note: Forward recovery support is available for recoverable files only—you cannot have forward recovery without backout recovery.

LOGSTREAMID(*log_stream_name*)
Specifies the name of the log stream to be used for forward recovery log records when LOG(ALL) is defined. Note that IDCAMS does not check for the presence of the LOGSTREAMID during DEFINE processing. CICS checks for a log stream name when attempting to open a data set defined with LOG(ALL), and the open fails if the log stream is not defined and cannot be created dynamically.

If you omit the LOG parameter when defining your VSAM data sets, recovery is assumed to be UNDEFINED, and the data set cannot be opened in RLS mode. You can also set the UNDEFINED status explicitly by specifying NULLIFY(LOG).

For information about the access methods services DEFINE and ALTER commands, see *z/OS DFSMS: Access Method Services for ICF* and *z/OS DFSMS: Using data sets*.

Inquiring on recovery attributes:

You can use CEMT, or EXEC CICS, INQUIRE FILE and INQUIRE DSNAME commands to determine the recovery options that are specified for files and data sets.

The INQUIRE FILE command shows the options from the CICS file definition until the first file for the data set is opened. If the options are obtained from the ICF catalog when the first file is opened, the ICF catalog values are returned. The

INQUIRE DSNAME command returns values from the VSAM base cluster block (BCB). However, because base cluster block (BCB) recovery values are not set until the first open, if you issue an INQUIRE DSNAME command before the first file is opened, CICS returns NOTAPPLIC for RECOVSTATUS.

BDAM files

You can specify CICS support for backward recovery for BDAM files using the LOG parameter on the DFHFCT TYPE=FILE macro. You can also specify that you want automatic journaling of the various types of file access, using the JREQ and JID parameters of the FCT macro.

LOG=YES
If you specify LOG=YES, CICS provides backout recovery for the file. Using the services of the CICS recovery manager and log manager, CICS file control writes before-images of updated records to the system log stream.

JREQ=request-type(s) and JID=nn
Specify the file requests that you want CICS to journal automatically to the log stream mapped by the journal identifier specified on the JID parameter.

Although CICS does not provide forward recovery support for BDAM files, you can use the autojournal records to provide your own facility. JREQ=(WU,WN) is the equivalent of the CSD file definition parameters JNLUPDATE(YES) combined with JNLADD(BEFORE), providing the necessary images for forward recovery to a journal specified by JID=nn.

For information about defining BDAM file resource definitions, see the *CICS Resource Definition Guide*.

The CSD data set

The CICS system definition (CSD) VSAM data set is unique in that it is a CICS system data set that is managed by CICS file control. For this reason the CSD can't be defined in the CSD, and is defined instead by system initialization parameters.

The recovery options equivalent to RECOVERY, FWDRECOVLOG, and BACKUPTYPE are CSDRECOV, CSDFRLOG, and CSDBKUP respectively. See the *CICS System Definition Guide* for information about defining the CSD.

File recovery attribute consistency checking (non-RLS)

For data sets accessed in non-RLS mode, ensure you have consistent recovery attributes between files that refer to the same base data set cluster or its paths.

The first file open for the base data set determines the base data set recovery attributes, and these are stored in the base cluster block (BCB). If, on a subsequent file open request, CICS detects an inconsistency between the file definition recovery attributes and those stored in the BCB, the open request fails.

See “Inquiring on recovery attributes” on page 128 for information about finding the recovery attributes for files and data sets.

Overriding open failures at the XFCNREC global user exit

CICS provides a global user exit point, XFCNREC, to enable you to continue processing regardless of any inconsistencies in the backout setting for files associated with the same data set.

About this task

If you use XFCNREC to suppress open failures that are a result of inconsistencies in the backout settings, CICS issues a message to warn you that the integrity of the data set can no longer be guaranteed.

Any **INQUIRE DSNAME RECOVSTATUS** command that is issued from this point onward will return NOTRECOVABLE, regardless of the recovery attribute that CICS has previously enforced on the base cluster. This condition remains until you remove the data set base control block (with a **SET DSNAME REMOVE** command), or perform a cold start.

The order in which files are opened for the same base data set determines 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 through an unrecoverable file before the data set is fully recovered.

See the *CICS 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 log before-images for a file request.

CICS takes the actions shown in the following list when opening a file for update processing in non-RLS mode—that is, the file definition specifies RLSACCESS(NO), and the operations parameters specify ADD(YES), DELETE(YES) or UPDATE(YES) (or the equivalent SERVREQ parameters in the FCT entry.) If you set only READ(YES) and/or BROWSE(YES) CICS does not make these consistency checks. These checks are not made at resource definition or install time.

- If a file definition 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 ACB for this AIX path fails.
- If a file is the first to be opened against a base cluster after the last cold start, the recovery options of the file definition are copied into the base cluster block.
- If a file is not the first to be opened for update against a base cluster after the last cold start, the recovery options in the file definition are checked against those copied into the base cluster block by the first open. There are the following possibilities:
  - Base cluster has RECOVERY(NONE):
    - File is defined with RECOVERY(NONE): the open proceeds.
    - File is defined with RECOVERY(BACKOUTONLY): the attempt to open the file fails, unless overridden by an XFCNREC global user exit program, which can allow inconsistencies in backout settings for files that are associated with the same base data set.
    - File is defined with RECOVERY(ALL): the open fails.
  - Base cluster has RECOVERY(BACKOUTONLY):
    - File is defined with RECOVERY(NONE): the attempt to open the file fails unless overridden by an XFCNREC global user exit program to allow inconsistencies in backout settings for files associated with the same base data set.
    - File is defined with RECOVERY(BACKOUTONLY): the open proceeds.

- File is defined with RECOVERY(ALL): the open fails.

– Base cluster has RECOVERY(ALL):
  - File is defined with RECOVERY(NONE): the open fails.
  - File is defined with RECOVERY(BACKOUTONLY): the open fails.
  - File is defined with RECOVERY(ALL): the open proceeds unless FWDRECOVLOG specifies a different journal id from the base cluster, in which case the open fails.

Any failure to open a file against a data set results in a message to the console. If necessary, the recovery options must be changed. To change the recovery attributes (held in the base cluster block) of a VSAM data set, you can use the CEMT, or EXEC CICS, SET DSNAME REMOVE command. This deletes the base cluster block, so CICS has no record of prior recovery settings for the 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, are preserved even when all the files relating to the block are closed, and across warm and emergency restarts.

Implementing forward recovery with user-written utilities

If you use your own forward recovery programs, you must ensure they use the after images from the forward recovery log streams. The use of autojournal records written to a general log stream is not recommended for VSAM forward recovery.

About this task

For details of procedures for performing forward recovery, see Chapter 17, "Forward recovery procedures," on page 187.

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

About this task

For details, see Forward recovery with CICS VSAM Recovery.

Recovery for intrapartition transient data

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

Backward recovery

CICS can recover only intrapartition transient data. The intrapartition data set is a VSAM ESDS data set, with a file name of DFHINTRA.

For more information about allocation and space requirements, see the *CICS System Definition Guide*.) For **extrapartition** transient data considerations, see "Recovery for extrapartition transient data" on page 134.

You must specify the name of every intrapartition transient data queue that you want to be recoverable in the queue definition. The recovery attributes you can specify for an intrapartition transient data queue are:

- Logical
- Physical
- None

Logical recovery

If you request logical recovery on an intrapartition queue definition, changes to a transient data queue by an interrupted unit of work are backed out. Backout occurs dynamically in the case of a task abend, or at a CICS emergency restart in the case of a CICS failure.

As a general rule, you should request logical recoverability. For example, if you make related changes to a set of resources that includes intrapartition transient data, and you want to commit (or back out) all the changes, you require logical recovery.

Physical recovery

Physical recoverability is unique to transient data and is effective on both warm and emergency restarts. By requesting physical recovery on an intrapartition queue definition, you ensure that changes to the queue are committed immediately and, with one exception, are not backed out.

The exception is in the case of the last read from a physically recoverable queue before a unit of work fails. CICS always backs out the *last* read from a physically recoverable transient data queue. In terms of the read and write pointers that CICS maintains for TD queues, this means that the read pointer is reset, but the write pointer never changes. This is illustrated by the diagram in Figure 12 on page 133. The sequence of TD actions in this example, and the subsequent recovery, is as follows:

- The unit of work has read items 1 and 2, leaving the read pointer at item 3.
- The unit of work has written item 4, leaving the write pointer ready for the next item to be written.
- CICS abends and is restarted with an emergency restart.
- As a result of the transient data recovery, the read pointer is reset to item 2, queue items 2, 3, and 4 are still available, and the write pointer is restored.

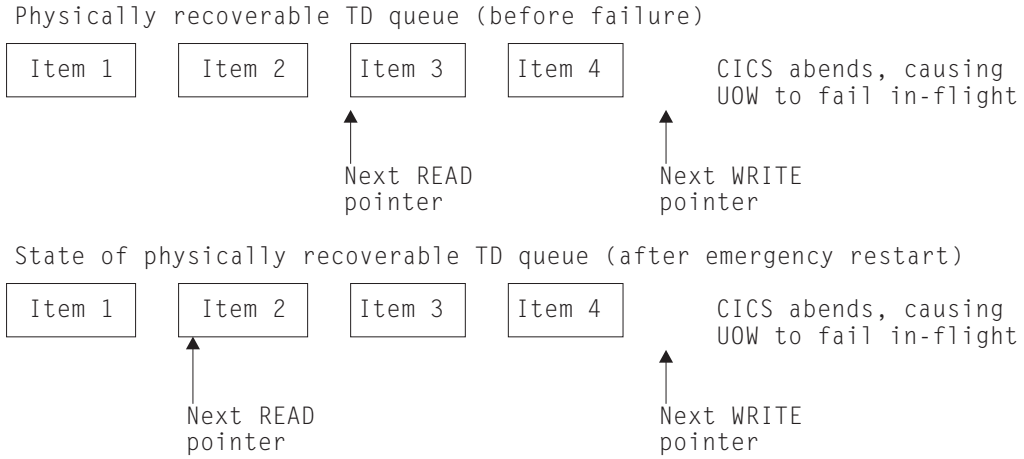


Figure 12. Illustration of recovery of a physically recoverable TD queue

Making intrapartition TD physically recoverable can be useful in the case of some CICS queues. For example, 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 queues 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.

No recovery

Recovery is not performed if you specify NO on the recovery attribute of an intrapartition transient data definition.

Forward recovery

CICS does not provide forward recovery support for transient data.

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

- Each WRITEQ, including the data that is written
- Each READQ
- Each DELETEQ of a queue
- For logically recoverable queues, each backout, syncpoint, or syncpoint rollback

You must provide the application program to rebuild the data by reading the journaled information and applying that information to the transient data queue. Your application program could run in the program list table (PLT) phase or after emergency restart. Until the data set is fully recovered, do not write to the queue, because that would probably result in wrongly-ordered data, and a read 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 perform a cold start with a corrupted intrapartition data set, you will lose the contents of the intrapartition data set. You also lose the contents of the intrapartition data set if you specify TDINTRA=EMPTY as a system initialization parameter.

Recovery for 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 queues should first enqueue exclusive access to those queues. This prevents interleaved access to the same queues 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 queue. 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 queues to permit other application programs to access those queues.

If uncontrolled shutdown occurs before this journaling, no records will appear on the journal data set for that 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.

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 queue 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 you need to recover information recorded before termination.

To avoid losing data in tape output buffers on termination, you can 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 through an automatically initiated task. On 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 post-initialization (PLTPI) programs

You can use initialization (PLTPI) programs as part of the processing required to recover extrapartition transient data or to enable exits required during recovery.

About this task

There are two PLT phases. The first phase occurs before the system initialization task is attached, and must not use CICS resources, because initialization is incomplete. The first phase is intended solely to enable exits that are needed during recovery processing. The second phase occurs after CICS initialization is complete and, at this point, you may use PLT programs to customize the environment.

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

Recovery for 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.

Define temporary storage queues as recoverable using temporary storage model resource definitions as shown in the following example define statements:

```
CEDA DEFINE DESCRIPTION(Recoverable TS queues for START requests) TSMODEL(RECOV1)
            GROUP(TSRECOV) PREFIX(DF) LOCATION(AUXILIARY) RECOVERY(YES)

CEDA DEFINE DESCRIPTION(Recoverable TS queues for BMS) TSMODEL(RECOV2)
            GROUP(TSRECOV) PREFIX(**) LOCATION(AUXILIARY) RECOVERY(YES)

CEDA DEFINE DESCRIPTION(Recoverable TS queues for BMS) TSMODEL(RECOV3)
            GROUP(TSRECOV) PREFIX($$) LOCATION(AUXILIARY) RECOVERY(YES)
```

CICS continues to support temporary storage tables (TST) and you could define recoverable TS queues in a TST, as shown in the following example:

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

The DATAID DF makes the temporary storage queues used on CICS start requests recoverable.

The DATAIDs ** and $$ make the 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 System Definition Guide*.

Forward recovery

If you want forward recovery of temporary storage you must provide application programs to record the changes made to temporary storage during the current CICS run.

1. At emergency restart time, your application program 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.

Recovery for Web services

If your Web services use the WMQ transport and persistent messages, you can use BTS to ensure that messages are recovered in the event of a CICS system failure.

Configuring CICS to support persistent messages

CICS provides support for sending persistent messages using the WMQ transport protocol to a Web service provider application that is deployed in a CICS region.

About this task

CICS uses Business Transaction Services (BTS) to ensure that persistent messages are recovered in the event of a CICS system failure. For this to work correctly, follows these steps:

Procedure

1. Use IDCAMS to define the local request queue and repository file to MVS. You must specify a suitable value for STRINGS for the file definition. The default value of 1 is unlikely to be sufficient, and you are recommended to use 10 instead.
2. Define the local request queue and repository file to CICS. Details of how to define the local request queue to CICS are described in "Defining local queues in a service provider." You must specify a suitable value for STRINGS in the file definition. The default value of 1 is unlikely to be sufficient, and it is recommended that you use 10 instead.
3. Define a PROCESSTYPE resource with the name DFHMQSOA, using the repository file name as the value for the FILE option.
4. Ensure that during the processing of a persistent message, a program issues an **EXEC CICS SYNCPOINT** command before the first implicit syncpoint is requested; for example, using an SPI command such as **EXEC CICS CREATE TDQUEUE** implicitly takes a syncpoint. Issuing an **EXEC CICS SYNCPOINT** command confirms that the persistent message has been processed successfully. If a program does not explicitly request a syncpoint before trying to implicitly take a syncpoint, CICS issues an ASP7 abend.

Results

What to do next

For one way request messages, if the Web service abends or backs out, sufficient information is retained to allow a transaction or program to retry the failing request, or to report the failure appropriately. You need to provide this recovery transaction or program. See "Persistent message processing" on page 138 for details.

Defining local queues in a service provider

To use the WebSphere® MQ transport in a service provider, you must define one or more local queues that store request messages until they are processed, and one trigger process that specifies the CICS transaction that will process the request messages.

Procedure

1. Define an initiation queue. Use the following command:

   ```
   DEFINE
   QLOCAL('initiation_queue')
   DESCR('description')
   ```

   where *initiation_queue* is the same as the value specified for the INITQNAME attribute of the MQINI resource definition for the CICS region. MQINI is an implicit resource that CICS installs when you install an MQCONN resource definition. Use the EXEC CICS or CEMT INQUIRE MQINI command to inquire on the initiation queue name.

2. For each local request queue, define a QLOCAL object. Use the following command:

```
DEFINE
QLOCAL('queuename')
DESCR('description')
PROCESS(processname)
INITQ('initiation_queue')
TRIGGER
TRIGTYPE(FIRST)
TRIGDATA('default_target_service')
BOTHRESH(nnn)
BOQNAME('requeuename')
```

where:

- *queuename* is the local queue name.
- *processname* is the name of the process instance that identifies the application started by the queue manager when a trigger event occurs. Specify the same name on each QLOCAL object.
- *initiation_queue* is the name of the initiation queue to be used; for example, the initiation queue specified in the MQINI definition for the CICS region.
- *default_target_service* is the default target service to be used if a service is not specified on the request. The target service is of the form '/string' and is used to match the path of a URIMAP definition; for example, '/SOAP/test/test1'. The first character must be '/' .
- *nnn* is the number of retries that are attempted.
- *requeuename* is the name of the queue to which failed messages are sent.

3. Define a PROCESS object that specifies the trigger process. Use the following command:

```
DEFINE
PROCESS(processname)
APPLTYPE(CICS)
APPLICID(CPIL)
```

where:

processname is the name of the process, and must be the same as the name that is used when defining the request queues.

Persistent message processing

When a Web service request is received in a WMQ persistent message, CICS creates a unique BTS process with the process type DFHMQSOA. Data relating to the inbound request is captured in BTS data-containers that are associated with the process.

The process is then scheduled to run asynchronously. If the Web service completes successfully and commits, CICS deletes the BTS process. This includes the case when a SOAP fault is generated and returned to the Web service requester.

Error processing

If an error occurs when creating the required BTS process, the Web service transaction abends, and the inbound Web service request is not processed. If BTS is

not usable, message DFHPI0117 is issued, and CICS continues without BTS, using the existing channel-based container mechanism.

If a CICS failure occurs before the Web service starts or completes processing, BTS recovery ensures that the process is rescheduled when CICS is restarted.

If the Web service abends and backs out, the BTS process is marked complete with an ABENDED status. For request messages that require a response, a SOAP fault is returned to the Web service requester. The BTS process is canceled, and CICS retains no information about the failed request. CICS issues message DFHBA0104 on transient data queue CSBA, and message DFHPI0117 on transient data queue CPIO.

For one way messages, there is no way to return information about the failure to the requester so the BTS process is retained in a COMPLETE ABENDED state. CICS issues message DFHBA0104 on transient data queue CSBA, and DFHPI0116 on transient data queue CPIO.

You can use the CBAM transaction to display any COMPLETE ABENDED processes, or you can supply a recovery transaction to check for COMPLETE ABENDED processes of the DFHMQSOA and take appropriate action.

For example, your recovery transaction could:

1. Reset the BTS process using the **RESET ACQPROCESS** command.
2. Issue the **RUN ASYNC** command to retry the failing Web service. It could keep a retry count in another data-container on the process, to avoid repeated failure.
3. Use information in the associated data-containers to report on the problem:

   The DFHMQORIGINALMSG data-container contains the message received from WMQ, which might contain RFH2 headers.

   The DFHMQMSG data-container contains the WMQ message with any RFH2 headers removed.

   The DFHMQDLQ data-container contains the name of the dead letter queue associated with the original message.

   The DFHMQCONT data-container contains the WMQ MQMD control block relating to the **MQ GET** for the original message.

Chapter 13. Programming for recovery

When you are designing your application programs, you can include recovery facilities that are provided by CICS; for example, you can use global user exits for backout recovery.

This section covers the following topics:

Designing applications for recovery

In this context, **application** refers to a set of one or more **transactions** designed to fulfill the particular needs of the user organization. A transaction refers to a set of actions within an application which the designer chooses to regard as an entity. It corresponds to a unit of execution, which is typically started in a CICS region by invoking the transaction by its identifier from a terminal attached to CICS.

As an application designer, you must decide how (if at all) to subdivide an application into transactions, and whether the transactions should consist of just one unit of work, or more than one.

Ideally, but not necessarily, a transaction would correspond to a unit of work. Dividing the business application into units of work that correspond to transactions simplifies the entire recovery process.

An example of a typical business application is an order-entry system. A typical order-entry application includes all the processes needed to handle one order from a customer, designed as a set of processing units, as follows:

1. Check the customer's name and address and allocate an order number.
2. Record the details of ordered items and update inventory files.
3. Print the invoice and shipping documents.

Depending on the agreed recovery requirements statement, you could design noting details of ordered items and updating files either as one large transaction or as several transactions, with one transaction for each item within the order.

Splitting the application into transactions

Specify how to divide the application into transactions.

About this task

Procedure

1. Name each transaction, and describe its function in terms that the terminal user can understand. Your application could include transactions to recover from failures, such as:

- *Progress transaction*, to check on progress through the application. Such a function could be used after a transaction failure or after an emergency restart, as well as at any time during normal operation. For example, it could be designed to find the correct restart point at which the terminal user should recommence the interrupted work. This would be particularly relevant in a pseudo-conversation.
- *Catch-up function*, for entering data that the user might have been forced to accumulate by other means during a system failure.

2. Specify the files and databases that can be accessed in each processing unit. 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.
3. Specify how to apply the updates for those files and databases that can be updated by an application processing unit. Factors to consider here include the consistency and the immediacy of updates.
   a. 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 might be necessary to ensure that a quantity subtracted from the inventory file is, at the same time, added to the to-be-shipped file.
   b. Specify when newly entered data must or can be applied to the files or databases:
      - The application processing unit updates the files and databases as soon as the data is accepted from the user.
      - The application processing unit accumulates updates for later action; for example, by a later processing unit within the same application or a batch application that runs overnight. If you choose the batch option, make sure that there is enough time for the batch work to complete the number of updates.

   Use the above information when deciding on the internal design of application processing units.
4. Specify what data needs to be passed from one application processing unit to another. For example, in an order-entry application, one processing unit might accumulate order items. Another separate processing unit might update the inventory file. Clearly, there is a need here for the data accumulated by the first processing unit to be passed to the second. Use this information when deciding what resources are required by each processing unit.

Example

What to do next

Relationships between processing units

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

For example, in an order-entry application, one processing unit may accumulate order items. Another, separate, processing unit may update the inventory file. Clearly, there is a need here for the data accumulated by the first processing unit to be passed to the second.

This information is needed when deciding what resources are required by each processing unit (see “Mechanisms for passing data between transactions” on page 145).

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. Therefore, you have 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 units of work

You must decide how to implement application processing units in terms of transactions, units of work, and programs.

About this task

You are recommended to plan your application processing units using the following advice:

Procedure

1. In programs that support a dialog with the user, consider implementing each unit of work to include only a single terminal read and a single terminal write. Using this approach can simplify the user restart procedures (see also "Processing dialogs with users" on page 144).

   Short units of work are preferable for several reasons:

   - Data resources are locked 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 re-enter when a transaction restarts after a failure.

     In applications for which little or no re-keying is feasible (discussed in question 9 under "Questions relating to recovery requirements" on page 101), short units of work are essential so that all entered data is **committed** as soon as possible.
2. Consider the recovery/restart implications when deciding whether to divide a transaction into many units of work.

   CICS functions such as dynamic transaction backout and transaction restart work most efficiently for transactions that have only one unit of work. But there can be situations in which multiple-unit of work transactions are necessary, for example if a set of file or database updates must be irrevocably

committed in **one** unit of work, but the transaction is to continue with one or more units of work for further processing.

3. Where file or database updates must be kept in step, make sure that your application does them in the same unit of work. This approach ensures that those updates will all be committed together or, in the event of the unit of work being interrupted, the updates will back out together to a consistent state.

Processing dialogs with users

An application may require several interactions (input and output) with the user.

CICS provides the following basic techniques for program design for use in such situations:

- Conversational processing
- Pseudo-conversational 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 locks 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 units of work).
- 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 several file or database updates made by several 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

In pseudoconversational processing, successive terminal interactions with the user are processed as separate tasks, usually consisting of one unit of work each.

This approach can result in a requirement to communicate between tasks or transactions (see "Mechanisms for passing data between transactions" on page 145) 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, pseudoconversational transactions are generally preferred to the conversational type.

When several terminal interactions with the user are related to each other, data for updates must accumulate on a recoverable resource 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

back out only the updates made during that individual step; the application is responsible for restarting at the appropriate point in the conversation, which might involve recreating a screen format.

However, other tasks might try to update the database between the time when update information is accepted and the time when it is applied to the database. Design your application to ensure that no other application can update the database at a time when it would corrupt the updating by your own application.

Mechanisms for passing data between transactions

In those applications where one transaction needs to access working data created by a previous transaction, you must decide which mechanism will pass the data between the transactions.

You have two options for passing data between transactions:

- Main storage areas
- CICS recoverable resources

Main storage areas

The advantages of main storage areas are realized only where recovery is not important, or when passing data between programs servicing the same task.

Main storage areas that you can use 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. Also, some of these storage areas can cause inter-transaction affinities, which are a hindrance to dynamic transaction routing. To avoid inter-transaction affinities, use either a COMMAREA or the TCTUA. For information about intertransaction affinities, see the *CICS Application Programming Guide*.

Design programs 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

Resources recoverable by backout that can be used for communication between transactions include:

- Temporary storage (auxiliary)
- Transient data queues
- User files and DL/I and DB2 databases

- Data tables (user-maintained)
- Coupling facility data tables

CICS can return all these resources to their status at the beginning of an in-flight unit of work if a task ends abnormally.

Temporary storage (auxiliary)

You can use a temporary storage item to communicate 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 to communicate between a sequence of transactions.

Transient data queues

Transient data (intrapartition) is similar to temporary storage (auxiliary) for communicating between transactions. The main difference is that you can read each record in a transient data queue only once, after which the record is no longer available.

Transient data must be specified as **logically** recoverable to achieve backout to the start of any in-flight unit of work.

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.

User-maintained data tables

User-maintained data tables (UMTs), which are recoverable after a unit of work failure can be a useful means of passing data between transactions. However, they are not forward recoverable, and not recoverable after a CICS restart.

Coupling facility data tables

Coupling facility data tables updated using the locking model, and which are recoverable after a unit of work failure, can be a useful means of passing data between transactions.

Unlike UMTs, coupling facility data tables are recoverable in the event of a CICS failure, CFDT server failure , or an MVS failure. However, they are not forward recoverable.

Designing to avoid transaction deadlocks

You must design your program to avoid transaction deadlocks. There are a number of techniques that you can use in your program to avoid this situation.

About this task

Consider using the following techniques:

Procedure

- 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.
- Enforce explicit installation enqueueing standards so that all applications do the following:
  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 you update several file or database records, 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.

  If you allow updates on a data set through the base and one or more alternate index (AIX) paths, or through multiple AIX paths, sequencing 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 several record updates, always use a single path or the base. Define such a procedure in your 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 unit of work, the new task will not be initiated, even though its start time may have passed. (See "START requests" on page 76 for more information about the PROTECT option.)

Consider also the possibility of a started task that fails. Unless you include abend processing in the program, only the master terminal will know about the failure. The abend processing should analyze the cause of failure as far as possible, and restart the task if appropriate. Ensure that either the user or the 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 (TD trigger level)

Specifying the TRANSID operand in the resource definition 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 "Recovery for temporary storage" on page 135.

The application program should then send a committed output message to the user to say that the task is complete, and that the output data is available in the form of terminal pages.

If an uncontrolled termination occurs while the user is viewing the pages of data, those pages are not lost (assuming that temporary storage for BMS is designated as recoverable). After emergency restart, the user can resume terminal paging by using the CSPG CICS-supplied transaction and terminal paging commands. (For more information about CSPG, see the *CICS Supplied Transactions*)

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.

About this task

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.
- If the transaction that presents the data fails, dynamic transaction backout will be called.

  If the terminal at which the transaction runs 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 unit of work corresponds to a printer page or form.

Managing transaction and system failures

To help you manage transaction failures and uncontrolled shutdowns of the system, a number of facilities are available to help you.

About this task

These facilities 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

The actions taken by CICS are described under Chapter 8, “Unit of work recovery and abend processing,” on page 73 and “Processing operating system abends and program checks” on page 94.

Transaction failures

When a transaction fails, you can invoke CICS facilities during and after the abend process.

These facilities include:

- 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)

You can use these facilities individually or together. During the internal design phase, specify which facilities to use and determine what additional (application or systems) programming might be involved.

The RESP option on a command returns a condition ID that can 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 might want to inform the master terminal operator, who can decide to terminate CICS, especially if one of the files is critical to the application.

Your installation might have standards relating to the use of RESP options or HANDLE CONDITION commands. Review these for each new application.

HANDLE ABEND commands

A **HANDLE ABEND** command can pass control to a routine within a transaction, or to 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.

Your installation might have standards relating to the use of HANDLE ABEND commands; review these for each new application.

EXEC CICS SYNCPOINT ROLLBACK command

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 unit of work only and not in the task as a whole.
- The SYNCPOINT command, with or without the ROLLBACK option, causes a new unit of work to start.
- If you have a transaction abend, and you do not want the transaction to continue processing, issue an **EXEC CICS ABEND** and allow dynamic transaction backout to back out the updates and ensure data integrity. Use rollback only if you want the application to regain control after nullifying the effects of a unit of work.

  For programming information about the SYNCPOINT command, see the *CICS Application Programming Reference*.

Dynamic transaction backout

Dynamic transaction backout (DTB) occurs automatically for all transactions that have recoverable resources. It is not an option you can specify on a transaction resource definition. The actions of DTB are described under "Transaction backout" on page 74.

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.

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.

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 93). These conditions can be changed, if absolutely necessary, by modifying the restart program DFHREST.

Use of the program error program (DFHPEP)

Decide whether or not to include your own functions, examples of which are given in "The CICS-supplied PEP" on page 163. (DFHPEP is invoked during abnormal task termination as described at "Abnormal termination of a task" on page 93.)

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 provide the following functions:

- User exits for transaction backout processing to handle:
  - Logically deleting records added to BDAM or VSAM-ESDS files (see the *CICS Customization Guide* for details of the XFCLDEL global user exit point)
  - Backing out file control log records (see the *CICS Customization Guide* for details of the XFCBOUT global user exit point)
  - File errors during transaction backout (see the *CICS Customization Guide* for details of the XFCBFAIL global user exit point)
  - User recovery records read from the system log during emergency restart (see the *CICS Customization Guide* for details of the XRCINPT global user exit point).
- 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

You can write program-level abend exit code to perform different actions, depending on the abend that occurs.

For example, you might want to perform any of the following actions, although you are recommended to keep abend exit code to a minimum:

- Record application-dependent information relating to that task in case it terminates abnormally.

If you want to initiate a dump, do so in the exit code at the same program level as the abend. If you initiate the dump at a program level higher than where the abend occurred, you may lose valuable diagnostic information.

- Attempt local recovery, and then continue running the program.
- Send a message to the terminal operator if, for example, you believe that the abend is due to bad input data.

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 instruction length code (ILC) and program interrupt code (PIC) 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

If an abend occurs during the invocation of a CICS service, issuing a further request for the same service might cause unpredictable results because the reinitialization of pointers and work areas and the freeing of storage areas in the exit routine might not have been completed. In addition, ASPx abends, which are task abends while in syncpoint processing, cannot be handled by an application program.

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.

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 *CICS Messages and Codes* for the transaction abend codes for abnormal terminations that CICS initiates, their meanings, and the recommended actions.

Programming information relating to the coding of program-level exit code (such as addressability and use of registers) is in the *CICS Application Programming Guide*. For background information, see the *CICS Application Programming Guide*.

Processing the IOERR condition

Any program that attempts to process an IOERR condition for a recoverable resource must not issue a RETURN or SYNCPOINT command, but must terminate by issuing an ABEND command. A RETURN or SYNCPOINT command causes the recovery manager to complete the unit of work and commit changes to recoverable resources.

START TRANSID commands

In a transaction that uses the **START TRANSID** command to start other transactions, you must maintain logical data integrity.

You can maintain data integrity by following these guidelines:

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. If you pass data to the started transaction (on one of the data options FROM, RTERMID, RTRANSID, or QUEUE), ensure you define the associated temporary storage queue as recoverable. The temporary storage queue is named on the REQID option of the START command when using any of the data options. To make the temporary storage queue recoverable, define the queue in a temporary storage table using the DATAID option of the DFHTST TYPE=RECOVERY macro (see "Recovery for temporary storage" on page 135).

   This ensures that data being passed to another task is deleted from the temporary storage queue if the START-issuing task fails and is 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 a recoverable DATAID to ensures that, if a system failure occurs after the START-issuing task has completed its syncpoint, the START command is preserved. CICS starts the transaction specified on a recoverable START command after an emergency restart, when the expiry time is reached and provided the terminal specified the TERMID option is available. A DATAID is relevant only if data is being passed to the started transaction.

Note: Consider using EXEC CICS RETURN TRANSID(...) with the IMMEDIATE option if the purpose is to start the next transaction in a sequence on the same terminal. This does not unlock the terminal, incurs less overhead, and, in a dynamic transaction routing (DTR) environment, the transaction is eligible for DTR.

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.

Locking (enqueuing on) resources in application programs

This topic describes locking (enqueuing) functions provided by CICS (and access methods) to protect data integrity.

About this task

There are two forms of locking:

1. The implicit locking functions performed by CICS (or the access method) whenever your transactions issue a request to change data. These are described under:
   - "Implicit locking for files"
   - "Implicit enqueuing on logically recoverable TD destinations" on page 157
   - "Implicit enqueuing on recoverable temporary storage queues" on page 157
   - "Implicit enqueuing on DL/I databases with DBCTL" on page 158.
2. The explicit enqueuing function that you request by means of an EXEC CICS command. This is described under "Explicit enqueuing (by the application programmer)" on page 158.

Note: Locking (implicit or explicit) 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 locking on performance, however, is minimized by implementing applications with short units of work, as discussed under "Dividing transactions into units of work" on page 143.

Implicit locking for files

This section first describes the implicit locking provided while nonrecoverable files are being updated. It then describes the extended locking actions when recoverable files are being updated.

Nonrecoverable files

For BDAM files that are nonrecoverable (that is, LOG=NO is specified in the FCT entry), CICS does not lock records that are being updated.

By default, you get BDAM exclusive control, which operates on a physical block, is system wide, but lasts only until the update is complete. If a transaction reads a record for update under BDAM exclusive control, and the transaction subsequently decides not to change the data, it must release the BDAM exclusive control. To do this, issue an EXEC CICS UNLOCK command, which causes CICS to issue a RELEX macro.

If you don't want BDAM exclusive control, specify SERVREQ=NOEXCTL on the file entry in the FCT.

For nonrecoverable VSAM files accessed in non-RLS mode, VSAM exclusive control locks the control interval during an update. For nonrecoverable VSAM files accessed in RLS mode, SMSVSAM locks the record during the update.

Figure 13 on page 155 illustrates the extent of locking for nonrecoverable files.

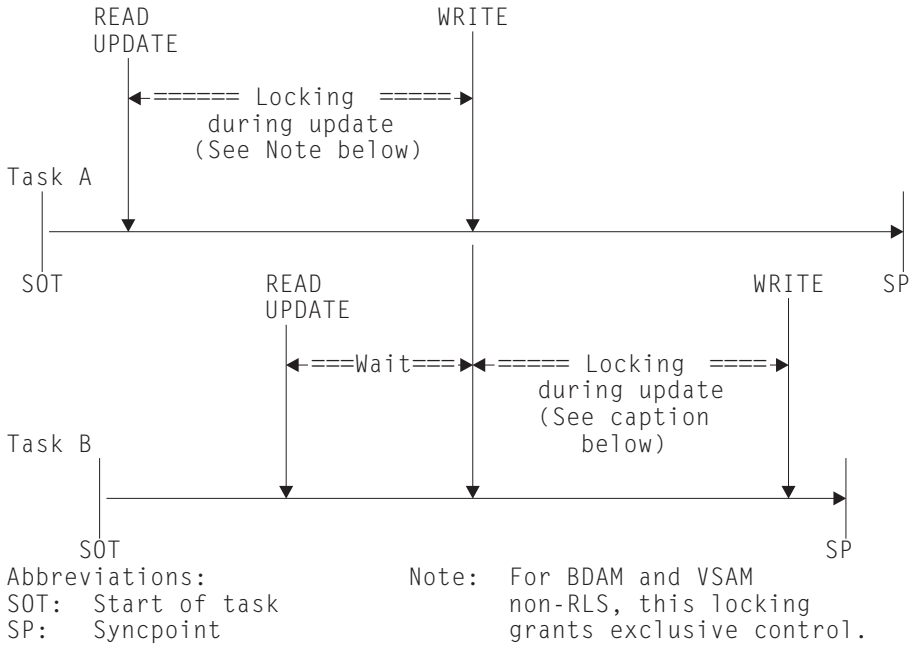


Figure 13. Locking during updates to nonrecoverable files. This figure illustrates two tasks updating the same record or control interval. Task A is given a lock on the record or control interval between the READ UPDATE and WRITE commands. During this period, task B waits.

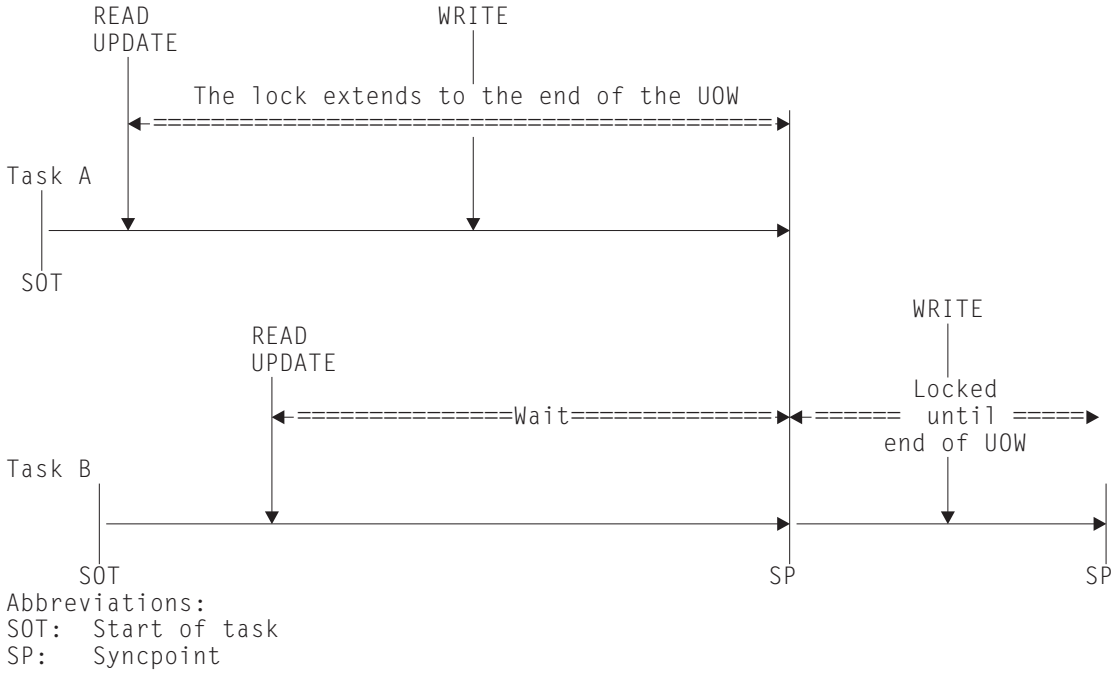


Figure 14. Locking (enqueuing on a resource) during updates to recoverable files. This figure illustrates two tasks updating the same record or control interval. Task A is given an exclusive lock on the record until the update is committed (at the end of the UOW). During this period, task B waits.

Recoverable files

For VSAM or BDAM files designated as recoverable, the duration of the locking action is extended. For VSAM files, the extended locking is on the updated record only, not the whole control interval.

The extended period of locking is needed to avoid an update committed by one task being backed out by another. (Consider what could happen if the nonextended locking action shown in Figure 13 on page 155 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 before update), CICS automatically locks the updated record until the change is committed (that is, until the end of the unit of work). 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 unit of work. Hence, it becomes impossible for Task B's update to be lost by a backout of Task A. For files opened in non-RLS mode, CICS provides this locking using the enqueue domain. For files opened in RLS mode, SMSVSAM provides the locking, and the locks are released at the completion of the unit of work at the request of CICS.

The file control commands that invoke automatic locking in this way are:

- READ (for UPDATE)
- WRITE
- DELETE

Note:

1. Enqueuing as described above can lead to **transaction deadlock** (see "Possibility of transaction deadlock" on page 159).
2. The scope of locks varies according to the access method, the type of access, and who obtains the lock:
   - BDAM exclusive control applies to the physical block
   - Non-RLS VSAM exclusive control applies to the control interval
   - CICS locks for BDAM (with NOEXCTL specified) apply to the record only
   - CICS locks for non-RLS VSAM apply to the record only
   - SMSVSAM locks for RLS apply to the record only
3. **VSAM exclusive control**. The CICS enqueuing action on recoverable files opened in non-RLS mode lasts until the end of the unit of work. When a transaction issues a READ UPDATE command, VSAM's exclusive control of the control interval containing the record is held only long enough for CICS to issue an ENQ on the record. CICS then notifies VSAM to release the exclusive control of the CI, and re-acquires this only when the task issues a REWRITE (or UNLOCK, DELETE, or SYNCPOINT) command. Releasing the VSAM exclusive CI lock until the task is ready to rewrite the record minimises the potential for transaction deadlock
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 unit of work, 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 indicated by message DFHFC4701, with a failure code of X'F0'. There is no locking on the AIX® key to prevent the second task taking the key before the end of the first task's unit of work. 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 unit of work.

5. To ensure that the data being read is up-to-date, the application program should:
   - For files accessed in non-RLS mode, issue a READ UPDATE command (rather than a simple READ), thus locking the data until the end of the unit of work
   - For files accessed in RLS mode, use the consistent read integrity option.

Implicit enqueuing on logically recoverable TD 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 unit of work). 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 unit of work).
- 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 unit of work). 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 unit of work).
  - If a task issues a DELETEQ TD request, the task is enqueued upon both the read and the write destinations. While the task is thus enqueued, no other task can read from, or write to, the queue.

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).")

Temporary storage control commands that invoke implicit enqueuing are:

- WRITEQ TS
- DELETEQ TS

Implicit enqueuing on DL/I databases with DBCTL

IMS™ program isolation scheduling ensures that, 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.

How long it is enqueued 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).

Explicit enqueuing (by the application programmer)

CICS provides enqueuing commands that can be useful in applications when you want to protect data and prevent transaction deadlocks.

CICS provides the following explicit enqueuing commands:

- **EXEC CICS ENQ RESOURCE**
- **EXEC CICS DEQ RESOURCE**

You can use these commands to perform the following functions:

- 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 unit of work ends.

Note: Enqueueing on more than one resource concurrently might create a deadlock between transactions.

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.**

As shown in Figure 15, 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, DL/I program isolation scheduling action, or VSAM RLS locking action protects resources until the next synchronization point is reached.)

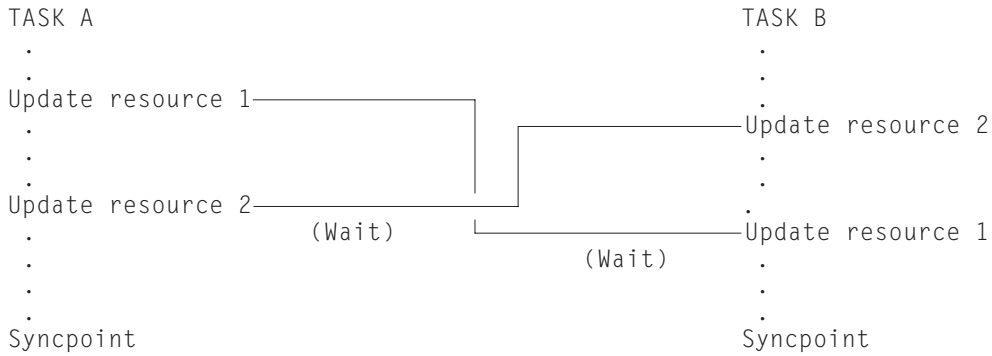


Figure 15. Transaction deadlock (generalized)

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

- If both deadlocked resources are non-RLS resources, CICS file control detects the deadlock and abends one of the transactions with an AFCF abend code.
- If both deadlocked resources are VSAM RLS resources, deadlock detection is performed by VSAM. If VSAM detects an RLS deadlock condition, it returns a deadlock exception condition to CICS, causing CICS file control to abend the transaction with an AFCW abend code. CICS also writes messages and trace entries that identify the members of the deadlock chain.

  Note: VSAM cannot detect a cross-resource deadlock (for example, a deadlock arising from use of RLS and DB2 resources) where another resource manager is involved. VSAM resolves a cross-resource deadlock when the timeout period expires, as defined by either the DTIMOUT or FTIMEOUT parameters, and the waiting request is timed out. In this situation, VSAM cannot determine whether the timeout is caused by a cross-resource deadlock, or by a timeout caused by another transaction acquiring an RLS lock and not releasing it.
- If the resources are both DL/I databases, 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 (with abend code ADCD).

- If both deadlocked resources are CICS resources (but not both VSAM resources), or one is CICS and the other DL/I, CICS abends the task whose DTIMOUT period elapses first. 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 canceled by a master terminal command.

The abended task may then be backed out by dynamic transaction backout, as described in "Transaction backout" on page 74. (Under certain conditions, the transaction can be restarted automatically, as described under "Abnormal termination of a task" on page 93. Alternatively, the terminal operator may restart the abended transaction.)

For more information, see "Designing to avoid transaction deadlocks" on page 146.

User exits for transaction backout

You can include your own logic in global user exit programs that run during dynamic transaction backout and backout at emergency restart.

There are exits in the file control recovery control program, and in the user log record recovery program (which is driven at emergency restart only). Transient data and temporary storage backouts do not have any exits.

Where you can add your own code

At emergency restart, you can add your own code in post-initialization programs that you nominate in the program list table.

About this task

You might want to include functions in global user exit programs that run during an emergency restart:

- Process file control log records containing the details of file updates that are backed out
- Deal with the backing-out of additions to data set types that do not support deletion (VSAM ESDS and BDAM), by flagging the records as logically deleted
- Handle file error conditions that arise during transaction backout
- Process user recovery records (in the XRCINPT exit of the user log record recovery program during emergency restart)
- Deal with the case of a non-RLS batch program having overridden RLS retained locks (this should not occur very often, if at all)

You could consider using the following exits:

1. XRCINIT—invoked at the beginning and end of the user log record recovery program
2. XRCINPT—invoked whenever a user log record is read from the system log
3. XFCBFAIL—file backout failure exit
4. XFCLDEL—file logical delete exit
5. XFCBOVER—file backout non-RLS override exit
6. XFCBOUT—file backout exit

You can use any of these exits to add your own processing if you do not want the default action, but do not set the UERCPURG return code for these exits because the exit tasks cannot be purged. To use these exits at emergency restart:

Procedure

- Enable them in PLT programs in the first part of PLT processing.
- Specify them on the system initialization parameter, **TBEXITS**. This takes the form TBEXITS=(*name1*,*name2*,*name3*,*name4*,*name5*,*name6*), where *name1, name2, name3, name4, name5*, and *name6* are the names of your global user exit programs for the XRCINIT, XRCINPT, XFCBFAIL, XFCLDEL, XFCBOVER, XFCBOUT exit points.

What to do next

For programming information on the generalized interface for exits, how to write exit programs, and the input parameters and return codes for each exit, see the *CICS Customization Guide*.

XRCINIT exit

XRCINIT is invoked at warm and emergency restart:

1. When the first user log record is read from the system log, and before it is passed to the XRCINPT exit
2. After the last user log record has been read from the system log and passed to XRCINPT

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

CICS passes information in the global user exit parameter list about user-written log records presented at the XRCINPT exit. The parameters includes a flag byte that indicates the disposition of the user log records. This can indicate the state of the unit of work in which the user log records were found, or that the record is an activity keypoint record. The possible values indicate:

- The record is an activity keypoint record
- The UOW was committed
- The UOW was backed out
- The UOW was in-flight
- The UOW was indoubt

XRCINPT exit

XRCINPT is invoked at warm and emergency restart, once for each user log record read from the system log.

The default action at this exit is to do nothing.

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 system log. Take care that this action does not put data integrity at risk.

XFCBFAIL global user exit

XFCBFAIL is invoked whenever an error occurs during backout of a unit of work.

An XFCBFAIL global user exit program can decide whether to bypass, or invoke, CICS backout failure control, The processing performed by CICS backout failure control is described under “Backout-failed recovery” on page 79.

XFCLDEL global user exit

XFCLDEL is invoked when backing out a unit of work that performed a write operation to a VSAM ESDS, or a BDAM data set.

XFCBOVER global user exit

XFCBOVER is invoked whenever CICS is about to decide not to backout an uncommitted update, because the record could have been updated by a non-RLS batch program.

This situation can occur after a batch program has opened a data set, even though it has retained locks, by overriding the RLS data set protection.

XFCBOUT global user exit

XFCBOUT is invoked when CICS is about to backout a file update.

Coding transaction backout exits

You have access to all CICS services, except terminal control services, during exit execution.

About this task

However, consider the following restrictions:

- The exit programs must be written in assembler code.
- They must be quasi-reentrant. They may use the exit programming interface (XPI) and issue EXEC CICS commands.
- If an exit program 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 opened in non-RLS mode 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 should be released by the exit as soon as its contents are no longer needed.
- If an exit is not used, the default actions are taken.
- We strongly recommend that emergency restart global user exits do not change any *recoverable* resource. If you do try to use temporary storage, transient data, or file control, these resource managers may also be in a state of recovery. Access to these services will, therefore, at best cause serialization of the recovery tasks and, at worst, cause a deadlock.

Chapter 14. Using a program error program (PEP)

The program error program (PEP) gains control after all program-level ABEND exit code has executed and after dynamic transaction backout has been performed.

About this task

There is only one program error program for the whole region.

Procedure

1. Decide whether you want to use the CICS-supplied program error program, DFHPEP or create your own. The CICS-provided DFHPEP program executes no functions, but you can include in it your own code to carry out an installation-level action following a transaction abend (see "The CICS-supplied PEP").
2. If you decide to create your own PEP, modify the CICS-supplied version. The default DFHPEP is in CICSTS41.CICS.SDFHLOAD, and is defined in the CICS DFHMISC group in the CSD.

   Read the guidance provided in the *CICS Customization Guide* for details on writing a PEP.
3. Update the **GRPLIST** system initialization parameter to include the DFHLIST group list. This includes a definition for the DFHMISC group.

The CICS-supplied PEP

The CICS-supplied PEP performs no processing. Its only effect, when CICS links to it, is to avoid the DFHAC2259 message that would otherwise be issued.

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

- 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

However, the following processing applies to the DFHPEP program:

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 is not given control if transaction manager detects that the abended transaction is to be restarted by DFHREST.
3. DFHPEP processing takes place after a transaction dump has been taken. DFHPEP cannot prevent a dump being taken.
4. DFHPEP is not given control if the transaction failure occurs during syncpoint processing.
5. DFHPEP is not given control when the conditions causing the task to be terminated are handled by the CICS abnormal condition program (ACP). The conditions handled by ACP are some kind of attach failure; for instance, when the transaction does not exist, or when a security violation is detected.
6. DFHPEP is not given control when a task has abended and CICS is short on storage.

7. The CICS transaction failure program, DFHTFP, links to DFHPEP before transaction backout is performed. This means resources used by the abending transaction may not have been released. DFHPEP needs to be aware of this, and might need logic to handle resources that are still locked.
8. Do not use the restart function for distributed transactions whose principal facilities are APPC links. In some error situations, CICS cannot resolve the APPC conversation states, and your transaction will abend with code AZCP.

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 might decide to incorporate these actions in 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 communications area (COMMAREA) containing information about the abend. You can add code to take appropriate actions to suit your requirements.

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 avoids the need to use CEMT commands 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.
- Disabling the facility, which would keep the system running longer than if you cause CICS to abend.

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.

When you have corrected the error, you can re-enable the relevant installed transaction definition to allow terminals to use it. You can also disable transaction identifiers when transactions are not to be accepted for application-dependent reasons, and can enable them again later. The *CICS Resource Definition Guide* 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.

Omitting the PEP

The CICS-supplied PEP is provided in the CICSTS41.CICS.SDFHLOAD library. The CICS abnormal condition program, however, will link to it only if a 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.

About this task

Chapter 15. Resolving retained locks on recoverable resources

This section describes how you can locate and resolve retained locks that are preventing access to resources, either by CICS transactions or by batch jobs.

About this task

Although the main emphasis in this section is on how you can switch from RLS to non-RLS access mode in preparation for batch operations involving the use of RLS-accessed data sets, it is also of interest for CICS online operation. It covers the following topics:

- "Quiescing RLS data sets"
- "Switching from RLS to non-RLS access mode" on page 172
- "Coupling facility data table retained locks" on page 182

Quiescing RLS data sets

Data sets opened by CICS in RLS mode can also be accessed by batch programs, provided the batch programs open them in RLS mode for read-only processing.

About this task

Batch programs cannot update recoverable data sets in RLS mode, because SMSVSAM prevents it. Thus, to update a recoverable data set from a batch program, first ensure that the data set is closed to all CICS regions. This allows the batch program to open the data set in non-RLS mode for update. (See "Switching from RLS to non-RLS access mode" on page 172.) To enable batch programs to access data sets opened in RLS mode, CICS supports the VSAM RLS data set **quiesce** and **unquiesce** functions.

The quiesce function enables you, with a single command, to close in an orderly manner throughout the sysplex any data sets that are open in RLS mode, and prevent the data sets being opened in RLS mode while they are in the quiesced state. This function is required in the data sharing environment because many CICS regions can have the same data set open for update at the same time. You can use the quiesce function to take a data set offline throughout the sysplex when:

- You want to switch between RLS and non-RLS VSAM access modes.
- You want to prevent data set access during forward recovery.

CICS supports the VSAM RLS quiesce interface by providing an RLS **quiesce exit** program that is driven for the quiesce and unquiesce functions. CICS regions initiating quiesce and unquiesce requests propagate these to other CICS regions, through the SMSVSAM control ACB and the CICS RLS quiesce exit program.

Other products such as DFSMSdss and CICSVR also communicate with CICS through their SMSVSAM control ACBs and the RLS quiesce exit.

The RLS quiesce and unquiesce functions

The RLS quiesce and unquiesce functions are initiated by a CICS command in one region, and propagated by the VSAM RLS quiesce interface to other CICS regions in the sysplex.

When these functions are complete, the ICF catalog shows the quiesce state of the target data set. The state of the quiesce flag in the ICF catalog controls whether a data set can be opened in RLS mode:

- When a quiesce operation has been completed throughout the sysplex, SMSVSAM sets the quiesced flag to prevent applications (either CICS or batch) from opening the data set in RLS mode.

  You can open a quiesced data set in non-RLS mode only.
- When an unquiesce operation has been completed, SMSVSAM clears the quiesced flag to permit the data set to be opened in RLS mode.

  You can open an unquiesced data set in either RLS or non-RLS mode, the mode for the entire sysplex being determined by the first ACB opened.

Normally, a data set cannot be opened for update in non-RLS access mode if SMSVSAM is holding retained locks against the data set. Thus, to enable a non-RLS batch program to update a recoverable data set that CICS regions have open in RLS mode, take the following steps:

1. Resolve shunted units of work that are holding retained locks
2. Quiesce the data set
3. Run the batch update job
4. Unquiesce the data set

CICS transactions that try to access a quiesced data in RLS-mode fail with a NOTOPEN condition. Non-RLS accesses are permitted subject to the data set's VSAM SHAREOPTIONS.

For more information, see "Switching from RLS to non-RLS access mode" on page 172.

Illustration of the quiesce flow across two CICS regions

Figure 16 on page 169 illustrates the operation of the CICS RLS quiesce operation. The notes that follow explain the numbered points.

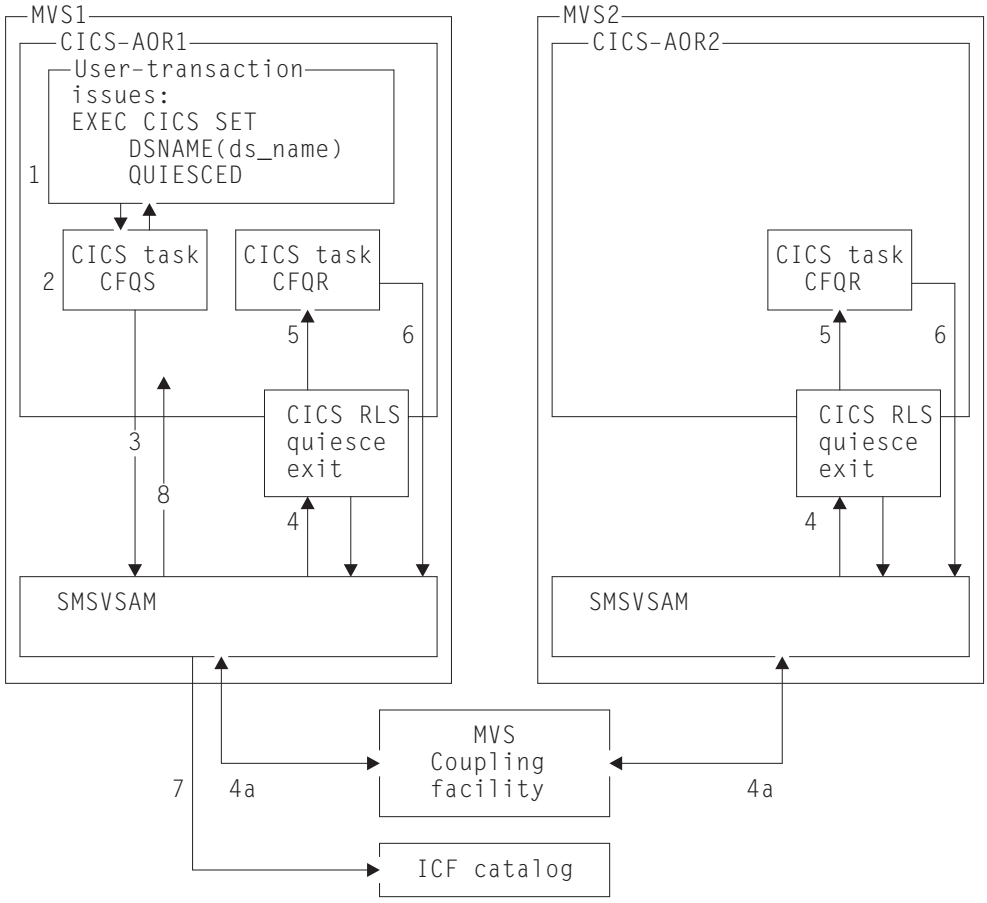


Figure 16. The CICS RLS quiesce operation with the CICS quiesce exit program

Note:

1. A suitably-authorized user application program (AOR1 in the diagram) issues an EXEC CICS SET DSNAME(...) QUIESCED command (or a terminal operator issues the equivalent CEMT command).

   If the command specifies the BUSY(NOWAIT) option, all phases of the quiesce operation are asynchronous, and the user application program continues processing.

   If the command specifies BUSY(WAIT), control is not returned to the user application program until SMSVSAM replies to CFQS that the quiesce function has completed (or failed).
2. CICS file control (AOR1 in the diagram) invokes CFQS to send the quiesce request to SMSVSAM.
3. The long-running CICS (AOR1) CFQS task passes the quiesce request to SMSVSAM across the control ACB interface using the IDAQUIES macro QUICLOSE function.
4. SMSVSAM drives the CICS RLS quiesce exit program of each CICS region that has an open RLS ACB for the specified data set (as shown in both AOR1 and AOR2 in the diagram).

   Note: This also applies to the CICS region in which a quiesce request is initiated if it has open RLS ACBs for the data set. Thus an initiator can also be a recipient of a quiesce request.

(4a) SMSVSAM uses the coupling facility to propagate the request to the other SMSVSAM servers in the sysplex.

5. The CICS RLS quiesce exit program schedules a CICS region task (CFQR) to perform asynchronously the required quiesce actions in that CICS region.
6. When CICS has closed all open RLS ACBs for the data set, CICS issues the "quiesce completed" notification (the IDAQUIES macro QUICMP function) direct to SMSVSAM through the control ACB interface.
7. When all CICS regions have replied with the IDAQUIES macro QUICMP function, the SMSVSAM server that handled the original request (from AOR1 in the diagram) sets the quiesced flag in the ICF catalog. This prevents any files being opened in RLS mode for the data set, but allows non-RLS open requests that are initiated, either implicitly or explicitly, by user transactions.
8. SMSVSAM returns to the CICS region that initiated the quiesce request. If the BUSY(WAIT) option was specified on the request, CFQS resumes the waiting application program.

The flow of an unquiesce request is virtually the same as that illustrated in Figure 16 on page 169, with the following differences in operation:

- As soon as SMSVSAM receives the unquiesce request from the initiating CICS region, it immediately sets the quiesce flag in the ICF catalog to the unquiesce state.
- SMSVSAM invokes the RLS quiesce exit program of each CICS region registered with a control ACB to notify it of the change in the quiesce state.
- The CICS RLS quiesce exit program returns a "quiesce completed" code to SMSVSAM immediately, and then passes the request to CFQR for processing.

Other quiesce interface functions

In addition to quiesce and unquiesce, there are some other data-set-related functions using the RLS quiesce interface that are not connected with quiescing or unquiescing activity on a data set.

These are functions initiated by DFSMSdss, CICSVR, or VSAM RLS, and may require some processing in those CICS regions that are registered with SMSVSAM. These functions do not cause any change in the ICF catalog quiesced flag.

The other functions provided on the RLS quiesce interface for data-set-related activities are as follows:

Non-BWO data set backup start

A quiesce interface function initiated by DFSMSdss in readiness for non-BWO backup processing for a data set that is open in RLS mode. This function prevents CICS file control issuing RLS update requests against a sphere so that the VSAM sphere can be backed up.

SMSVSAM invokes the CICS RLS quiesce exit program in each region that has an open RLS ACB for the data set.

If any in-flight units of work are using the data set when a QUICOPY notification is received, CICS allows them to complete (or be shunted). CICS then flags its data set name block (DSNB) for the data set to disable further updates. Any unit of work that attempts to update the data set thereafter is abended with an AFCK abend, and SMSVSAM prevents any new file opens for update.

Note: If enabled, the CICS XFCVSDS exit is invoked for this function, with the UEPVSACT parameter set to a value of UENBWST.

With the new RLS quiesce mechanism, you do not have to close a data set to take a non-BWO backup. However, because this causes new transactions to be abended, you may prefer to quiesce your data sets before taking a non-BWO backup.

Non-BWO data set backup end
A quiesce interface function initiated by DFSMSdss at the end of non-BWO backup processing (or to cancel a non-BWO backup request). This function enables CICS file control to permit RLS update requests against the data set now that backup processing is complete.

SMSVSAM invokes the CICS RLS quiesce exit program in each region that is registered with an SMSVSAM control ACB.

In addition to permitting updates, CICS writes tie-up log records to the forward recovery log and the log of logs, and retries any shunted units of work for the data set.

Note: If enabled, the CICS XFCVSDS exit is invoked for this function, with the UEPVSACT parameter set to a value of UENBWCMP.

BWO backup start
A quiesce interface function initiated by DFSMSdss in readiness for BWO backup processing for a data set that is open in RLS mode. This function enables CICS to ensure the data set is in a suitable state for a BWO backup to be taken.

SMSVSAM invokes the CICS RLS quiesce exit program in each region that has an open RLS ACB for the data set.

In response to this form of request, CICS writes tie-up records to the forward recovery log and log of logs, and waits for any in-flight units of work to complete (or be shunted). New units of work can then update the data set.

Note: If enabled, the CICS XFCVSDS exit is invoked for this function, with the UEPVSACT parameter set to a value of UEBWOST.

BWO backup end
A quiesce interface function initiated by DFSMSdss at the end of BWO backup processing (or to cancel a BWO backup request). It notifies CICS that a BWO backup of a data set is complete.

SMSVSAM invokes the CICS RLS quiesce exit program in each region that is registered with an SMSVSAM control ACB.

CICS does not perform any processing for this form of request.

Note: If enabled, the CICS XFCVSDS exit is invoked for this function, with the UEPVSACT parameter set to a value of UEBWOCMP.

Forward recovery complete
A quiesce interface function initiated by VSAM in response to a request from CICSVR. VSAM takes action associated with a sphere having completed forward recovery, which includes notifying CICS.

SMSVSAM invokes the CICS RLS quiesce exit program in each region that is registered with an SMSVSAM control ACB.

CICS retries any backout-failed shunted units of work for the data set.

Lost locks recovery complete
A quiesce interface function initiated by VSAM. VSAM takes action associated with a sphere having completed lost locks recovery on all CICS regions that were sharing the data set.

SMSVSAM invokes the CICS RLS quiesce exit program in each region that is registered with an SMSVSAM control ACB.

Until lost locks recovery is complete, CICS disallows any new requests to the data set (that is, only requests issued as part of the recovery processing are possible). When lost locks recovery is complete, CICS allows all requests to the data set.

Quiesce coupling facility cache available
A quiesce interface function initiated by VSAM. VSAM takes action associated with an MVS coupling facility cache structure being restored.

SMSVSAM invokes the CICS RLS quiesce exit program in each region that is registered with an SMSVSAM control ACB.

CICS retries any backout-failed shunted units of work for all data sets.

Switching from RLS to non-RLS access mode

The main reason why you might need to switch a data set from RLS access mode to non-RLS access mode is to run a batch update program that cannot be changed to run as a CICS transaction.

About this task

Non-RLS access mode is a generic term embracing NSR, LSR, and GSR access modes. It is likely that your existing batch programs open VSAM data sets in NSR access mode, although it is also possible for batch programs to use LSR or GSR.

As a general rule, you are recommended not to switch between RLS and non-RLS within CICS. After a data set is accessed in RLS mode by CICS, it should **always** be accessed in RLS mode by CICS. The following topic describes an exception to this general rule for read-only operations.

Exception for read-only operations

There is an exception to the general rule about not switching between RLS and non-RLS within CICS.

You can switch to non-RLS access on a data set that is normally opened in RLS mode provided access is *restricted to read-only operations*. You might want to do this, for example, to allow continued access for read-only transactions while the data set is being updated by a batch job. CICS and VSAM permit quiesced data sets to be opened in non-RLS mode, but you must ensure that CICS transactions do not update a data set that is being updated concurrently by a batch program.

The recommended procedure for providing CICS read access to a recoverable data set while it is being updated by a batch job is:

- Resolve retained locks.
- Quiesce the data sets.
- Redefine the files as non-RLS and read-only mode in all relevant CICS regions. You can do this using the CEMT, or EXEC CICS, SET FILE command.

Note: If your file definitions specify an LSR pool id that is built dynamically by CICS, consider using the RLSTOLSR system initialization parameter.

- Open the files non-RLS read-only mode in CICS.
- Concurrently, run batch non-RLS.
- When batch work is finished:
  - Close the read-only non-RLS mode files in CICS.
  - Re-define the files as RLS mode and with update operations. You can do this using the CEMT, or EXEC CICS, SET FILE command.
  - Unquiesce the data sets.
  - Open the files in CICS, if not using open on first reference.
  - Resume normal running.

You should also take data set copies for recovery purposes before and after a batch run as you would normally, regardless of whether you are switching from RLS to non-RLS access mode.

What can prevent a switch to non-RLS access mode?

You cannot open a data set in non-RLS access mode if there are any ACBs open against it in RLS access mode.

To switch from RLS to non-RLS access mode, first ensure that all files that are open against the data set are closed. You can use the quiesce function for this purpose. As discussed in "Quiescing RLS data sets" on page 167, the VSAM RLS quiesce mechanism causes each CICS region in the sysplex to close any RLS ACBs that are open against a specified data set. Once closed under the quiesce mechanism, data sets can only be opened in non-RLS mode. To re-enable quiesced data sets to be reopened in RLS mode, all open non-RLS ACBs must be closed and then the data sets must be unquiesced.

The quiesce mechanism cannot inform batch programs that have the data set open in RLS access mode about the quiesce request. If you have such programs, use the access method services SHCDS LIST subcommand to check whether any non-CICS jobs have ACBs open in RLS mode against the data set. For information about the SHCDS LIST subcommand, see *z/OS DFSMS: Access Method Services for ICF*.

Quiescing a data set sets the quiesce flag in the ICF catalog so that the data set can be opened in non-RLS mode only. This way of making data sets available for batch programs without having to shut down all the CICS regions is particularly suited to the Parallel Sysplex environment. Note that if you do shut CICS regions down, it is best to avoid immediate shutdowns when you have not specified a shutdown transaction, because this could cause large numbers of locks to be retained. A shutdown transaction can enable you to perform a quick but controlled shutdown.

Even if a data set has been quiesced, you still cannot open it for update in non-RLS access mode if SMSVSAM is holding retained locks against the data set. This is because the locks are needed to preserve data integrity: they protect changes that are waiting to be either committed or backed out. It is possible to open a data set for input (read-only) in non-RLS mode even when there are retained locks, because a reader cannot corrupt the data integrity that such locks are preserving. Note that, when you are opening a data set in a batch job, there cannot be any active locks because, when an ACB is closed, any active locks are converted automatically into retained locks.

The remainder of this topic on switching to non-RLS access mode describes the options that are available if you need to switch to non-RLS mode and are prevented from doing so by retained locks.

Resolving retained locks before opening data sets in non-RLS mode

VSAM sets an 'RLS-in-use' indicator in the ICF catalog cluster entry when a data set is successfully opened in RLS mode.

About this task

This indicator remains set, even when a data set opened in RLS mode is closed by all CICS regions, to show that the data set was last accessed in RLS mode. The indicator is cleared only when a data set is successfully opened in non-RLS mode. However, if SMSVSAM also holds retained locks against a data set, a non-RLS open fails and the RLS-in-use indicator remains set. In this event, try to resolve retained locks before attempting to use a data set in non-RLS mode for batch processing.

The procedure described here lists the steps you can take to switch from RLS to non-RLS mode in readiness for batch processing. The procedure is for a single data set, and should be repeated for each data set that is to be accessed from batch in non-RLS mode. CICS provides a suite of sample programs designed to help you to automate most of this procedure, which you can tailor to suit your own requirements (see "The batch-enabling sample programs" on page 178):

Procedure

1. Prevent further use of the applications that access the data set, so that no more locks can be created. One way to do this is to disable transactions that use the data set; but you may have your own technique for quiescing applications.
2. Check for any retained locks by using the INQUIRE DSNAME command on each CICS region that has been accessing the data set (see "INQUIRE DSNAME" on page 175). Generally there should not be any, and you can continue by quiescing the data set and running your batch program(s).
3. If there are retained locks, the RLS-in-use indicator prevents the data set being opened in non-RLS mode. In this case, use the CAUSE and REASON options on the INQUIRE UOWDSNFAIL command to find out what has caused them. You can then take steps to resolve the retained locks, preserving data integrity or not, using the SET DSNAME command. Do this before you quiesce the data set, because in some instances the locks cannot be resolved while the data set is quiesced.
4. Use the access method services SHCDS LIST subcommand to check whether there are any locks held by CICS regions on which you have not issued INQUIRE DSNAME; for example, because the system is not currently running. See *z/OS DFSMS: Access Method Services for ICF*.
5. If, for some reason, there are retained locks that you cannot resolve, you can choose to use the SHCDS PERMITNONRLSUPDATE subcommand that allows the batch program to override the retained locks and force the open for the data set.
6. When you are satisfied that there are no more retained locks, quiesce the data set and run your batch program(s).

Investigating which retained locks are held and why

CICS does not know exactly which locks are held by VSAM RLS.

About this task

However, it does know about the uncommitted changes that are protected by such locks, and why the changes have not yet been committed successfully. CICS uses this information to help you resolve any retained locks that are preventing you from switching to non-RLS access mode.

INQUIRE DSNAME

You can use the **INQUIRE DSNAME** command to find out whether the data set has any retained locks or if it is waiting for lost locks recovery by any CICS region.

INQUIRE DSNAME(...) RETLOCKS returns a value of RETAINED if the CICS region has any shunted units of work that have made uncommitted changes to the named data set, and NORETAINED if it does not. (RETLOCKS may return the value NORETAINED while changes are being made or units of work are being retried.) You can use this command to find out whether or not there are any retained locks that you need to resolve before running batch.

The INQUIRE DSNAME(...) LOSTLOCKS command returns:

- RECOVERLOCKS if the CICS region has lost locks recovery work to complete for the named data set. In this case, the RETLOCKS option returns RETAINED (except briefly while recovery work is being carried out, when RETLOCKS could be NORETAINED).
- REMLOSTLOCKS if the CICS region in which the command is issued does not have any lost locks recovery work to complete, but one or more other CICS regions in the sysplex do have. In this case the data set is still in a lost locks condition (but the RETLOCKS option in this CICS returns NORETAINED).
- NOLOSTLOCKS if the data set is not in a lost locks condition. This is the normal case. RETLOCKS could be either RETAINED or NORETAINED.

INQUIRE UOWDSNFAIL

CEMT INQUIRE UOWDSNFAIL DSN(*datasetname*) can return information about shunted units of work that have failed to commit changes to the named data set. The information includes the resource that caused the failure and, where there can be more than one reason, the reason for the failure. The EXEC CICS INQUIRE UOWDSNFAIL command provides a browse interface, allowing an application program to browse through all the shunted units of work for a data set or a group of data sets.

There are three main categories of failure:

- **Indoubt failure**, where a communication failure has occurred while a commit participant is in the indoubt state. The cause of an indoubt failure is returned on an INQUIRE UOWDSNFAIL command as CONNECTION. Indoubt failures are retried automatically when the connection is resynchronized.
- **Backout failure**, where a unit of work that is backing out changes fails during the backout of:
  - One or more data sets (which returns a CAUSE of DATASET)
  - A number of data sets bound to a failing cache (which returns a CAUSE of CACHE)
  - All the RLS data sets updated in the unit of work when the SMSVSAM server fails (which returns a CAUSE of RLSSERVER with REASON RLSGONE).

  Backout failures can be retried after the cause of the failure has been corrected. In some cases, the retry occurs automatically.

- **Commit failure**, where a unit of work has failed during the commit action. The commit action may be either to commit the changes made by a completed unit of work, or to commit the successful backout of a unit of work. This failure is caused by a failure of the SMSVSAM server, which is returned as RLSSERVER on the CAUSE option and COMMITFAIL or RRCOMMITFAIL on the REASON option of the INQUIRE UOWDSNFAIL command. (RLSSERVER is also returned as the WAITCAUSE on an INQUIRE UOW command.)

  Commit failures can be retried after the cause of the failure has been corrected. The retry usually occurs automatically.

Note that INQUIRE UOWDSNFAIL displays information about UOWs that are currently failed with respect to one or more data sets. The command does not display information for a unit of work that was in backout-failed or commit-failed state and is being retried, or for a unit of work that was indoubt and is being completed. The retry, or the completion of the indoubt unit of work, could fail, in which case a subsequent INQUIRE UOWDSNFAIL displays the relevant information.

SHCDS LIST subcommands

A CICS region that has been accessing a data set may not be running at the time you want to inquire on unit of work failures in readiness for a batch job.

Instead of bringing up such a CICS region just to issue INQUIRE UOWDSNFAIL commands, you can use the SHCDS LISTDS, LISTSUBSYS, or LISTSUBSYSDS commands. These commands enable you to find out which CICS regions (if any) hold retained locks, and to obtain other information about the status of your data sets with regard to each CICS region. You then know which regions you need to start in order to resolve the locks. These commands are described in *z/OS DFSMS: Access Method Services for ICF*.

The SHCDS LIST commands could also be useful in the unlikely event of a mismatch between the information that CICS holds about uncommitted changes, and the information that the VSAM RLS share control data set holds about retained locks.

Resolving retained locks and preserving data integrity

Unless prevented by time constraints, you are recommended to resolve retained locks by using the **INQUIRE UOWDSNFAIL** command and retrying shunted UOWs.

About this task

Procedure

1. Use the **INQUIRE UOWDSNFAIL** command to find out which failed component has caused the UOW to have retained locks for this data set.
2. If a unit of work that has been shunted with a CAUSE of DATASET and a REASON of DELEXITERROR, enable a suitable global user exit program at the XFCLDEL (logical delete) global user exit point and then issue a **SET DSNAME ACTION(RETRY)** command to retry the backout and drive the global user exit.
3. If a unit of work that has been shunted with a CAUSE of CONNECTION, use the SYSID and NETNAME returned by **INQUIRE UOWDSNFAIL** to find out what has caused the indoubt condition and correct the failure, after which automatic resynchronization should allow the unit of work to complete, either by being backed out or by committing the changes.

4. If a unit of work has been shunted with a different CAUSE and REASON, review the descriptions of these values in the **INQUIRE UOWDSNFAIL** command to determine what action to take to allow the shunted unit of work to complete.

Choosing data availability over data integrity

There may be times when you cannot resolve all the retained locks correctly, either because you cannot easily remedy the situations preventing the changes from being committed, or because of insufficient time.

About this task

In such a situation, you can still release the locks, using the SET DSNAME command, although in most cases you will lose some data integrity.

SET DSNAME UOWACTION(COMMIT, BACKOUT, or FORCE) operates on shunted indoubt units of work. It causes any indoubt units of work that have made changes to the data set (these will appear on an INQUIRE UOWDSNFAIL command with a CAUSE of CONNECTION) to complete in one of the following ways:

- By backing out or committing the changes according to the COMMIT or BACKOUT option on UOWACTION.
- By backing out or committing the changes according to the indoubt ACTION attribute specified on the transaction resource definition (when you specify the FORCE option on UOWACTION).

When the unit of work completes, the locks are released. If the action chosen is backout (either by explicitly specifying BACKOUT or as a result of the FORCE option), diagnostic information is written to the CSFL transient data queue. Therefore BACKOUT is normally the best option, because you can use the diagnostic information to correct the data if backout was the wrong decision. Diagnostic messages DFHFC3004 and DFHFC3010 are issued for each backed-out update. Choose COMMIT for the units of work only if you know this is the decision that would be communicated by the coordinator when resynchronization takes place following reconnection.

Note: The CEMT or EXEC CICS SET UOW command operates on a single unit of work and therefore gives better control of each unit of work. The SET DSN(...) UOWACTION operates on *all* indoubt units of work that have updated the specified data set. Because these may have also updated resources other than those in the specified data set, you may prefer to use the SET UOW command and consider the effect of your actions separately for each individual unit of work. However, if you are confident that the ACTION option specified on your transaction definitions indicates the best direction for each individual transaction, you can use SET DSNAME FORCE.

The SET DSNAME(*datasetname*) ACTION(RESETLOCKS) command operates on units of work that are shunted for reasons other than indoubt failure. CICS resets the locks for the named data set; that is, it:

- Releases any locks held against this data set by shunted units of work (other than indoubt failures)
- Discards the log records that represent the uncommitted changes made to the data set by those units of work
- Writes diagnostic information to the CSFL transient data queue for each discarded log record.

Diagnostic messages DFHFC3003 and DFHFC3010 are issued for each log record.

If a data set has both indoubt-failed and other (backout- or commit-) failed units of work, deal with the indoubt UOWs first, using SET DSNAME UOWACTION, because this might result in other failures which can then be cleared by the SET DSNAME RESETLOCKS command.

The batch-enabling sample programs

CICS provides a suite of sample programs to help you automate batch preparation procedures.

The three coordinating programs, which use CICS distributed program link (DPL) commands to run programs on a set of nominated CICS regions, are:

DFH0BAT1
This sample program disables, through the programs to which it links, a set of nominated transactions.

DFH0BAT2
This sample program identifies, through the programs to which it links, the retained lock information for nominated data sets:

- For each data set, it issues a SET DSNAME RETRY command to try to resolve any retained locks that are due to transient failures, or failures that have been corrected.
- After a timed delay to allow retries to run, it issues, for each data set, an INQUIRE DSNAME RETLOCKS command to determine whether there are any retained locks. If there are, it issues an INQUIRE UOWDSNFAIL command to obtain information about any remaining shunted units of work that have made uncommitted changes to the data set. It displays the information returned by the command, together with recommended procedures for resolving the locks.

DFH0BAT3
Through the programs to which it links, this sample program forces locks for nominated data sets:

- For each data set that has retained locks (determined by issuing INQ DSNAME RETLOCKS), it forces backout for the shunted indoubt units of work.
- After a timed delay to allow the forced backouts to run, it resets the locks for any commit-failed or backout-failed units of work.

The DFH0BAT3 sample program is also useful for resolving pending backouts after a failure to forward recover a data set. See "Procedure for failed RLS mode forward recovery operation" on page 198.

For more information about these batch-enabling sample programs see the *CICS Operations and Utilities Guide*, which describes how to install the programs and how to prepare the required input for them. See also the comments contained in the source code supplied in the CICSTS41.CICS.SDFHSAMP library.

CEMT command examples

The following is a series of CEMT command examples showing how you can use CEMT to resolve unit of work failures in preparation for batch processing.

1. A CEMT INQUIRE UOWDSNFAIL command for a specified data set shows in the following display that there are two failed units of work:

```
CEMT INQUIRE UOWDSNFAIL DSN('RLS.ACCOUNTS.ESDS.DBASE1')
STATUS:  RESULTS
 Dsn(RLS.ACCOUNTS.ESDS.DBASE1                    ) Dat Del
    Uow(AA6DB080C40CEE01)                         Rls
 Dsn(RLS.ACCOUNTS.ESDS.DBASE1                    ) Dat Ind
    Uow(AA6DB08AC66B4000)                         Rls
```

The display shows a REASON code of DELEXITERROR (Del) for one unit of work, and INDEXRECFULL (Ind) for the other.

2. A CEMT SET DSNAME(...) RETRY command, after fixing the delete exit error (caused because an XFCLDEL exit program was not enabled) invokes a retry of both units of work with the following result:

```
SET DSNAME('RLS.ACCOUNTS.ESDS.DBASE1') RETRY
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Dsn(RLS.ACCOUNTS.ESDS.DBASE1                    ) Vsa       NORMAL
    Fil(0001) Val Bas Rec Sta         Ava     Ret
```

3. Another CEMT INQ UOWDSNFAIL command shows that one of the two units of work failed again. It is the unit of work with the INDEXRECFULL error, as follows:

```
INQUIRE UOWDSNFAIL
STATUS:  RESULTS
 Dsn(RLS.ACCOUNTS.ESDS.DBASE1                     ) Dat Ind
    Uow(AA6DB08AC66B4000) Dat Ind                  Rls
```

This type of error can occur when adding a value to a non-unique alternate index while trying to back out a change to a record; after the original update was made, other units of work have added values to the alternate index, filling all the available space. Such an error should be extremely rare. The solution is to define a larger alternate index record size for the data set.

4. However, the data set with the remaining failed unit of work is needed urgently for a batch run, so you may decide not to complete the unit of work now, and to resolve the problem later. In the meantime, to allow the batch job to start, you issue `SET DSN('RLS.ACCOUNTS.ESDS.DBASE1') RESETLOCKS` to release the retained locks associated with the unit of work.

5. To check that there are no retained locks left, you can issue another INQUIRE DSNAME command. The expanded display shows:

```
Dsname(RLS.ACCOUNTS.ESDS.DBASE1)
Accessmethod(Vsam)
Action(              )
Filecount(0001)
Validity(Valid)
Object(Base)
Recovstatus(Recoverable)
Backuptype(Static)
Frlog()
Availability( Available )
Lostlocks()
Retlocks(Noretained)
Quiescestate()
Uowaction(              )
Basedsname(RLS.ACCOUNTS.ESDS.DBASE1)
Fwdrecovlsn(RLS.ACCOUNTS.FWDRECOV.LOG)
```

6. Diagnostic messages have been written to the CSFL transient data queue about the locks that were reset.

```
DFHFC3003 01/04/95 12:08:03 CICSHDA1 Record not backed out because locks for
          a backout-failed data set have been reset.
          Diagnostic information follows in message DFHFC3010. The record was
          updated by unit of work X'AA6DB08AC66B4000' for file ACCNT1  , base
          data set RLS.ACCOUNTS.ESDS.DBASE1

DFHFC3010 01/04/95 12:08:03 CICSHDA1 Diagnostic information for unit of work
```

```
X'AA6DB08AC66B4000' and file ACCNT1  .
Update was a write-add made by transaction WKLY at terminal T583
under task number 00027.  Key length 4, data length 7, base ESDS
RBA X'00000DDF', record key X'00000DDF'
```

A special case: lost locks

If a lost locks condition occurs, any affected data set remains in a lost locks state until all CICS regions have completed lost locks recovery for the data set.

Lost locks recovery is complete when all uncommitted changes, which were protected by the locks that were lost, have been committed. Therefore, after a lost locks condition occurs, you may need to follow the same procedures as those for preparing for batch jobs (described in "The batch-enabling sample programs" on page 178), to ensure that the data set is made available again as soon as possible. Preserving data integrity should be the priority, but you may decide to force indoubt units of work and to reset locks in order to make the data set available sooner.

Overriding retained locks

There may be situations in which it is difficult or inconvenient to use CICS commands to remove all the retained locks held against a data set.

About this task

For these situations, DFSMS access method services provides a SHCDS subcommand that allows you to run a non-RLS batch program despite the fact that there are retained locks.

The PERMITNONRLSUPDATE subcommand

The SHCDS PERMITNONRLSUPDATE subcommand allows you to run a non-RLS batch job where it is not possible to resolve all the retained locks that are held against the data set.

This subcommand overrides the DFSMS controls that prevent non-RLS opens-for-update when a data set still has retained locks. Use this option only as a last resort.

The effect of the PERMITNONRLSUPDATE command is canceled as soon as the data set is re-opened in RLS mode after the batch work is complete. Re-issue the command when you next want to override retained locks.

The DENYNONRLSUPDATE subcommand

The SHCDS DENYNONRLSUPDATE subcommand allows you to reset the permit status of a data set to prevent non-RLS updates.

This subcommand is provided for the situation where you issue the PERMITNONRLSUPDATE subcommand in error, and want to turn the permit status off again without running any non-RLS work:

- Specify DENYNONRLSUPDATE if you do *not* run a non-RLS batch job after specifying PERMITNONRLSUPDATE. This is because CICS, when it next opens the data set in RLS mode, takes action assuming that the data set has been opened for update in non-RLS mode if the PERMITNONRLSUPDATE state is set.

- Do not use DENYNONRLSUPDATE if you run non-RLS work after specifying PERMITNONRLSUPDATE. The permit status is automatically reset by the CICS regions that hold retained locks when they open the data set in RLS mode.

Post-batch processing

After a non-RLS program has been permitted to override retained locks, the uncommitted changes that were protected by those locks must not normally be allowed to back out.

This is because the non-RLS program may have changed the protected records. To ensure this, all CICS regions that held retained locks are notified by VSAM that a non-RLS update may have occurred because of the PERMITNONRLSUPDATE override. The CICS regions receive this notification when they open the affected data set in RLS mode.

CICS stores information about any uncommitted changes that are currently outstanding, and then informs VSAM RLS that the PERMITNONRLSUPDATE state can be cleared with respect to this CICS region. CICS uses the stored information to prevent automatic backouts during retries, because this could cause unpredictable results if backouts were applied to data modified by a batch program. Whenever a change made by one of these UOWs is about to be backed out, CICS detects its special status and invokes the XFCBOVER global user exit program. This allows you to determine what action to take over the uncommitted changes. The default action for these units of work is not to retry the backouts.

Diagnostic information is provided to help you to correct the data if necessary. Diagnostic messages DFHFC3001 and DFHFC3010 are issued for each affected update. CICS uses an internal form of the SET DSNAME RETRY command to drive any pending backout failures immediately. In the case of indoubt failures, CICS has to wait for the indoubt to be resolved following resynchronization before knowing whether the changes were to have been backed out.

Note that neither CICS nor VSAM knows whether the non-RLS program did change any of the locked records, only that it had the potential to do so. With your knowledge of the batch application, you may know that the records, or certain of the records, could not have been changed by batch and could therefore be safely backed out. For example, the batch application might only ever add records to the data set. CICS provides a global user exit point, XFCBOVER, which allows you to request that records are backed out, and which you can also use to perform other actions. If you choose to back out an update, CICS issues diagnostic message DFHFC3002 instead of DFHFC3001.

The SHCDS LISTDS subcommand shows whether the non-RLS batch update PERMITNONRLSUPDATE state is currently set. It also shows the status of a PERMIT first-time switch, which is a switch that is cleared as soon as a data set for which non-RLS update has been allowed is next opened for RLS access. The non-RLS permitted state is used to inform each contributing CICS region that a non-RLS program has potentially overridden its retained locks. The first-time switch is used to ensure that a subsequent non-RLS open for update requires a new PERMITNONRLSUPDATE to be issued even if some permitted states have not yet been cleared.

Coupling facility data table retained locks

Recoverable coupling facility data table records can be the subject of retained locks, like any other recoverable CICS resource that is updated in a unit of work that subsequently fails.

A recoverable CFDT supports indoubt and backout failures. If a unit of work fails when backing out an update to a CFDT, or if it fails indoubt during syncpoint processing, the locks are converted to retained locks and the unit of work is shunted. If a CFDT record is locked by a retained lock, an attempt to modify the record fails with the LOCKED condition. This can be returned on the following API file control requests issued against CFDT records:

- DELETE with RIDFLD
- READ with UPDATE
- READNEXT with UPDATE
- READPREV with UPDATE
- WRITE

To resolve retained locks against CFDT records, use the CEMT INQUIRE UOWLINK command to find the unresolved units of work associated with links to the coupling facility data table pool containing the table. When you have found the UOW links, take the appropriate action to resolve the link failure or force completion of the UOWs.

The owner of the failed UOW could be another CICS region in the sysplex, and the cause of the retained lock could be one of the following:

- The CICS region cannot resolve the UOW because its CFDT server has failed.
- The CICS region that owns the UOW has failed to connect to its CFDT server.
- The CICS region that owns the UOW cannot resynchronize with its CFDT server.
- An indoubt failure caused by the loss of the CICS region that is coordinating a distributed UOW.
- An indoubt failure caused by the loss of connectivity to the CICS region that is coordinating a distributed UOW.

Chapter 16. Moving recoverable data sets that have retained locks

There may be times when you need to re-define a VSAM data set by creating a new data set and moving the data from the old data set to the new data set.

About this task

For example, you might need to do this to make a data set larger. In this case, note that special action is needed in the case of data sets that are accessed in RLS mode by CICS regions.

Recoverable data sets may have retained locks associated with them. For data sets accessed in RLS mode, SMSVSAM associates locks with the physical location of the data set on disk. If you move the data set from one disk location to another, ensure that any retained locks are associated with the new data set location after it is moved. DFSMS access method services provide some SHCDS subcommands that enable you to preserve these locks when moving a data set.

Procedure for moving a data set with retained locks

This topic outlines some procedures to follow when moving a data set from one location on DASD to another.

About this task

There are two basic methods that you can use to move a VSAM data set:

1. Use the REPRO function of the access method services utility, then delete the original data set.
2. Use the EXPORT and IMPORT functions of access method services.

With either of these methods, you need to take extra steps to preserve any retained locks for data sets accessed in RLS mode.

Using the REPRO method

If the recoverable data set you want to move has not been used in RLS mode, you can use access method services commands to perform a number of actions on it.

About this task

These actions include:

1. Create a new data set
2. Copy (REPRO) the data from the old data set to the new
3. Delete the old data set
4. Rename the new data set back to the old

In the case of a non-RLS mode data set, retained locks are not a problem and no other special action is needed.

The following access method services examples assume that CICS.DATASET.A needs to be redefined and the data moved to a data set named CICS.DATASET.B, which is then renamed:

```
DEFINE CLUSTER (NAME(CICS.DATASET.B) ...
REPRO INDATASET(CICS.DATASET.A) OUTDATASET(CICS.DATASET.B)
DELETE CICS.DATASET.A
ALTER CICS.DATASET.B NEWNAME(CICS.DATASET.A)
```

If the recoverable data set has associated RLS locks, these steps are not sufficient because:

- The REPRO command copies the data from CICS.DATASET.A to CICS.DATASET.B but leaves the locks associated with the original data set CICS.DATASET.A.
- The DELETE command deletes both the original data set and its associated locks.
- After you issue the ALTER command, the data set has been moved but the new data set does not have any associated locks, which have been lost when you deleted the original data set.

To enable you to move data and keep the locks associated with the data, access method services provide the following SHCDS subcommands. These are subcommands of the VSAM sharing control SHCDS command and must always be preceded by the SHCDS keyword:

SHCDS FRSETRR
This command marks the data set as being under maintenance, by setting a flag in the data set's ICF catalog entry. (This flag is shown as 'Recovery Required' in a catalog listing (LISTCAT).)

SHCDS FRUNBIND
This command unbinds any retained locks held against the data set. The locks are no longer associated with any specific disk location or data set.

SHCDS FRBIND
This command re-binds to the new data set all retained locks that were unbound from the old data set. The locks are now associated with the named data set. The data set name used in FRBIND must match the name used in the earlier FRUNBIND.

SHCDS FRRESETRR
This frees the data set from being under maintenance, resetting the flag in the ICF catalog.

Based on the above example of using REPRO to move a VSAM data set, the complete solution for a data set accessed in RLS mode, and which preserves the RLS locks, is as follows:

Procedure

1. Quiesce the data set that is being moved, to prevent access by CICS regions while maintenance is in progress.
2. Create a new data set into which the data is to be copied. At this stage, it cannot have the same name as the old data set. For example:
   ```
   DEFINE CLUSTER (NAME(CICS.DATASET.B) ...
   ```
3. Issue the access method services (AMS) SHCDS FRSETRR subcommand to mark the old data set as being under maintenance. For example:
   ```
   SHCDS FRSETRR(CICS.DATASET.A)
   ```

This makes the data set unavailable while the move from old to new is in progress, and also allows the following unbind operation to succeed.

4. Issue the SHCDS FRUNBIND subcommand to unbind any retained locks against the old data set. For example:

   ```
   SHCDS FRUNBIND(CICS.DATASET.A)
   ```

   This enables SMSVSAM to preserve the locks ready for rebinding later to the new data set.

   You can include the SHCDS FRSETRR and FRUNBIND subcommands of steps 3 and 4 in the same IDCAMS execution, but they must be in the correct sequence. For example, the SYSIN input to IDCAMS would look like this:

   ```
   //SYSIN   DD *
    SHCDS FRSETRR(old_dsname)
    SHCDS FRUNBIND(old_dsname)
   /*
   ```

5. After the unbind, use REPRO to copy the data from the old data set to the new data set created in step 1. For example:

   ```
   REPRO INDATASET(CICS.DATASET.A) OUTDATASET(CICS.DATASET.B)
   ```

6. Use the SHCDS FRSETRR subcommand to mark the new data set as being under maintenance. This is necessary to allow the later (step 8) rebind operation to succeed. For example:

   ```
   SHCDS FRSETRR(CICS.DATASET.B)
   ```

7. Delete the old data set to enable you to rename the new data set to the name of the old data set. For example:

   ```
   DELETE CICS.DATASET.A
   ```

8. Use access method services to rename the new data set to the name of the old data set. For example:

   ```
   ALTER CICS.DATASET.B NEWNAME(CICS.DATASET.A)
   ```

   You must give the new data set the name of the old data set to enable the following bind operation to succeed.

9. Use the SHCDS FRBIND subcommand to rebind to the recovered data set all the retained locks that were unbound from the old data set. For example:

   ```
   SHCDS FRBIND(CICS.DATASET.A)
   ```

10. Use the SHCDS FRRESETRR subcommand after the rebind to reset the maintenance flag and enable the data set for use. For example:

    ```
    SHCDS FRRESETRR(CICS.DATASET.A)
    ```

    You can include the SHCDS FRBIND and FRRESETRR subcommands of steps 8 and 9 in one IDCAMS execution, but they must be in the correct sequence. For example, the SYSIN input to IDCAMS would look like this:

    ```
    //SYSIN   DD *
     SHCDS FRBIND(dataset_name)
     SHCDS FRRESETRR(dataset_name)
    /*
    ```

Using the EXPORT and IMPORT functions

Similar considerations to those for the REPRO method also apply to the use of IMPORT when recovering data from a copy created by EXPORT.

About this task

The following steps are required to restore a data set copy that has been created by an access method services EXPORT command:

- Create a new empty data set into which the copy is to be restored, and use IMPORT to copy the data from the exported version of the data set to the new empty data set.
- Use SHCDS FRSETRR to mark the original data set as being under maintenance.
- Use SHCDS FRUNBIND to unbind the locks from the original data set.
- Use SHCDS FRSETRR to mark the new data set as being under maintenance.
- Delete the original data set.
- Rename the new data set back to the old name.
- Use SHCDS FRBIND to associate the locks with the new data set.
- Use SHCDS FRRESETRR to take the data set out of maintenance.

An example follows of the access method services commands used to restore a portable copy of a data set created by EXPORT. The example assumes that EXPORT has been used to create a copy of data set CICS.DATASET.A and that an empty data set with the correct VSAM characteristics called CICS.DATASET.B has been created into which the data will be imported.

```
IMPORT INTOEMTPTY INDATASET(CICS.DATASET.A.COPY) OUTDATASET(CICS.DATASET.B)
SHCDS FRSETRR(CICS.DATASET.A)
SHCDS FRUNBIND(CICS.DATASET.A)
SHCDS FRSETRR(CICS.DATASET.B)
DELETE CICS.DATASET.A
ALTER CICS.DATASET.B NEWNAME(CICS.DATASET.A)
SHCDS FRBIND(CICS.DATASET.A)
SHCDS FRRESETRR(CICS.DATASET.A)
```

Rebuilding alternate indexes

If you delete and rebuild an alternate index (using BLDINDEX) it is equivalent to moving the alternate index data set.

About this task

It is possible that VSAM might be holding locks on a unique key alternate index when you want to delete and rebuild an alternate index. In this case, ensure that you preserve the locks over the delete and rebuild process by setting 'Recovery Required' and transferring the locks to the rebuilt index. The steps you need to preserve the locks across a rebuild are:

1. Issue SHCDS FRSETRR to indicate the data set is under maintenance.
2. Issue SHCDS FRUNBIND to get VSAM to save the locks, to be re-applied after the rebuild.
3. Delete and redefine the alternate index.
4. Rebuild the alternate index (using BLDINDEX).
5. Issue FRBIND to bind the retained locks to the newly built alternate index.
6. Issue SHCDS FRRESETRR to unset the under-maintenance flag.

Chapter 17. Forward recovery procedures

If a data set that is being used by CICS fails, perhaps because of physical damage to a disk, you can recover the data by performing forward recovery of the data set.

About this task

Your forward recovery procedures can be based either on your own, or an ISV-supplied utility program for processing the relevant CICS forward recovery log streams, or you can use CICS VSAM Recovery. See *CICS VSAM Recovery for z/OS* for details of forward recovery using CICS VR.

This section covers the following topics:

Forward recovery of data sets accessed in RLS mode

A recoverable data set that is updated in RLS mode can have retained locks held for individual records.

In the event of a data set failure, it is important to ensure that you preserve any retained locks as part of the data set recovery process. This is to enable the locks associated with the original data set to be attached to the new data set. If the data set failure is caused by anything other than a volume failure, retained locks can be "unbound" using the SHCDS FRUNBIND subcommand. The data set can then be recovered and the locks rebound to the recovered data using the SHCDS FRBIND subcommand.

Note: If you use CICSVR to recover a data set, the unbinding and subsequent binding of locks is handled by CICSVR. CICSVR also uses the SHCDS FRSETRR and FRRESETRR commands to prevent general access to the data set during the recovery process.

If a data set failure is caused by the loss of a volume, it is *not* possible to preserve retained locks using FRUNBIND and FRBIND because SMSVSAM no longer has access to the failed volume. When recovering from the loss of a volume, you can ensure data integrity only by deleting the entire IGWLOCK00 lock structure, which forces CICS to perform lost locks recovery. CICS uses information from its system log to perform lost locks recovery. (For more information about lost locks processing, see "Lost locks recovery" on page 89.) Recovering data from the loss of a volume requires a different procedure from the simple loss of a data set.

The procedures to recover data sets that could have retained locks are described in the following topics:

- For recovery of failed data sets where the volume remains operational, see "Recovery of data set with volume still available" on page 188
- For recovery of failed data sets where the volume itself has failed, see "Recovery of data set with loss of volume" on page 189.

Recovery of data set with volume still available

The procedure described here is necessary to preserve any retained locks that are held by SMSVSAM against the data in the old data set. Unless you follow all the steps of this procedure, the locks will not be valid for the new data set, with potential loss of data integrity.

The following steps outline the procedure to forward recover a data set accessed in RLS mode. Note that the procedure described here refers to two data sets—the failed data set, and the new one into which the backup is restored. When building your JCL to implement this process, be sure you reference the correct data set at each step.

1. **Quiesce the data set**
   To prevent further accesses to the failed data set, quiesce it using the CEMT, or EXEC CICS, SET DSNAME QUIESCED command.
2. **Create a new data set**
   Create a new data set into which the backup is to be restored. At this stage, it cannot have the same name as the failed production data set.
3. **Issue FRSETRR**
   Use this access method services SHCDS subcommand to mark the failed data set as being subject to a forward recovery operation. This makes the data set unavailable to tasks other than those performing recovery functions, and also allows the following unbind operation to succeed.
4. **Issue FRUNBIND**
   Use this access method services SHCDS subcommand to unbind any retained locks against the failed data set. This enables SMSVSAM to preserve the locks ready for re-binding later to the new data set used for the restore. This is necessary because there is information in the locks that relates to the old data set, and it must be updated to refer to the new data set. Unbinding and re-binding the locks takes care of this.

   Note: You can include the access method services SHCDS FRSETRR and FRUNBIND subcommands of steps 3 and 4 in the same IDCAMS execution, but they *must* be in the correct sequence. For example, the SYSIN input to IDCAMS would look like this:

   ```
   //SYSIN   DD *
    SHCDS FRSETRR(old_dsname)
    SHCDS FRUNBIND(old_dsname)
   /*
   ```
5. **Restore the backup**
   After the unbind, restore a full backup of the data set to the new data set created in step 2. You can use the recovery function (HRECOVER) of DFSMShsm™ to do this.
6. **Issue the FRSETRR subcommand**
   Use this access method services SHCDS subcommand to mark the new data set as being subject to a forward recovery operation. This is necessary to allow the later bind operation to succeed.
7. **Run the forward recovery utility**
   Run your forward recovery utility to apply the forward recovery log to the restored data set, to redo all the completed updates.
8. **Delete the old data set**
   Delete the old data set to enable you to rename the new data set to the name of the failed data set.

9. Alter the new data set name
Use access method services to rename the new data set to the name of the old data set.

```
ALTER CICS.DATASETB NEWNAME(CICS.DATASETA)
```

You must give the restored data set the name of the old data set to enable the following bind operation to succeed.

10. Issue the FRBIND subcommand
Use this access method services SHCDS subcommand to re-bind to the recovered data set all the retained locks that were unbound from the old data set.

11. Issue the FRRESETRR subcommand
Use this access method services SHCDS subcommand after the re-bind to re-enable access to the data set by applications other than the forward recovery utility.

Note: You can include the SHCDS FRBIND and FRRESETRR subcommands of steps 10 and 11 in one IDCAMS execution, but they must be in the correct sequence. For example, the SYSIN input to IDCAMS would look like this:

```
//SYSIN   DD *
 SHCDS FRBIND(dataset_name)
 SHCDS FRRESETRR(dataset_name)
/*
```

These steps are summarized in the following commands, where the data set names are labeled with A and B suffixes:

```
CEMT SET DSNAME(CICS.DATASETA) QUIESCED
DEFINE CLUSTER(NAME(CICS.DATASETB) ...
SHCDS FRSETRR(CICS.DATASETA)
SHCDS FRUNBIND(CICS.DATASETA)
HRECOVER (CICS.DATASETA.BACKUP) ... NEWNAME(CICS.DATASETB)
SHCDS FRSETRR(CICS.DATASETB)
EXEC PGM=fwdrecov_utility
DELETE CICS.DATASETA
ALTER CICS.DATASETB NEWNAME(CICS.DATASETA)
SHCDS FRBIND(CICS.DATASETA)
SHCDS FRRESETRR(CICS.DATASETA)
```

If you use CICSVR, the SHCDS functions are performed for you (see Forward recovery with CICS VSAM Recovery).

After successful forward recovery, CICS can carry out any pending backout processing against the restored data set. Backout processing is necessary because the forward recovery log contains after images of all changes to the data set, including those that are uncommitted, and were in the process of being backed out when the data set failed.

Recovery of data set with loss of volume

Moving a data set that has retained locks means the locks associated with the original data set have somehow to be attached to the new data set.

In the event of a lost volume, a volume restore implicitly moves data sets. Even if you are using CICSVR, which normally takes care of re-attaching locks to a recovered data set, the movement of data sets caused by loss of a volume cannot be managed entirely automatically.

There are several methods you can use to recover data sets after the loss of a volume. Whichever method you use (whether a volume restore, a logical data set recovery, or a combination of both), you need to ensure SMSVSAM puts data sets into a lost locks state to protect data integrity. This means that, after you have carried out the initial step of recovering the volume, your data recovery process must include the following command sequence:

1. ROUTE *ALL,VARY SMS,SMSVSAM,TERMINATESERVER
2. VARY SMS,SMSVSAM,FORCEDELETELOCKSTRUCTURE
3. ROUTE *ALL,VARY SMS,SMSVSAM,ACTIVE

The first command terminates all SMSVSAM servers in the sysplex and temporarily disables the SMSVSAM automatic restart facility. The second command (issued from any MVS) deletes the lock structure. The third command restarts all SMSVSAM servers, as a result of which SMSVSAM records, in the sharing control data set, that data sets are in lost locks state. The automatic restart facility is also reenabled.

Each CICS region detects that its SMSVSAM server is down as a result of the TERMINATESERVER command, and waits for the server event indicating the server has restarted before it can resume RLS-mode processing. This occurs at step 3 in the above procedure.

It is important to realize the potential impact of these commands. Deleting the lock structure puts *all* RLS-mode data sets that have retained locks, or are open at the time the servers are terminated, into the lost locks condition. A data set which is in lost locks condition is not available for general access until all outstanding recovery on the data set is complete. This is because records are no longer protected by the lost locks, and new updates can only be permitted when all shunted UOWs with outstanding recovery work for the data set have completed.

When CICS detects that its server has restarted, it performs dynamic RLS restart, during which it is notified that it must perform lost locks recovery. During this recovery process, CICS does not allow new RLS-mode work to start for a given data set until all backouts for that data set are complete. Error responses are returned on open requests issued by any CICS region that was not sharing the data set at the time SMSVSAM servers were terminated, and on RLS access requests issued by any new UOWs in CICS regions that were sharing the data set. Also, indoubt UOWs must be resolved before the data set can be taken out of lost locks state.

For RLS-mode data sets that are not on the lost volume, the CICS regions can begin lost locks recovery processing as soon as they receive notification from their SMSVSAM servers. For the data sets on these other volumes, recovery processing completes quickly and the data sets are removed from lost locks state.

For those data sets that are unavailable (for example, they are awaiting forward recovery because they are on the lost volume), CICS runs the backouts only when forward recovery is completed. In the case of CICSVR-managed forward recovery, completion is signalled automatically, and recovered data sets are removed from lost locks state when the associated backouts are run.

Volume recovery procedure using CFVOL QUIESCE

If a volume is lost and you logically recover the data sets using CICS VR, you do not need to use the CFVOL QUIESCE command (step 1 in the procedure described below).

This is because CICS cannot run the lost locks recovery process until the data sets are available, and the data sets are made available only after the CICS VR recovery jobs are finished.

If you physically restore the volume, however, the data sets that need to be forward recovered are immediately available for backout. In this case you must use CFVOL QUIESCE before the volume restore to prevent access to the restored volume until that protection can be transferred to CICS (by using the CICS SET DSNAME(...) QUIESCED command). When all the data sets that need to be forward recovered have been successfully quiesced, you can enable the volume again (CFVOL ENABLE). The volume is then usable for other SMSVSAM data sets.

Use the command `D SMS,CFVOL(volser)` to display the CFVOL state of the indicated volume.

CICS must not perform backouts until forward recovery is completed. The following outline procedure, which includes the three VARY SMS commands described above, prevents CICS opening for backout a data set on a restored volume until it is safe to do so. In this procedure *volser* is the volume serial of the lost volume:

1. VARY SMS,CFVOL(*volser*),QUIESCE

   Perform this step before volume restore. Quiescing the volume ensures that the volume remains unavailable, even after the restore, so that attempts to open data sets on the volume in RLS mode will fail with RC=8, ACBERFLG=198(X'C6'). Quiescing the volume also ensures CICS can't perform backouts for data sets after the volume is restored until it is re-enabled.
2. ROUTE *ALL,VARY SMS,SMSVSAM,TERMINATESERVER
3. VARY SMS,SMSVSAM,FORCEDELETELOCKSTRUCTURE
4. ROUTE *ALL,VARY SMS,SMSVSAM,ACTIVE

   Note at this point, as soon as they receive the “SMSVSAM available” event notification (ENF), CICS regions are able to run backouts for the data sets that are available. RLS-mode data sets on the lost volume, however, remain unavailable until a later ENABLE command.
5. At this point the procedure assumes the volume has been restored. This step transfers the responsibility of inhibiting backouts for those data sets to be forward recovered from SMSVSAM to CICS. Quiescing the data sets that need to be forward recovered is a first step to allowing the restored volume to be used for recovery work for other data sets:
   a. SET DSNAME(...) QUIESCED

      Use this command for all of the data sets on the lost volume that are to be eventually forward recovered. Issue the command before performing any of the forward recoveries.

      Note: A later SET DSNAME(...) UNQUIESCED command is not needed if you are using CICSVR.
   b. VARY SMS,CFVOL(*volser*),ENABLE

      Issue this command when CICS regions have successfully completed the data set QUIESCE function. You can verify that data sets are successfully quiesced by inquiring on the quiesced state of each data set using the CEMT INQUIRE DSNAME(...) command. If a data set is still quiescing, CICS displays the words `BEING QUIESCED`.

This clears the SMSVSAM CFVOL-QUIESCED state and allows SMSVSAM RLS access to the volume. CICS ensures that access is not allowed to the data sets that will eventually be forward recovered, but the volume is available for other data sets.

6. Run data set forward recovery jobs.

The following two examples show forward recovery after the loss of a volume, based on the procedure outline above:

Example of recovery using data set backup:

For this illustration, involving two data sets, we simulated the loss of a volume by varying the volume offline. The two data sets (RLSADSW.VF04D.DATAENDB and RLSADSW.VF04D.TELLCTRL) were being updated in RLS mode by many CICS AORs at the time the volume was taken offline. The CICS file names used for these data sets were F04DENDB and F04DCTRL.

The failed data sets were recovered onto another volume without first recovering the failed volume. For this purpose, you have to know what data sets are on the volume at the time of the failure. In "Example of recovery using volume backup" on page 196, we describe the recovery process by performing a volume restore before the forward recovery of data sets. Here are the steps followed in this example:

1. We simulated the volume failure using the MVS command:

   ```
   ROUTE *ALL,VARY 4186,OFFLINE,FORCE
   ```

   The loss of the volume caused I/O errors and transaction abends, producing messages on the MVS system log such as these:

   ```
   DFHFC0157 ADSWA04B 030
   TT1P 3326 CICSUSER An I/O error has occurred on base data set
   RLSADSW.VF04D.TELLCTRL accessed via file F04DCTRL component code
   X'00'.
   DFHFC0158 ADSWA04B 031
   96329,13154096,0005EDC00000,D,9S4186,A04B    ,CICS
   ,4186,DA,F04DCTRL,86- OP,UNKNOWN COND.  ,000000A5000403,VSAM

   DFHFC0157 ADSWA03C 301
   DE1M 0584 CICSUSER An I/O error has occurred on base data set
   RLSADSW.VF04D.DATAENDB accessed via file F04DENDB component code
   X'00'.
   DFHFC0158 ADSWA03C 031
   ...
   ```

   As a result of the transaction abends, CICS attempted to back out in-flight UOWs. The backouts failed because CICS couldn't access the data sets on the lost volume. The associated backout failures were reported by CICS, as follows:

   ```
   +DFHFC4701 ADSWA03A 336
    11/24/96 13:15:48 ADSWA03A Backout failed for transaction DE1H, VSAM
    file F04DENDB, unit of work X'ADD18C07DCB70A05', task 46752, base
    RLSADSW.VF04D.DATAENDB, path RLSADSW.VF04D.DATAENDB, failure code
    X'24'.

   +DFHFC0152 ADSWA03A 339
    11/24/96 13:15:49 ADSWA03A ???? DE1H An attempt to retain locks for
    data set within unit of work X'ADD18C07DCB70A05' failed.  VSAM return
    code X'00000008' reason code X'000000A9'.
   +DFHME0116 ADSWA03A 340
    (Module: DFHMEME) CICS symptom string for message DFHFC0152 is
   ```

```
  PIDS/565501800 LVLS/510 MS/DFHFC0152 RIDS/DFHFCCA PTFS/UN92873
  REGS/GR15 VALU/00000008 PCSS/IDARETLK PRCS/000000A9
 +DFHFC0312 ADSWA03A Message DFHFC0152 data set RLSADSW.VF04D.DATAENDB
```

We used the CEMT command INQUIRE UOWDSNFAIL IOERROR to display the UOWS that were shunted as a result of the I/O errors. For example, on the CICS region ADSWA01D the command showed the following shunted UOWs:

```
INQUIRE UOWDSNFAIL IOERROR
STATUS:  RESULTS
 Dsn(RLSADSW.VF04D.TELLCTRL                    ) Dat Ioe
    Uow(ADD18C2DA4D5FC03)                       Rls
 Dsn(RLSADSW.VF04D.DATAENDB                    ) Dat Ioe
    Uow(ADD18C2E693C7401)                       Rls
```

2. The next step was to stop the I/O errors by closing the RLS-mode files that were open against failed data sets. In our example, file F04DENDB was open against data set RLSADSW.FV04D.DATAENDB, and file F04DCTRL was open against data set RLSADSW.FV04D.TELLCTRL.

   The normal way of closing RLS-mode files across a sysplex is to quiesce the data set using the CEMT command SET DSNAME QUIESCED in one CICS region. However, the quiesce operation requires access to the data set, and fails if the data set cannot be accessed. The alternative is to issue the SET FILE(F04DENDB) CLOSED and SET FILE(F04DCTRL) CLOSED commands, which we did using CICSPlex® SM to send the command to all the relevant regions. (Without CICSPlex SM, issue the CEMT SET FILE CLOSED command to each CICS region individually, either from the MVS console or from a CICS terminal).

3. To enable CICSVR to recover the failed data sets, we first deleted the catalog entries for the two affected data sets using the IDCAMS DELETE command:

   ```
   DELETE RLSADSW.VF04D.TELLCTRL NOSCRATCH
   DELETE RLSADSW.VF04D.DATAENDB NOSCRATCH
   ```

4. The impact of the recovery process is greater if there are inflight tasks updating RLS mode files. For this reason, it is recommended at this point that you quiesce the data sets that are being accessed in RLS mode on other volumes before terminating the SMSVSAM servers. To determine which data sets are being accessed in RLS-mode by a CICS region, use the SHCDS LISTSUBSYSDS subcommand. For example, the following command lists those data sets that are being accessed in RLS-mode by CICS region ADSWA01D:

   ```
   SHCDS LISTSUBSYSDS('ADSWA01D')
   ```

   For the purpose of this example, we did not quiesce data sets; hence there is no sample output to show.

   Note: You can issue SHCDS subcommands as a TSO command or from a batch job.

5. We terminated the SMSVSAM servers using the MVS command:

   ```
   ROUTE *ALL,VARY SMS,SMSVSAM,TERMINATESERVER
   ```

   We received message IGW572 on each MVS image confirming that the servers are terminating:

   ```
   IGW572I REQUEST TO TERMINATE SMSVSAM
           ADDRESS SPACE IS ACCEPTED:
           SMSVSAM SERVER TERMINATION SCHEDULED.
   ```

   In our example, terminating the servers caused abends of all in-flight tasks that were updating RLS-mode data sets. This, in turn, caused backout failures and shunted UOWs, which were reported by CICS messages. For example, the

effect in CICS region ADSWA03C was shown by the following response to an INQUIRE UOWDSNFAIL command for data set RLSADSW.VF01D.BANKACCT:

```
INQUIRE UOWDSNFAIL DSN(RLSADSW.VF01D.BANKACCT)
STATUS:  RESULTS
 Dsn(RLSADSW.VF01D.BANKACCT                    ) Dat Ope
    Uow(ADD19B8166268E02)                       Rls
 Dsn(RLSADSW.VF01D.BANKACCT                    ) Rls Com
    Uow(ADD19B9D93DE1200)                       Rls
```

After the SMSVSAM servers terminated, all RLS-mode files were automatically closed by CICS and further RLS access prevented.

6. When we were sure that all servers were down, we deleted the IGWLOCK00 lock structure with the MVS command:

   ```
   VARY SMS,SMSVSAM,FORCEDELETELOCKSTRUCTURE
   ```

   followed by the response "FORCEDELETELOCKSTRUCTURESMSVSAMYES" to allow the lock structure deletion to continue.

   Successful deletion of the lock structure was indicated by the following message:

   ```
   IGW527I SMSVSAM FORCE DELETE LOCK STRUCTURE PROCESSING IS NOW COMPLETE
   ```

7. It was safe at this point to restart the SMSVSAM servers with the MVS command:

   ```
   ROUTE *ALL,VARY SMS,SMSVSAM,ACTIVE
   ```

   Initialization of the SMSVSAM servers resulted in the creation of a new lock structure, shown by the following message:

   ```
   IGW453I SMSVSAM ADDRESS SPACE HAS SUCCESSFULLY
           CONNECTED TO DFSMS LOCK STRUCTURE IGWLOCK00
           STRUCTURE VERSION: ADD1A77F0420E001 SIZE: 35072K bytes
           MAXIMUM USERS: 32 REQUESTED:32
           LOCK TABLE ENTRIES: 2097152 REQUESTED: 2097152
           RECORD TABLE ENTRIES: 129892 USED: 0
   ```

   The SMSVSAM server reported that there were no longer any retained locks but that instead there were data sets in the "lost locks" condition:

   ```
   IGW414I SMSVSAM SERVER ADDRESS SPACE IS NOW ACTIVE.
   IGW321I No retained locks
   IGW321I 45 spheres in Lost Locks
   ```

   CICS was informed during dynamic RLS restart about the data sets for which it must perform lost locks recovery. In our example, CICS issued messages such as the following to tell us that lost locks recovery was needed on one or more data sets:

   ```
   DFHFC0555 ADSWA04A One or more data sets are in lost locks status.
             CICS will perform lost locks recovery.
   ```

8. (If we had quiesced data sets before terminating the servers (see the comments between steps 3 and 4) this is the point at which we would unquiesce those data sets before continuing with the recovery.

   If there were many data sets in lost locks it would take some time for lost locks recovery to complete. Error responses are returned on open requests issued by any CICS region that was not sharing the data set at the time SMSVSAM servers were terminated, and on RLS access requests issued by any new UOWs in CICS regions that were sharing the data set. Also, it may be necessary to open explicitly files that suffer open failures during lost locks recovery.

   Each data set in a lost locks state is protected from new updates until all CICS regions have completed lost locks recovery for the data set. This means that all shunted UOWs must be resolved before the data set is available for new

work. Assuming that all CICS regions are active, and there are no indoubt UOWs, lost locks processing, for all data sets except the ones on the failed volume, should complete quickly.

9. In this example, CEMT INQUIRE UOWDSNFAIL on CICS region ADSWA01D showed UOW failures only for the RLSADSW.VF04D.TELLCTRL and RLSADSW.VF04D.DATAENDB data sets:

```
INQUIRE UOWDSNFAIL
STATUS:  RESULTS
 Dsn(RLSADSW.VF04D.TELLCTRL                    ) Dat Ope
    Uow(ADD18C2DA4D5FC03)                       Rls
 Dsn(RLSADSW.VF04D.DATAENDB                    ) Dat Ope
    Uow(ADD18C2E693C7401)                       Rls
```

   The command INQUIRE DSN(RLSADSW.VF04D.DATAENDB) on the same region showed that the lost locks status for the data set was **Recoverlocks**. This meant that the data set had suffered lost locks and that CICS region ADSWA01D had recovery work to complete:

```
INQUIRE DSN(RLSADSW.VF04D.DATAENDB)
RESULT - OVERTYPE TO MODIFY
  Dsname(RLSADSW.VF04D.DATAENDB)
  Accessmethod(Vsam)
  Action(              )
  Filecount(0001)
  Validity(Valid)
  Object(Base)
  Recovstatus(Fwdrecovable)
  Backuptype()
  Frlog(00)
  Availability( Available )
  Lostlocks(Recoverlocks)
  Retlocks(Retained)
  Quiescestate()
  Uowaction(              )
  Basedsname(RLSADSW.VF04D.DATAENDB)
  Fwdrecovlsn(ADSW.CICSVR.F04DENDB)
```

10. At this point, all data sets were available for new work except the two data sets on the failed volume. It was now possible to recover these using CICSVR. For information on using CICSVR see Forward recovery with CICS VSAM Recovery and the *CICSVSAM Recovery User's Guide and Reference*.
11. All CICS regions are automatically notified when CICSVR processing for a data set is complete. CICSVR preserves the lost locks state for the recovered data set and CICS disallows all new update requests until all CICS regions have completed lost locks recovery. When all CICS regions have informed SMSVSAM that they have completed their lost locks recovery, the data set lost locks state changes to **Nolostlocks**.
12. At this point recovery was complete and the recovered data sets were re-enabled for general access by issuing (through the CICSPlex SM "LOCFILES" view) the CEMT commands:

```
SET FILE(F04DENDB) ENABLED
SET FILE(F04DCTRL) ENABLED
```

    These commands are issued to each CICS AOR that requires access.
13. All data sets were now available for general access. We confirmed this using the SHCDS subcommand LISTSUBSYS(ALL), which showed that no CICS region had lost locks recovery outstanding.

If you follow the above example, but find that a CICS region still has a data set in lost locks, you can investigate the UOW failures on that particular CICS region using the CEMT commands INQUIRE UOWDSNFAIL and INQUIRE UOW. For indoubt UOWs that have updated a data set that is in a lost locks condition, CICS

waits for indoubt resolution before allowing general access to the data set. In such a situation you can still release the locks immediately, using the SET DSNAME command, although in most cases you will lose data integrity. See "Lost locks recovery" on page 89 for more information about resolving indoubt UOWs following lost locks processing.

Example of recovery using volume backup:

In this example, we simulated the recovery from the loss of a volume by performing a volume restore before the forward recovery process. Backout-failed UOWs were the result of the I/O errors that occurred when the volume failed.

Note: It is important to ensure that CICS cannot retry the shunted UOWs when the volume is restored, until after the forward recovery work is complete. This is done by quiescing the volume before it is restored, as described under "Volume recovery procedure using CFVOL QUIESCE" on page 190 (step 1).

Many of the steps in this second example are the same as those described under the "Example of recovery using data set backup" on page 192, and are listed here in summary form only.

1. We simulated the volume failure using the MVS command.

   ```
   ROUTE *ALL,VARY 4186,OFFLINE,FORCE
   ```

2. We stopped the I/O errors by closing the files that were open against failed data sets. In our example, file F04DENDB was open against data set RLSADSW.FV04D.DATAENDB and file F04DCTRL was open against data set RLSADSW.FV04D.TELLCTRL.
3. Because the failed data sets were restored from the same volume, there was no need to delete the catalog entries for these data sets.
4. Before restoring the failed volume, we quiesced the volume to ensure that CICS could not access the restored data sets. by issuing the command:

   ```
   VARY SMS,CFVOL(9S4186),QUIESCE
   ```

   In this example, for volume serial **9S4186**, the command produced the message:

   ```
   IGW462I DFSMS CF CACHE REQUEST TO QUIESCE VOLUME 9S4186 IS ACCEPTED
   ```

   We confirmed that the volume was quiesced by issuing the MVS command:

   ```
   DISPLAY SMS,CFVOL(9S4186)
   ```

   which confirmed that the volume was quiesced with the message:

   ```
   IGW531I DFSMS CF VOLUME STATUS
   VOLUME = 9S4186
   DFSMS VOLUME CF STATUS = CF_QUIESCED
   VOLUME 9S4186 IS NOT BOUND TO ANY DFSMS CF CACHE STRUCTURE
   ```

5. We simulated the volume restore for this example by using the MVS VARY command to bring the volume back online:

   ```
   ROUTE *ALL,VARY 4186,ONLINE
   ```

   Because the volume was quiesced, attempts to open files on this volume failed, with messages such as the following:

   ```
   DFHFC0500 ADSWA02A RLS OPEN of file F04DENDB failed. VSAM has
             returned code X'0008' in R15 and reason X'00C6'.
   ```

6. The impact of the recovery process is greater if there are inflight tasks updating RLS mode files. To minimize the impact, you are recommended at this point to quiesce all data sets that are being accessed in RLS mode.
7. We terminated the SMSVSAM servers with the MVS command:

```
ROUTE *ALL,VARY SMS,SMSVSAM,TERMINATESERVER
```

8. When all SMSVSAM servers were down, we deleted the IGWLOCK00 lock structure with the MVS command:

   ```
   VARY SMS,SMSVSAM,FORCEDELETELOCKSTRUCTURE
   ```

9. We restarted the SMSVSAM servers with the MVS command:

   ```
   ROUTE *ALL,VARY SMS,SMSVSAM,ACTIVE
   ```

   CICS was informed during dynamic RLS restart about the data sets for which it must perform lost locks recovery. CICS issued messages such as the following to inform you that lost locks recovery was being performed on one or more data sets:

   ```
   +DFHFC0555 ADSWA04A One or more data sets are in lost locks status.
              CICS will perform lost locks recovery.
   ```

10. If we had quiesced data sets before terminating the servers, this is the point at which we would unquiesce those data sets before proceeding.

    If there were many data sets in lost locks it would take some time for lost locks recovery to complete. It may be necessary to explicitly open files which suffer open failures during lost locks recovery.

11. At this point it was possible that there were data sets on the restored volume which did not require forward recovery. In order to make these data sets available, we needed to re-allow access to the volume. Before doing this, however, we first had to quiesce the data sets that still required forward recovery, thus transferring the responsibility of preventing backouts from SMSVSAM to CICS. In our example, we quiesced our two data sets using the CEMT commands:

    ```
    SET DSN(RLSADSW.VF04D.DATAENDB) QUIESCED
    SET DSN(RLSADSW.VF04D.TELLCTRL) QUIESCED
    ```

12. When we were sure that all data sets requiring forward recovery were quiesced, we used the following MVS command to allow access to the restored volume:

    ```
    VARY SMS,CFVOL(9S4186),ENABLE
    ```

    The above command produced the following message:

    ```
    IGW463I DFSMS CF CACHE REQUEST TO ENABLE
                  VOLUME 9S4186 IS COMPLETED.
                  DFSMS CF VOLUME STATUS = "CF_ENABLED"
    ```

13. At this point, all data sets were available for new work except the two quiesced data sets on the restored volume. We recovered these using CICSVR.

    All CICS regions were automatically notified when CICSVR processing for each data set was complete, and each data set was automatically unquiesced by CICSVR to allow the backout shunted UOWs to be retried.

    After all backout shunted UOWs were successfully retried, the recovery was complete and we re-enabled the recovered data sets for general access on each CICS region using the CEMT commands:

    ```
    SET FILE(F04DENDB) ENABLED
    SET FILE(F04DCTRL) ENABLED
    ```

14. Finally, we used the SHCDS command LISTSUBSYS(ALL) to confirm that no CICS region had lost locks recovery outstanding, indicating that recovery was complete.

Catalog recovery

If a user catalog is lost, follow the procedures documented in *DFSMS/MVS Managing Catalogs*. Before making the user catalog available, run the SHCDS CFREPAIR command to reconstruct critical RLS information in the catalog. Note that before running SHCDS CFREPAIR, the restored user catalog must be import

connected to the master catalog on all systems (see the "Recovering Shared Catalogs" topic in *DFSMS/MVS Managing Catalogs*).

Forward recovery of data sets accessed in non-RLS mode

For data sets accessed in non-RLS mode, use the following forward recovery procedure:

1. **Close all files**
   Close all the files that are open against the failed data set, by issuing CEMT, or EXEC CICS, SET FILE(...) CLOSED commands in the CICS region that owns the files.
2. **Create a new data set**
   Create a new data set into which the backup is to be restored. At this stage, it cannot have the same name as the failed production data set.
3. **Restore the backup**
   Restore a full backup of the data set to the new data set created in step 2. You can use the recovery function (HRECOVER) of DFSMShsm to do this.
4. **Run the forward recovery utility**
   Run your forward recovery utility to apply the forward recovery logs to the restored data set, to redo all the completed updates.
5. **Delete the old data set**
   Delete the old data set to enable you to rename the new data set to the name of the failed data set.
6. **Alter the new data set name**
   Use access method services to rename the new data set to the name of the old data set.

   ```
   ALTER CICS.DATASETB NEWNAME(CICS.DATASETA)
   ```

   Renaming is necessary to enable CICS to perform any outstanding backout processing.

These steps are summarized in the following commands, where the data set names are labeled with A and B suffixes:

```
CEMT SET FILE(...) CLOSED   (for each file opened against
                             the failed data set)
DEFINE CLUSTER(NAME(CICS.DATASETB) ...
HRECOVER (CICS.DATASETA.BACKUP) ... NEWNAME(CICS.DATASETB)
EXEC PGM=fwdrecov_utility
DELETE CICS.DATASETA
ALTER CICS.DATASETB NEWNAME(CICS.DATASETA)
```

Procedure for failed RLS mode forward recovery operation

There are some forward recovery failures that can be resolved.

For example, when:

- FRSETRR fails because the data set is already allocated to another job
- The restore fails because the backup has gone to tape and operator intervention is required
- The forward recovery doesn't work because the log data sets have been migrated

In these cases, you can resolve the cause of the failure and try the whole process again.

This topic describes what to do when the failure in forward recovery cannot be resolved. In this case, where you are unsuccessful in applying all the forward recovery log data to a restored backup, you are forced to abandon the forward recovery, and revert to your most recent full backup. For this situation, the access method services SHCDS command provides the FRDELETEUNBOUNDLOCKS subcommand, which allows you to delete the retained locks that were associated with the data set, instead of re-binding them to the recovered data set as in the case of a successful forward recovery.

The most likely cause of a forward recovery failure is the loss or corruption of one or more forward recovery logs. In this event, you probably have no alternative other than to restore the most recent backup and reapply lost updates to the data set manually. In this case, it is important that you force CICS to discard any pending (shunted) units of work for the data set that has failed forward recovery *before* you restore the most recent backup. This is because, during recovery processing, CICS assumes that it is operating on a data set that has been correctly forward recovered.

CICS performs most of its recovery processing automatically, either when the region is restarted, or when files are opened, or when a data set is unquiesced. There isn't any way that you can be sure of preventing CICS from attempting this recovery processing. How you force recovery processing before restoring the backup depends on whether or not the affected CICS regions are still running:

- For a CICS region that is still running, issue the appropriate CICS commands to initiate the retry of pending units of work.
- For a CICS region that is shut down, restart it to cause CICS to retry automatically any pending units of work.

Note: Ensure you issue any CICS commands, or restart a CICS region, *before* you restore the most recent backup, otherwise CICS will perform recovery processing against the restored data set, which you don't want.

In the event of a failed forward recovery of a data set, use the following procedure:

1. Tidy up any outstanding CICS recovery work, as follows:
   a. Make sure that any CICS regions that are not running, and which could have updated the data set, are restarted to enable emergency restart processing to drive outstanding backouts.
   b. Using information returned from the INQUIRE UOWDSNFAIL command issued in each CICS region that uses the data set, compile a list of all shunted UOWs that hold locks on the data set.
   c. If there are shunted indoubt units of work, try to resolve the in-doubts before proceeding to the next step. This is because the indoubt units of work may have updated resources other than the failed data set, and you don’t want to corrupt these other resources.

      If the resolution of an indoubt unit of work results in backout, this will fail for the data set that is being restored, because it is still in a recovery-required state. The (later) step to reset locks for backout-failed UOWs allows you to tidy up any such backout failures that are generated by the resolution of in-doubts.
   d. In all CICS regions that could have updated the failed data set:

1) Force shunted indoubt units of work using SET DSNAME(...) UOWACTION(COMMIT | BACKOUT | FORCE).

   Before issuing the next command, wait until the SET DSNAME(...) UOWACTION has completed against all shunted indoubt units of work.

   If the UOWACTION command for an indoubt unit of work results in backout, this will fail for the data set that is being restored, because it is still in a recovery-required state. The (next) step to reset locks for backout-failed UOWs allows you to tidy up any such backout failures that are generated by the resolution of in-doubts.

2) Reset locks for backout-failed units of work using SET DSNAME(...) RESETLOCKS.

   Do not issue this command until the previous UOWACTION command has completed. RESETLOCKS operates only on backout-failed units of work, and does not affect units of work that are in the process of being backed out. If you issue RESETLOCKS too soon, and shunted indoubt units of work fail during backout, the data set will be left with recovery work pending.

The DFH0BAT3 sample program provides an example of how you can do this.

There should not now be any shunted units of work on any CICS region with locks on the data set.

2. When you are sure that all CICS regions have completed any recovery work for the data set, delete the unbound locks using SHCDS FRDELETEUNBOUNDLOCKS.

   Note: It is very important to enter this command (and the following SHCDS FRRESETRR) at this stage in the procedure. If you do not, and the failed data set was in a lost locks condition, the data set will remain in a lost locks condition unless you cold start all CICS regions which have accessed it. If you mistakenly issued this command before step 1 of the procedure, then the problem concerning lost locks will still arise even if you issue the command again at this stage.

3. Allow access to the data set again using SHCDS FRRESETRR (even if you use CICSVR, you have to allow access manually if you have abandoned the forward recovery).
4. Finally, restore the backup copy from which you intend to work.

If the restored data set is eligible for backup-while-open (BWO) processing, you may need to reset the BWO attributes of the data set in the ICF catalog. This is because the failed forward recovery may have left the data set in a 'recovery-in-progress' state. You can do this using the CEMT, or EXEC CICS, SET DSNAME RECOVERED command.

If you do not follow this sequence of operations, the restored backup could be corrupted by CICS backout operations.

All the parts of step 1 of the above procedure may also be appropriate in similar situations where you do not want CICS to perform pending backouts. An example of this might be before you convert an RLS SMS-managed data set to non-SMS when it has retained locks, because the locks will be lost.

Procedure for failed non-RLS mode forward recovery operation

If you are not successful in applying all the forward recovery log data to a restored backup, you are forced to abandon the forward recovery, and revert to your most recent full backup.

However, during its recovery processing, CICS assumes that it is operating on a data set that has been correctly forward recovered, as in the case of recovery of a data set accessed in RLS mode (see "Procedure for failed RLS mode forward recovery operation" on page 198).

If you are not able to complete forward recovery on a data set, ensure that all CICS regions tidy up any pending recovery processing on the data set (as described below) before you restore the backup copy from which you intend to work. You can do this in the following way:

1. Make sure that any CICS regions that are not running, and which could have updated the data set, are restarted to enable emergency restart processing to drive outstanding backouts.
2. If there are shunted indoubt units of work, try to resolve the in-doubts before proceeding to the next step. This is because the indoubt units of work may have updated resources other than the failed data set, and you don't want to corrupt these other resources.
3. In all CICS regions that could have updated the failed data set:
   - Force shunted indoubt units of work using set DSNAME(...) UOWACTION(COMMIT | BACKOUT | FORCE).

     Before issuing the next command, wait until the SET DSNAME(...) UOWACTION has completed against all shunted indoubt units of work.
   - Reset locks for backout-failed units of work using SET DSNAME(...) RESETLOCKS.

     Do not issue this command until the previous UOWACTION command has completed. RESETLOCKS operates only on backout-failed units of work, and does not affect units of work that are in the process of being backed out. If you issue RESETLOCKS too soon, and shunted indoubt units of work fail during backout, the data set will be left with recovery work pending.
4. Finally, when you are sure that all CICS regions have completed any recovery work for the data set, restore the backup.

If you do not follow this sequence of operations, the restored backup could be corrupted by CICS backout operations.

Chapter 18. Backup-while-open (BWO)

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.

Many CICS applications depend on their data sets being open for update over a long period of time. Normally, you cannot take a backup of the data set 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 you must keep all forward recovery logs that have been produced since the data set was opened. A heavily used data set that has been open for update for several days or weeks might need much forward recovery.

Using BWO, 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.

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 that you can request:

- **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

The BWO function allows backups to be taken by DFSMSdss when applications are running in continuous operation while the data set is open for update with full data integrity of copied data.

Continuous operation is typically 24 hours-a-day for five to seven days a week. 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 continuously available 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.

BWO requirements

To use the backup-while-open (BWO) support provided by CICS, you can use the Data Facility Storage Management Subsystem/MVS (DFSMS/MVS) or a program product that provides equivalent function.

You must have your environment configured with the following modules and components:

- Use Release 2, or later, for data sets used in non-RLS access mode and Release 3 for data sets used in RLS access mode.
- You must install the DFSMSdfp IGWAMCS2 callable services module in the link pack area (LPA).
- You must install the IGWABWO module, supplied in SYS1.CSSLIB, in the LPA, or include SYS1.CSSLIB in the link list. Do not include the library in the STEPLIB or JOBLIB library concatenations.
- You must have the DFSMSdfp, DFSMSdss, and DFSMShsm components of DFSMS installed on the processors that perform backup and recovery.

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 DFSMSdss release level by calling the DFSMSdss module ADRRELVL. If access to this DFSMSdss module 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 this module.

Note that CICS VSAM Recovery for z/OS, which performs forward recovery, must be installed on the processor where forward recovery is to be done. CICS VSAM Recovery is required for forward recovery and backout of CICS VSAM data sets backed up with BWO and concurrent copy functions of DFSMS/MVS.

Table 2 cross-references the storage management component names of DFSMS to the previous names of the individual licensed programs:

Table 2. Cross-reference of DFSMS/MVS product terminology

Component name	Full DFSMS/MVS name	Previous product name
DFSMSdfp	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

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.

Which data sets are eligible for 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 (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 determines 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

For a KSDS (or an ESDS with AIX) that has frequent insert or update activity, it is only practical (and safe) to take BWO backups 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 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. However, 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 can define files as eligible for BWO in one of two ways.

Procedure

Decide which method you want to use for data sets:

- If your data set is accessed in RLS mode, you must define the BWO option in the ICF catalog. Defining BWO in the ICF catalog requires DFSMS 1.3.
- If your data set is accessed only in non-RLS mode, you can define the BWO option in either the ICF catalog or the CICS file definition.

Defining BWO in the ICF catalog requires DFSMS 1.3. For data sets that are accessed in RLS mode, the BWO option must be defined in the ICF catalog. Recovery attributes for data sets that are accessed only in non-RLS mode can be defined either in the ICF catalog or in the CICS FILE resource. If BWO is defined in the ICF catalog definition, by default it overrides any BWO option defined in the FILE resource. To force CICS to use the FILE resource attributes instead of the catalog recovery options, set the **NONRLSRECOV** system initialization parameter to FILEDEF.

Results

Do not define BWO for the CICSPlex SM data repository using the IDCAMS DEFINE CLUSTER definition in the ICF catalog because performance is degraded. See the *CICS Transaction Server for z/OS Installation Guide* for information on taking backups of the CICSPlex SM data repository.

Specifying BWO using access method services

There is a BWO parameter on the access method services DEFINE CLUSTER statement.

About this task

You can specify the BWO parameter as follows:

TYPECICS
The data set is eligible for BWO in CICS.

NO The data set is not eligible for BWO.

TYPEIMS
The data set is eligible for BWO in IMS, but CICS treats this as NO.

TYPEOTHER
The data set is eligible for BWO, but CICS treats this as NO.

If you specify BWO(TYPECICS), you might need to know whether the PTF associated with APAR OA04220 has been applied to your z/OS system. That APAR extended the capabilities of VSAM so that BWO processing for RLS is no longer disabled for non-recoverable files.

- If you specify BWO(TYPECICS), and you have also specified LOG(ALL) and a forward recovery log stream name, LOGSTREAMID(*logstream_name*), your file is recoverable, and the status of the PTF is not important.
- If you specify BWO(TYPECICS), and the PTF has been applied, BWO processing is now enabled for all VSAM RLS files whether you have specified LOG(ALL), LOG(NONE) or LOG(UNDO).

- But if you specify BWO(TYPECICS), and the PTF has not been applied, and you have not specified LOG(ALL) and a forward recovery log stream name, BWO processing for RLS remains disabled for such files. To achieve BWO for the file, you must either:
  - apply the PTF,
  - or specify LOG(ALL) and a forward recovery log stream name (if those actions are appropriate for the file in question).

If you omit the BWO parameter from the DEFINE statement, by default it is UNDEFINED in the ICF catalog, and the BWO attribute from the CICS file resource definition is used.

BWO(TYPECICS) is the equivalent of BACKUPTYPE(DYNAMIC) in a CICS file resource definition. All other values, including UNDEFINED, are treated by CICS as the equivalent of BACKUPTYPE(STATIC) in a CICS file resource definition. For simplicity, the CICS terms BACKUPTYPE(DYNAMIC) and BACKUPTYPE(STATIC) are used unless it is necessary to specifically mention the access method services BWO parameters.

The BWO options for the CSD are taken from the ICF catalog if they are defined there, and the system initialization parameters (CSDBKUP, CSDRECOV, and CSDFRLOG) are ignored.

Specifying BWO on CICS file resource definitions

You define a file as eligible for BWO with the BACKUPTYPE attribute on a CICS FILE resource definition in the CSD.

About this task

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 of zero and NORMALBKOUT state.

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

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

Removing BWO attributes

If you want to remove BWO attributes from your data sets, you must follow the correct procedure to avoid problems when taking subsequent back ups.

Procedure

1. Close the VSAM data set either by shutting down CICS normally or issuing the command **CEMT SET FILE CLOSED**. Do not perform an immediate shutdown, as CICS does not close the files and the status of BWO does not reset. The BWO status of your data sets will not be correct when you restart CICS.
2. Alter the attributes for the data set to remove the BWO options. You can use the IDCAMS **ALTER NULLIFY BWO** command.
3. Restart CICS or reopen the data set.

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 (non-RLS mode)” on page 216.
- 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 RACF Security Guide*.

BWO processing

The information in the remainder of this section 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. The DFSMSdfp field dictionary name is VVRSMFLG. For more information about the attribute flags used in connection with BWO, see *DFSMS/MVS Managing Catalogs*.
- 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  = hours
  MM  = minutes
  SS  = seconds
  T   = tenths of a second
  F   = + sign
  ```

  The DFSMSdfp field dictionary name is VVRRDATA.
- The BACKUPTYPE attribute in the CICS file resource definition (for DFSMS 1.2), or the BWO option in the ICF catalog (for DFSMS 1.3). When CICS has determined the BWO option from one of these sources, it stores the value into the data set name block (DSNB) for the base cluster at the time the first file referencing the data set is opened.

The attribute flags and recovery point are managed by VSAM in the primary data VSAM volume record (VVR) of the base cluster, which is 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 *DFSMS/MVS Managing Catalogs*.

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.

File opening

Different processing is done for each of the three cases when a file is opened for an update.

The following processing takes place:

- 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)

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, 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 occurs irrespective of whether BACKUPTYPE(DYNAMIC) or BACKUPTYPE(STATIC) is specified.

First file opened in non-RLS mode 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 is 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 message, and you can discard any BWO backups 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 the current time. CICS issues a message, and you can discard any BWO backups already taken in a previous batch window.

If BWO support is requested and the appropriate level of DFSMSdfp (as described in "BWO requirements" on page 204) 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 an attention message.

CICS also issues an attention 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. 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 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 inconsistent, the file-open operation fails with error messages. Either correct the CEDA definition, or REMOVE the DSNB after closing all files that are open against the base cluster data set. If the BACKUPTYPE attributes 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 an attention message; you can 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 an attention message; you should discard any BWO backup copies already taken in a previous batch window.

File closing (non-RLS mode)

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 indicating 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 213.

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

The way CICS closes files is determined by whether the shutdown is controlled, immediate, or uncontrolled.

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, 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.

Use the DFSMSdfp IGWABWO callable service to set the attributes (see "An assembler program that calls DFSMS callable services" on page 218 for an example of how to do this). Do 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
  - The original base cluster DSNB has been discarded
  - 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
  - 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 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 before 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 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 takes place. If this serialization fails, DFSMSdss checks the BWO attributes in the ICF catalog to see if the data set is eligible for BWO. If it is, a BWO backup is attempted. If it is not eligible, the backup attempt fails.

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

DFSMS access method services import and export operations do not support BWO in releases earlier than DFSMS 1.2. Access method services always serializes the data set before exporting it; and when the IMPORT function is used, the BWO attributes in the ICF catalog are not updated.

For DFSMS 1.2 onward, access method services supports the import and export of BWO attributes.

Invalid state changes for BWO attributes

CICS, DFSMSdfp, DFSMSdss, and an SMSVSAM server 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 is 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 must 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 must 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.

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.

However, there might be many files associated with the same data set; therefore, when a file is opened, the association between the file and the data set is recorded on the forward recovery log by a tie-up record. This information is also written to the log of logs. For non-BWO backups, the forward recovery utility uses this tie-up record to apply the log records to the correct data sets.

When a BWO is taken for a data set opened in RLS mode, DFSMSdss notifies each CICS region having an open ACB for the data set. On receipt of this notification,

each CICS allows all units of work with updates for the data set to complete, and then they write the tie-up records to the forward recovery log and the log of logs, and replies to DFSMSdss.

For BWO backups, it is usually not necessary for the forward recovery utility to process a log from file-open time. Therefore, the tie-up records 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 tie-up records if the activity keypoint frequency is high (such as for some large systems), CICS ensures that there is at least 30 minutes' separation between sets of tie-ups on the log.

Recovery point (non-RLS mode)

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 initialized 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 tie-up records and that completed earlier than the oldest uncommitted write to a forward recovery log.

Note:

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 updating 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 zero), BWO support is seriously affected because, after the file-open operation, no more tie-up records 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 and the log of logs.

If you do not use CICSVR, 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 DFSMS callable services" on page 218 for a sample program.
- Recognize the additional tie-up records on the forward recovery logs and, optionally, recognize tie-up records on the log of logs. These are written so that the forward recovery utility can quickly find the correct position without having to scan the entire forward recovery log.
- 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. 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:

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 before 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.

About this task

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 access method services BLDINDX command.

An assembler program that calls 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 'data_set_name'
*   Sxxx 100 'data_set_name'
*
*   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.
*    'data_set_name' is the fully-qualified 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
PROG     CSECT
PROG     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  PRGREAD
*
* 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   PRGBIT2
         ST   R4,BWOF1            Set BWO indicator 1 if required
PRGBIT2  DS   0H
         CLI  BWOC2,C'0'
         BE   PRGBIT3
         ST   R4,BWOF2            Set BWO indicator 2 if required
PRGBIT3  DS   0H
         CLI  BWOC3,C'0'
         BE   PRGCONT
         ST   R4,BWOF3            Set BWO indicator 3 if required
         B    PRGCONT
PRGREAD  DS   0H
         CLI  TRANFUNC,C'R'
         BNE  PRGABORT            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
PRGCONT  DS   0H
*
* OK, our parameters are set up, so create the address list, and make
* the call
*
         LOAD EP=IGWABWO,ERRET=PRGABORT
         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  PRGABORT
*
* 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  PRGREADO
*
* Got all the info we need, so put it out and exit
*
         EXEC CICS SEND TEXT FROM(SUCMSG) LENGTH(55) ERASE WAIT
*
         B    PRGEXIT
*
* It's a read so we also need the BWO flags for output
*
PRGREADO 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
*
PRGEXIT  DS 0H
         EXEC CICS RETURN
*
PRGABORT 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 PROG
```

Chapter 19. Disaster recovery

If your CICS system is normally available about 99 percent of the time, it would be wise to look at your disaster recovery plan. The same pressure that drives high availability drives the need for timely and current disaster recovery.

You must plan what level of disaster recovery you require for your CICS environment. If you are using DB2 or IMS, you can read more specific details related to database recovery in the following publications:

- *DB2 for z/OS Administration Guide*, for DB2 database recovery
- *IMS Operations Guide*, for IMS database recovery

See also the ITSC *Disaster Recovery Library: Planning Guide* for information that should help you set up a detailed disaster recovery plan if you use a combination of databases, such as DB2 and IMS.

Why have a disaster recovery plan?

If your business cannot continue to function without CICS, you must have a disaster recovery plan.

To build a disaster recovery plan you must take into account a number of items unique to disaster recovery:

- What data is vital to my business?
- How long can the data be unavailable?
- How current does the data need to be?
- What is the cost of a disaster to my company?
- What is the cost of my disaster recovery plan?
- Is performance after a disaster a consideration?
- What type of disaster is possible, or even likely, and how long will it affect my system?

You might consider some, or all, of your CICS applications as vital to the operations of your business. If all applications are vital, you need to recover all the data that your CICS systems use. If only some of your applications are vital, you have to determine what data is associated with those applications.

The length of time between the disaster and recovery of your vital applications is a key factor. If your business cannot continue without access to your CICS data, your disaster recovery plan must take this into account.

The time-sensitive nature of your recovered data can be an overriding factor. If your vital application is a high volume, high change application, recovering week-old data may not be acceptable—even hour-old data may be unacceptable. You may need to recover right up to the point of the disaster.

The type of disaster from which you plan to recover can determine where your disaster recovery site is located. If you foresee only fire and water damage to your computer floor, a disaster recovery site in the building next door may be

acceptable. If you are located in an area prone to hurricanes or earthquakes, for example, a disaster recovery site next door would be pointless.

When you are planning for disaster recovery, consider the cost of being unable to operate your business for a period of time. You have to consider the number of lost transactions, and the future loss of business as your customers go elsewhere. Your disaster recovery solution should not be more expensive than the loss from the disaster, unless your business would fail as a result of the outage caused by a disaster.

What is the real cost of your disaster recovery plan? Keeping track of the total cost of your disaster recovery procedures allows you to look at available options and judge the benefits and cost of each.

Your disaster recovery plan should include some performance considerations once you have recovered. Unless your disaster site mirrors your production site, you should determine acceptable levels of throughput and transaction times while operating from the disaster recovery site. The length of time it takes to recover your primary site can also determine what your disaster recovery site has to support in the interim.

In summary, be aware that risk, speed of recovery, and completeness of recovery have to be balanced against cost.

Disaster recovery testing

Testing is an essential part of disaster recovery planning. All too frequently, just creating a disaster recovery plan results in a false sense of security. If you don't test your disaster recovery plan, there's a risk that it won't work when you really need it.

Whenever possible, you should choose a remote site recovery strategy that you can test frequently. Testing your disaster recovery process has the following benefits:

- You know that your recovery plan works.
- You discover problems, mistakes, and errors, and can resolve them before you have to use the procedures.
- Your staff are educated in executing tests and managing disaster recovery situations.
- Your recovery plan becomes a living document.
- Members of your IT organization recognize the necessity of such a disaster recovery concept, and plan accordingly.
- Awareness of your disaster recovery strategy is increased.

After each test, use the detailed logs and schedules to identify any errors in your procedures, and eliminate them. Retest the changed procedures, and then incorporate them into your recovery plan. After changing the recovery plan, completely revise all existing disaster recovery documents.

Make frequent tests early in the implementation of your disaster recovery plan. Once you have removed the major problems, you can test less frequently. The frequency will depend on:

- The interval between major changes in your hardware and software
- How current you want to keep the recovery plan

- How critical and sensitive your business processes are: the more critical they are, the more frequently testing may be required.

Six tiers of solutions for off-site recovery

One blueprint for recovery planning describes a scheme consisting of six tiers of off-site recoverability (tiers 1-6), with a seventh tier (tier 0) that relies on local recovery only, with no off-site backup.

The tiers cover a full range of recovery options, ranging from no data moved off-site to full off-site copies with no loss of data. The following figures and text describe them from a CICS perspective.

Tier 0: no off-site data

Tier 0 is defined as having no requirements to save information off-site, establish a backup hardware platform, or develop a disaster recovery plan. Tier 0 is the no-cost disaster recovery solution.

Figure 17 summarizes the tier 0 solution.

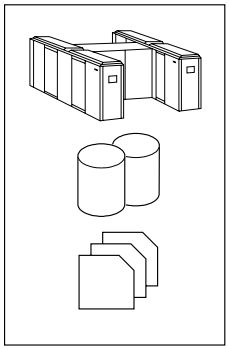


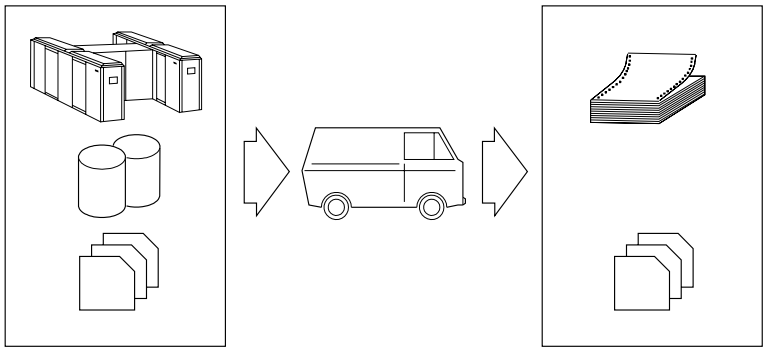
Figure 17. Disaster recovery tier 0: no off-site backup

Any disaster recovery capability would depend on recovering on-site local records. For most true disasters, such as fire or earthquake, you would not be able to recover your data or systems if you implemented a tier 0 solution.

Tier 1 - physical removal

Tier 1 is defined as having a disaster recovery plan, required data set backups physically removed and transported to an off-site storage facility, and optionally, a backup site, but without the required hardware currently installed.

Figure 18 on page 226 summarizes the tier 1 solution.



Approach

- Backups kept offsite
- Procedures and inventory offsite
- Recovery - install required hardware, restore system and data, reconnect to network

Recovery

- Relatively low cost
- Difficult to manage
- Most important applications resolved first
- Recovery possible, but may take a long time

Figure 18. Disaster recovery tier 1: physical removal

Your disaster recovery plan has to include information to guide the staff responsible for recovering your system, from hardware requirements to day-to-day operations.

The backups required for off-site storage must be created periodically. After a disaster, your data can only be as up-to-date as the last backup—daily, weekly, monthly, or whatever period you chose—because your recovery action is to restore the backups at the recovery site (when you have one).

This method may not meet your requirements if you need your online systems to be continuously available.

- If you require data from two or more subsystems to be synchronized, for example, from DB2 and VSAM, you would have to stop updates to both, then copy both sets of data.
- Such subsystems would both be unavailable for update until the longest running copy is finished.
- If you require a point-in-time copy for all your data, your application may be unavailable for updates for a considerable time.

The major benefit of tier 1 is the low cost. The major costs are the storage site and transportation.

The drawbacks are:

- Setting up a computer floor, and obtaining the necessary hardware after a disaster can take a long time.
- Recovery is to the point in time of your last backups. You have no computer record, such as forward recovery logs, of any transactions that took place after these backups were taken.
- The process is difficult to manage.
- Your disaster recovery plan is difficult to test.

Tier 1

Tier 1 provides a very basic level of disaster recovery. You will lose data in the disaster, perhaps a considerable amount. However, tier 1 allows you to recover and provide some form of service at low cost. You must assess whether the loss of data and the time taken to restore a service will prevent your company from continuing in business.

Tier 2 - physical removal with hot site

Tier 2, like tier 1, provides a very basic level of disaster recovery. You will lose data in the disaster, perhaps a considerable amount.

However, tier 2 allows you to recover and provide some form of service at low cost and more rapidly than tier 1. You must assess whether the loss of data and the time taken to restore a service will prevent your company from continuing in business.

Figure 19 summarizes the tier 2 solution.

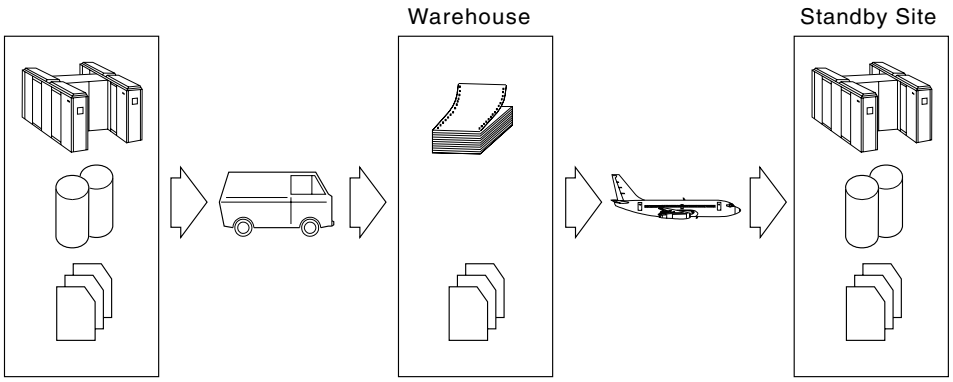


Figure 19. Disaster recovery tier 2: physical removal to a 'hot' standby site

Tier 2 is similar to tier 1. The difference in tier 2 is that a secondary site already has the necessary hardware installed, which can be made available to support the vital applications of the primary site. The same process is used to backup and store the vital data; therefore the same availability issues exist at the primary site as for tier 1.

The benefits of tier 2 are the elimination of the time it takes to obtain and setup the hardware at the secondary site, and the ability to test your disaster recovery plan.

The drawback is the expense of providing, or contracting for, a 'hot' standby site.

Tier 3 - electronic vaulting

Tier 3, like tiers 1 and 2, provides a basic level of disaster recovery. You will lose data in the disaster, perhaps a considerable amount of data.

The advantage of tier 3 is that you should be able to provide a service to your users quite rapidly. You must assess whether the loss of data will prevent your company from continuing in business.

Figure 20 summarizes the tier 3 solution.

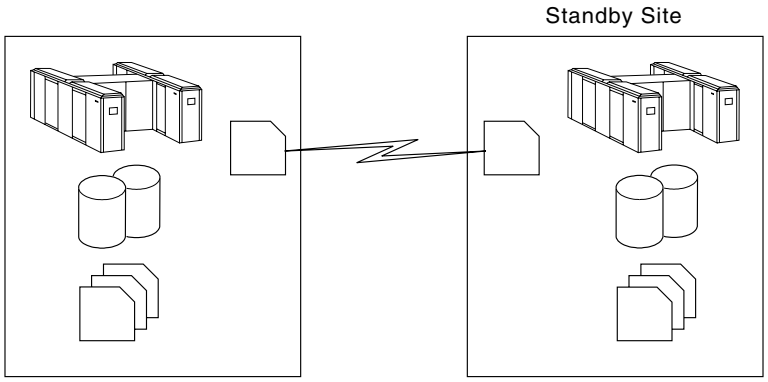


Figure 20. Disaster recovery tier 3: electronic vaulting

Tier 3 is similar to tier 2. The difference is that data is electronically transmitted to the hot site. This eliminates physical transportation of data and the off-site storage warehouse. The same process is used to backup the data, so the same primary site availability issues exist in tier 3 as in tiers 1 and 2.

The benefits of tier 3 are:

- Faster recovery, as the data does not have to be retrieved from off-site and down-loaded.
- No need to ship the backups manually to a warehouse and store them.

The drawbacks are the cost of reserving the DASD at the hot standby site, and that you must have a link to the hot site, and the required software, to transfer the data.

Procedures and documentation still have to be available at the hot site, but this can be achieved electronically.

Tier 0–3 solutions

Tiers 0 to 3 cover the disaster recovery plans of many CICS users. With the exception of tier 0, they employ the same basic design using a point-in-time copy of the necessary data. That data is then moved off-site to be used when required after a disaster.

Figure 21 on page 229 summarizes the solutions for tiers 0 through 3, and shows the approximate time required for a recovery with each tier of solution.

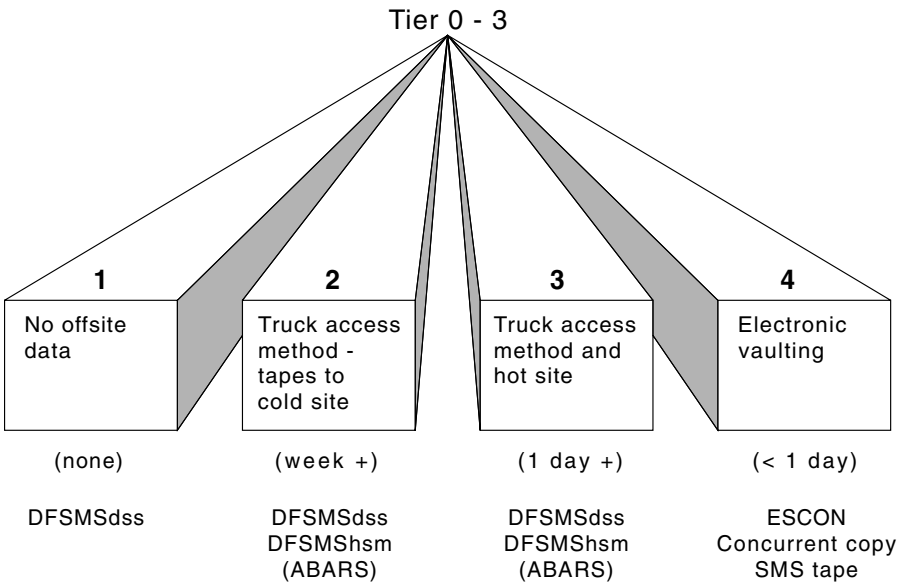


Figure 21. Disaster recovery tier 0-3: summary of solutions

The advantage of these methods is their low cost.

The disadvantages of these methods are:

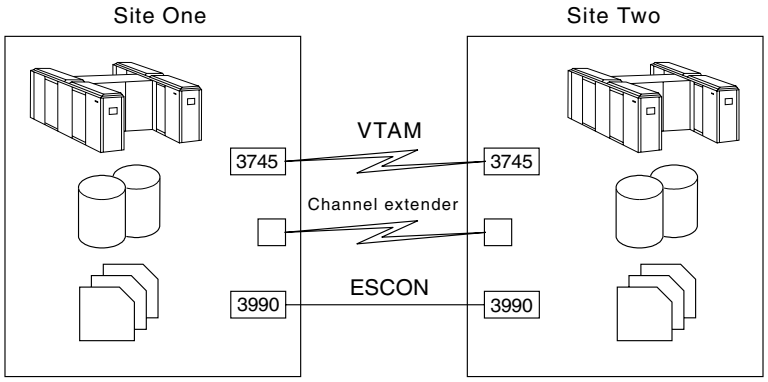
- Recovery is slow, and it can take days or weeks to recover.
- Any recovery is incomplete, because any updates made after the point-in-time backup are lost.
- Disaster recovery is risky, and difficulties in testing your disaster recovery plan could lead to incomplete recovery.

Tier 4 - active secondary site

Tier 4 provides a more advanced level of disaster recovery. You will lose data in the disaster, but only a few minutes- or hours-worth.

You must assess whether the loss of data will prevent your company from continuing in business, and what the cost of lost data will be.

Figure 22 on page 230 summarizes the tier 4 solution.



Approach

- Workload may be shared
- Site one backs up site two and the reverse
- Critical applications and data are online
- Switch network
- Recover other applications

Recovery

- Continuous transmission of data
- Dual online for critical data
- Network switching capability
- Recovery in minutes to hours

Figure 22. Disaster recovery tier 4: active secondary site

Tier 4 closes the gap between the point-in-time backups and current online processing recovery. Under a tier 4 recovery plan, site one acts as a backup to site two, and site two acts as a backup to site one.

Tier 4 duplicates the vital data of each system at the other's site. You must transmit image copies of data to the alternate site on a regular basis. You must also transmit CICS system logs and forward recovery logs, after they have been archived. Similarly, you must transmit logs for IMS and DB2 subsystems. Your recovery action is to perform a forward recovery of the data at the alternate site. This allows recovery up to the point of the latest closed log for each subsystem.

You must also copy to the alternate site other vital data that is necessary to run your system. For example, you must copy your load libraries and JCL. You can do this on a regular basis, or when the libraries and JCL change.

The benefits of tier 4 are:

- Recovery is fast, with the required hardware and software already in place at the secondary site.
- Recovery is more complete than in the tier 1 to 3 solutions. You can recover all data to the end of the log for each of your data subsystems.
- Recovery risk is low, because you can easily test your procedures.

The drawbacks are:

- Recovery is more difficult to manage, as you have to ensure that all the logs and copies are transmitted to the other system.
- This solution introduces synchronization problems. Logs for different data subsystems are transmitted at different times. When you complete your recovery at the secondary site, you could find that your VSAM data is complete up to 30 minutes before the disaster, whereas your DB2 data is complete up to 15 minutes before the disaster. If your data must be synchronized, you may have to develop further procedures to resynchronize data in different subsystems.

- Cost is higher than for the tier 1 to 3 solutions, because you need dedicated hardware, software, and communication links.

Tier 5 - two-site, two-phase commit

A tier 5 solution is appropriate for a custom-designed recovery plan with special applications. Because these applications must be designed to use this solution, it cannot be implemented at most CICS sites.

Figure 23 summarizes the tier 5 solution.

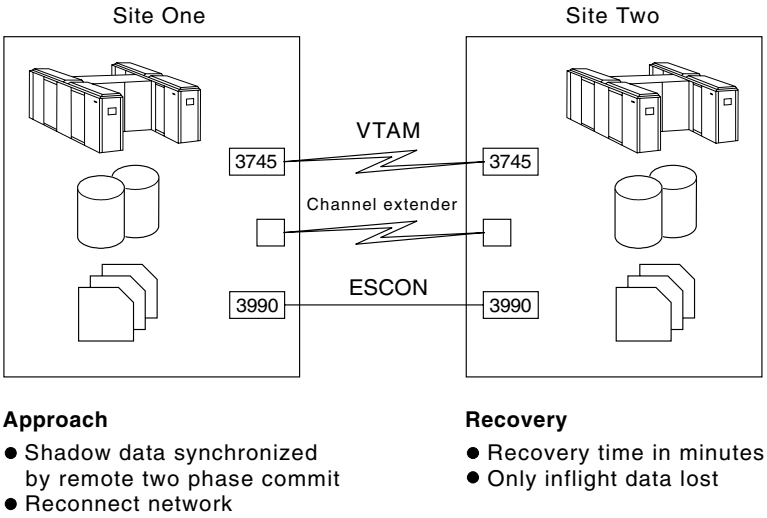


Figure 23. Disaster recovery tier 5: two site, two-phase commit

Tier 5, remote two-phase commit, is an application-based solution to provide high currency of data at a remote site. This requires partially or fully dedicated hardware at the remote site to keep the vital data in image format and to perform the two-phase commit. The vital data at the remote site and the primary site is updated or backed out as a single unit of work (UOW). This ensures that the only vital data lost would be from transactions that are in process when the disaster occurs.

Other data required to run your vital application has to be sent to the secondary site as well. For example, current load libraries and documentation has to be kept up-to-date on the secondary site.

The benefits of tier 5 are fast recovery using vital data that is current. The drawbacks are:

- The cost of maintaining and running two sites.
- The solution is application program dependent. You must write your applications so that they write data both locally and remotely, by transmitting data to a partner application at the remote site that also writes the data.
- This solution increases the time transactions take to respond. Your application program waits every time it needs to transmit data to the remote site.

Tier 6 - minimal to zero data loss

Tier 6 provides a very complete level of disaster recovery. You must assess whether the cost of achieving this level of disaster recovery is justified for your company.

Figure 24 summarizes the tier 6 solution.

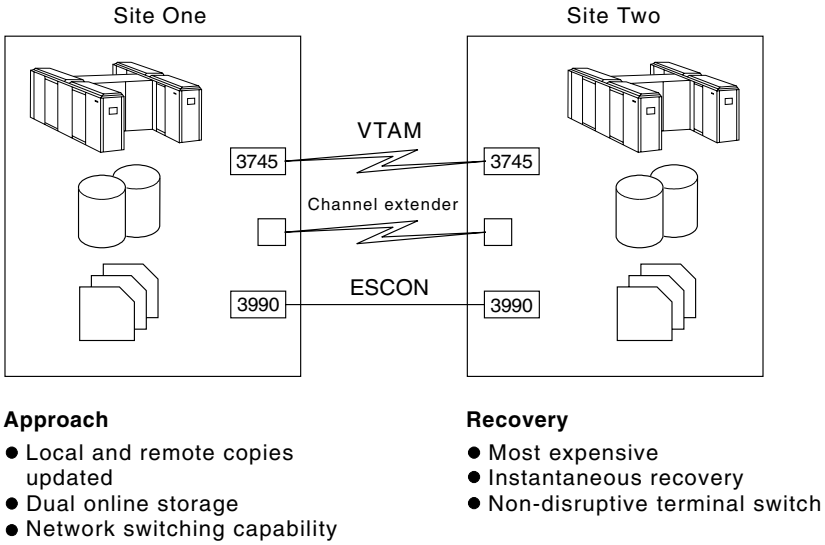


Figure 24. Disaster recovery tier 6: minimal to zero data loss

Tier 6, minimal to zero data loss, is the ultimate level of disaster recovery.

There are two tier 6 solutions, one hardware-based and the other software-based. For details of the hardware and software available for these solutions, see "Peer-to-peer remote copy (PPRC) and extended remote copy (XRC)" on page 234 (hardware) and "Remote Recovery Data Facility" on page 236 (software).

The hardware solution involves the use of IBM 3990-6 DASD controllers with remote and local copies of vital data. There are two flavors of the hardware solution: (1) peer-to-peer remote copy (PPRC), and (2) extended remote copy (XRC).

The software solution involves the use of Remote Recovery Data Facility (RRDF). RRDF applies to data sets managed by CICS file control and to the DB2, IMS, IDMS, CPCS, ADABAS, and SuperMICR database management systems, collecting real-time log and journal data from them. RRDF is supplied by E-Net Corporation and is available from IBM as part of the IBM Cooperative Software Program.

The benefits of tier 6 are:

- No loss of data.
- Recovery in a very short period of time.
- Emergency restart at remote site should be possible.

The drawbacks are the cost of running two sites and the communication overhead. If you are using the hardware solution based on 3990-6 controllers, you are limited in how far away your recovery site can be. If you use PPRC, updates are sent from the primary 3990-6 directly to the 3990-6 at your recovery site using enterprise systems connection (ESCON®) links between the two 3990-6 devices. The 3990-6 devices can be up to 43 km (26.7 miles) apart subject to quotation.

If you use XRC, the 3990-6 devices at the primary and recovery sites can be attached to the XRC DFSMS/MVS host at up to 43 km (26.7 miles) using ESCON links (subject to quotation). If you use three sites, one for the primary 3990, one to

support the XRC DFSMS/MVS host, and one for the recovery 3990, this allows a total of 86 km (53.4 miles) between the 3990s. If you use channel extenders with XRC, there is no limit on the distance between your primary and remote site.

For RRDF there is no limit to the distance between the primary and secondary sites.

Tier 4–6 solutions

This summary shows the three tiers and the various tools for each that can help you reach your required level of disaster recovery.

Figure 25 summarizes the solutions for tiers 4 through 6, and shows the approximate time required for a recovery with each tier of solution.

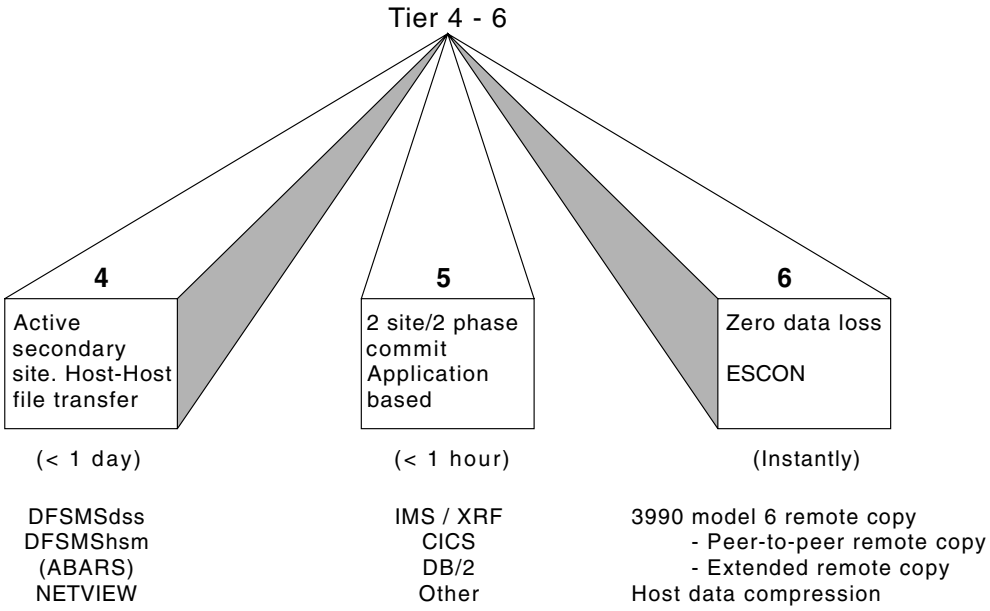


Figure 25. Disaster Recovery Tier 4-6: Summary of Solutions

Tier 4 relies on automation to send backups to the remote site. NetView® provides the ability to schedule work in order to maintain recoverability at the remote site.

Tier 5 relies on the two-phase commit processing supported by various database products and your application program's use of these features. Tier 5 requires additional backup processing to ensure that vital data, other than databases, is copied to the remote system.

Tier 6 is divided into two sections: software solutions for specific access methods and database management systems, and hardware solutions for any data.

RRDF can provide very high currency and recoverability for a wide range of data. However, it does not cover all the data in which you may be interested. For example, RRDF does not support load module libraries.

The 3990-6 hardware solution is independent of the data being stored on the DASD. PPRC and XRC can be used for databases, CICS files, logs, and any other data sets that you need to ensure complete recovery on the remote system.

Disaster recovery and high availability

This topic describes the tier 6 solutions for high availability and data currency when recovering from a disaster.

Peer-to-peer remote copy (PPRC) and extended remote copy (XRC)

PPRC and XRC are both 3990-6 hardware solutions that provide data currency to secondary, remote volumes.

Updates made to secondary DASD are kept in time sequence. This ensures that updates are applied consistently across volumes. PPRC and XRC also ensure that updates are applied in time sequence across control units as well. This sequencing offers a very high degree of data integrity across volumes behind different control units.

Because PPRC and XRC are hardware solutions, they are application, and data, independent. The data can be DB2, VSAM, IMS, or any other type of data. All your vital data on DASD can be duplicated off-site. This reduces the complexity of recovery. These solutions can also make use of redundant array of independent disks (RAID) DASD to deliver the highest levels of data integrity and availability.

PPRC synchronously shadows updates from the primary to the secondary site. This ensures that no data is lost between the data committed on the primary and secondary DASD. The time taken for the synchronous write to the secondary unit has an impact on your application, increasing response time. This additional time (required for each write operation) is approximately equivalent to a DASD fastwrite operation. Because the implementation of PPRC is almost entirely in the 3990-6, you must provide enough capacity for cache and non-volatile storage (NVS) to ensure optimum performance.

XRC is an asynchronous implementation of remote copy. The application updates the primary data as usual, and XRC then passes the updates to the secondary site. The currency of the secondary site lags slightly behind the primary site because of updates in transit. As part of XRC data management, updates to the secondary site are performed in the same sequence as at the primary site. This ensures data integrity across controllers and devices. Because XRC does not wait for updates to be made at the secondary site, the application's performance is not directly affected. XRC uses cache and non-volatile storage, so you must provide enough capacity to ensure optimum performance.

In the event of a disaster, check the state of all secondary volumes to ensure data consistency against the shadowed log data sets. This ensures that the same sequence of updates is maintained on the secondary volumes as on the primary volumes up to the point of the disaster. Because PPRC and XRC do not require restores or forward recovery of data, your restart procedures on the secondary system may be the same as for a short-term outage at the primary site, such as a power outage.

When running with PPRC or XRC, the data you replicate along with the databases includes:

- CICS logs and forward recovery logs
- CICS system definition (CSD) data sets, SYSIN data sets, and load libraries
- Recovery control (RECON) and restart data set (RDS) for IMS

- IMS write-ahead data set (WADS) and IMS online log data set (OLDS)
- ACBLIB for IMS
- Boot-strap data set (BSDS), the catalog and the directory for DB2
- DB2 logs
- Any essential non-database volumes

CICS applications can use non-DASD storage for processing data. If your application depends on this type of data, be aware that PPRC and XRC do not handle it.

For more information on PPRC and XRC, see *Planning for IBM Remote Copy*, SG24-2595-00, and *DFSMS/MVS Remote Copy Administrator's Guide and Reference*.

PPRC or XRC?

You need to choose between PPRC and XRC for transmitting data to your backup site. This topic compares the two methods to help you make your choice.

Choose PPRC as your remote copy facility if you:

- Require data currency at the secondary site
- Have your recovery site within ESCON distance
- Can accept some performance degradation
- Have a duplicate DASD configuration available at the remote site

The synchronous nature of PPRC ensures that, if you have a disaster at your main site, you lose only inflight transactions. The committed data recorded at the remote site is the same as that at the primary site.

Use PPRC for high value transactions

Consider PPRC if you deal with high value transactions, and data integrity in a disaster is more important to you than day-to-day performance. PPRC is more likely to be the solution for you if you characterize your business as being low volume, high value transactions; for example, a system supporting payments of thousands, or even millions, of dollars.

Choose XRC as your remote copy facility if you:

- Can accept that your data at the secondary site will be a few seconds behind the primary
- Have your secondary site outside ESCON distance
- Require high performance at the primary site

The asynchronous nature of XRC means that the remote site may have no knowledge of transactions that ran at the primary site, or does not know that they completed successfully. XRC ensures that the data recorded at the remote site is consistent (that is, it looks like a snapshot of the data at the primary site, but the snapshot may be several seconds old).

Use XRC for high volume transactions

Consider XRC if you deal with low value transactions, and data integrity in a disaster is less important to you than day-to-day performance. XRC is more likely to be the solution for you if you characterize your business as being high volume, low value transactions; for example, a system supporting a network of ATMs,

where there is a high volume of transactions, but each transaction is typically less than 200 dollars in value.

Other benefits of PPRC and XRC

PPRC or XRC may eliminate the need for disaster recovery backups to be taken at the primary site, or to be taken at all.

PPRC allows you to temporarily suspend the copying of updates to the secondary site. This allows you to suspend updates at the secondary site so that you can make image copies or backups of the data there. After the backups are complete, you can reestablish the pairing of data sets on the primary and secondary sites. Updates to the primary that have been recorded by the 3990-6 are applied to the secondary to resynchronize the pair.

XRC supports the running of concurrent copy sessions to its secondary volumes. This enables you to create a point-in-time copy of the data.

PPRC and XRC also allow you to migrate data to another or larger DASD of similar geometry, behind the same or different control units at the secondary site. This can be done for workload management or DASD maintenance, for example.

Forward recovery

Whether you use PPRC or XRC, you have two fundamental choices. You can either pair volumes containing both the data and the log records or you can pair only the volumes containing the log records.

In the first case you should be able to perform an emergency restart of your systems, and restore service very rapidly. In the latter case you would need to use the log records, along with an image copy transmitted separately, to perform a forward recovery of your data, followed by an emergency restart.

Pairing the data volumes, as well as the log volumes, costs more because you have more data flowing between the sites, and therefore you need a greater bandwidth to support the flow. In theory you can restart much faster than if you have to perform a forward recovery. When deciding which to use, you must determine whether this method is significantly faster, and whether you think it is worth the additional costs.

Remote Recovery Data Facility

The Remote Recovery Data Facility (RRDF), Version 2 Release 1, a product of the E-Net Corporation, minimizes data loss and service outage time in the event of a disaster by providing a real-time remote logging function.

Real-time remote logging provides data currency at the remote site, enabling the remote site to recover databases within seconds of the outage—typically in less than 1 second.

RRDF runs in its own address space. It provides programs that run in the CICS or database management system address space. These programs are invoked through standard interfaces—for example, at exit points associated with writing out log records.

The programs that run in the CICS or database management system address space use MVS cross-memory services to move log data to the RRDF address space. The RRDF address space maintains a virtual storage queue at the primary site for records awaiting transmission, with provision for spill files if communication

between the primary and secondary sites is interrupted. Remote logging is only as effective as the currency of the data that is sent off-site. RRDF transports log stream data to a remote location in real-time, within seconds of the log operation at the primary site.

When the RRDF address space at the remote site receives the log data, it formats it into archived log data sets. Once data has been stored at the remote site, you can use it as needed to meet business requirements. The recovery process uses standard recovery utilities. For most data formats, first use the log data transmitted by RRDF in conjunction with a recent image copy of the data sets and databases that you have to recover. Then perform a forward recovery. If you are using DB2, you have the option of applying log records to the remote copies of your databases as RRDF receives the log records.

If you use DB2, you can use the optional RRDF log apply feature. With this feature you can maintain a logically consistent "shadow" copy of a DB2 database at the remote site. The RRDF log apply feature updates the shadow database at selected intervals, using log data transmitted from the primary site. Thus restart time is shortened because the work needed after a disaster is minimal. The currency of the data depends on the log data transmitted by RRDF and on how frequently you run the RRDF log apply feature. The RRDF log apply feature also enhances data availability, as you have read access to the shadow copy through a remote site DB2 subsystem. RRDF supports DB2 remote logging for all environments, including TSO, IMS, CICS, batch, and call attach.

At least two RRDF licenses are required to support the remote site recovery function, one for the primary site and one for the remote site. For details of RRDF support needed for the CICS Transaction Server, see "Remote Recovery Data Facility support" on page 239.

Choosing between RRDF and 3990-6 solutions

About this task

Table 3 summarizes the characteristics of the products you can use to implement a tier 6 solution. You must decide which solution or solutions is most appropriate for your environment.

Table 3. Selecting a tier 6 implementation. This table compares the strengths of the tier 6 solutions.

	RRDF	**3990-6**
Data type supported	Various data sets[1]	Any on DASD
Database shadowing	Optional. Available for DB2 and IDMS only	Optional
Forward recovery required	Yes	Depends on implementation
Distance limitation	None	About 40 km for ESCON. Unlimited for XRC with channel extenders
Note: [1] Data sets managed by CICS file control and the DB2, IMS, IDMS, CPCS, ADABAS, and SuperMICR database management systems.		

Disaster recovery personnel considerations

When planning for disaster recovery, you need to consider personnel issues.

You should ensure that a senior manager is designated as the disaster recovery manager. The recovery manager must make the final decision whether to switch to a remote site, or to try to rebuild the local system (this is especially true if you have adopted a solution that does not have a warm or hot standby site).

You must decide who will run the remote site, especially during the early hours of the disaster. If your recovery site is a long way from the primary site, many of your staff will be in the wrong place.

Finally, and to show the seriousness of disaster recovery, it is possible that some of your key staff may be severely injured and unable to take part in the recovery operation. Your plans need to identify backups for all your key staff.

Returning to your primary site

One aspect of disaster recovery planning which can be overlooked is the need to include plans for returning operations from the recovery site back to the primary site (or to a new primary site if the original primary site cannot be used again).

About this task

Build the return to normal operations into your plan. The worst possible time to create a plan for moving back to your primary site is after a disaster. You will probably be far too busy to spend time building a plan. As a result, the return to your primary site may be delayed, and may not work properly.

Disaster recovery facilities

This section looks at the various alternatives for achieving disaster recovery with CICS and also looks at utilities that can aid the process.

MVS system logger recovery support

The MVS system logger provides support that enables a recovery resource manager to be associated with a log stream (that is, a local recovery resource manager operating on behalf of a remote site).

The name of the recovery resource manager is specified when a new log stream definition is created or an existing log stream definition is updated. When a recovery resource manager connects to the log stream, through the IXGCONN service, it requests that a resource manager-owned exit be given control when specified events occur. When such an event occurs, the MVS system logger invokes the exit specified by the resource manager and passes it details of the event. It is then the responsibility of the recovery resource manager to transmit log records to a remote site.

The remote site needs to be able to import log streams transmitted by the recovery resource manager; this too is provided by MVS system logger services. Importation is the process whereby log blocks resident in one log stream (the source log stream) are created in (or copied into) another log stream (the target log stream) while maintaining (in the target log stream) the same MVS system logger-assigned log block id and GMT time stamp that is assigned to the source log stream. The result of a log stream import is a copy of the source log stream in the target log stream.

CICS VSAM Recovery QSAM copy

CICS VSAM Recovery (CICS VR) provides a QSAM copy function that can copy MVS log streams to a QSAM data set.

Copies of the QSAM data can be sent either electronically or physically to the remote site. On arrival at the remote site, you can use the MVS system logger import services to put the log records into an MVS system logger log stream. Alternatively, you can use CICS VR to perform forward recovery of a data set using the QSAM data directly.

Remote Recovery Data Facility support

The Remote Recovery Data Facility (RRDF) product from the E-Net Corporation supports the CICS log manager.

RRDF Version 2 Release 1 uses the disaster recovery services (for export and import of log streams) provided by the MVS system logger. RRDF connects to a log stream at the local site where the resource manager exit is specified, to register its interest. The recovery manager is given control whenever writes or deletes occur. Typically, writes are intercepted for transmission to the remote site. Delete requests are intercepted to prevent CICS from deleting system log records before RRDF has sent them to the remote site. RRDF at the remote site receives the transmitted log records, establishes an import connection to a log stream, and imports the log records.

CICS VR shadowing

CICS VR provides a data shadowing facility. Shadowing helps to reduce recovery time by applying forward recovery logs periodically at the remote site. See CICS VR documentation for a complete explanation.

CICS emergency restart considerations

It is important to consider the differences between CICS Transaction Server and earlier releases of CICS when planning for off-site recovery.

Indoubt and backout failure support

Support during emergency restarts for units-of-work that failed indoubt or failed during backout is provided by the CICS recovery manager.

This support is available at the remote site only if the system log is transmitted and the CICS regions at the remote site are running under CICS Transaction Server.

It is possible for system log records to be transmitted to the remote site for units-of-work that subsequently become indoubt-failed or backout-failed. The log records for failed units of work could be moved to the secondary log stream at the local site. However, resource managers such as RRDF are aware of such movements and act accordingly.

Remote site recovery for RLS-mode data sets

CICS provides support for remote site recovery where VSAM data sets are used in RLS mode at the primary site. Using this RLS support for remote recovery, you can switch over to the remote site without suffering indeterminate or unreported loss of data integrity.

If a disaster occurs at the primary site, your disaster recovery procedures should include recovery of VSAM data sets at the designated remote recovery site. You can then emergency restart the CICS regions at the remote site so that they can backout any uncommitted data. Special support is needed for RLS because record locks, which were protecting uncommitted data from being updated by other transactions at the primary site, are not present at the remote site. You invoke CICS RLS support for off-site recovery using the OFFSITE system initialization parameter.

The OFFSITE system initialization parameter protects all RLS data sets until all emergency restart recovery is complete. You can specify this OFFSITE system initialization parameter at run-time only—it cannot be specified and assembled in the SIT—and it is valid only when START=AUTO is specified. You specify OFFSITE=YES when restarting CICS regions at a remote site when recovery involves VSAM data sets opened in RLS mode.

When you specify OFFSITE=YES, file control performs RLS off-site recovery processing, under which file control does not allow any new RLS access during startup. With RLS recovery in operation during an emergency restart, CICS does not allow any data sets to be accessed in RLS mode until:

- CICS has completed all outstanding RLS recovery work.
- CICS file control has issued a message requesting a "GO" response from an operator when all CICS regions have completed RLS recovery processing.
- An operator has replied to the message.

  Operators should reply to the message only when all the CICS regions being restarted with OFFSITE=YES have issued the message, indicating that they have all completed their RLS recovery.

A CICS-supplied sample NetView EXEC, DFH$OFAR, is provided to automate the detection of, and reply to, the WTOR console messages. For more information, see the prolog in the source member of the CICSTS41.CICSSDFHSAMP. library.

Final summary

Your disaster recovery plan will be truly tested only at a very difficult time for your business—during a disaster. Careful planning and thorough testing may mean the difference between a temporary inconvenience and going out of business.

The goal of your disaster recovery plan is to get your CICS systems back online. The currency of the data and the time it takes to get back online is a function of which disaster recovery tier you use. Unless legal requirements for disaster recovery dictate the type of disaster recovery you must have, the choice of tiers is usually a business decision.

Making your disaster recovery work requires a good plan, up-to-date documentation, and regular testing.

Part 4. Appendixes

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply in the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM United Kingdom Laboratories, MP151, Hursley Park, Winchester, Hampshire, England, SO21 2JN.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Bibliography

CICS books for CICS Transaction Server for z/OS

General

CICS Transaction Server for z/OS Program Directory, GI13-0536
CICS Transaction Server for z/OS What's New, GC34-6994
CICS Transaction Server for z/OS Upgrading from CICS TS Version 2.3, GC34-6996
CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.1, GC34-6997
CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.2, GC34-6998
CICS Transaction Server for z/OS Installation Guide, GC34-6995

Access to CICS

CICS Internet Guide, SC34-7021
CICS Web Services Guide, SC34-7020

Administration

CICS System Definition Guide, SC34-6999
CICS Customization Guide, SC34-7001
CICS Resource Definition Guide, SC34-7000
CICS Operations and Utilities Guide, SC34-7002
CICS RACF Security Guide, SC34-7003
CICS Supplied Transactions, SC34-7004

Programming

CICS Application Programming Guide, SC34-7022
CICS Application Programming Reference, SC34-7023
CICS System Programming Reference, SC34-7024
CICS Front End Programming Interface User's Guide, SC34-7027
CICS C++ OO Class Libraries, SC34-7026
CICS Distributed Transaction Programming Guide, SC34-7028
CICS Business Transaction Services, SC34-7029
Java Applications in CICS, SC34-7025

Diagnosis

CICS Problem Determination Guide, GC34-7034
CICS Performance Guide, SC34-7033
CICS Messages and Codes, SC34-7035
CICS Diagnosis Reference, GC34-7038
CICS Recovery and Restart Guide, SC34-7012
CICS Data Areas, GC34-7014
CICS Trace Entries, SC34-7013
CICS Supplementary Data Areas, GC34-7015
CICS Debugging Tools Interfaces Reference, GC34-7039

Communication

CICS Intercommunication Guide, SC34-7018
CICS External Interfaces Guide, SC34-7019

Databases

CICS DB2 Guide, SC34-7011
CICS IMS Database Control Guide, SC34-7016

CICS Shared Data Tables Guide, SC34-7017

CICSPlex SM books for CICS Transaction Server for z/OS

General

CICSPlex SM Concepts and Planning, SC34-7044
CICSPlex SM Web User Interface Guide, SC34-7045

Administration and Management

CICSPlex SM Administration, SC34-7005
CICSPlex SM Operations Views Reference, SC34-7006
CICSPlex SM Monitor Views Reference, SC34-7007
CICSPlex SM Managing Workloads, SC34-7008
CICSPlex SM Managing Resource Usage, SC34-7009
CICSPlex SM Managing Business Applications, SC34-7010

Programming

CICSPlex SM Application Programming Guide, SC34-7030
CICSPlex SM Application Programming Reference, SC34-7031

Diagnosis

CICSPlex SM Resource Tables Reference, SC34-7032
CICSPlex SM Messages and Codes, GC34-7035
CICSPlex SM Problem Determination, GC34-7037

Other CICS publications

The following publications contain further information about CICS, but are not provided as part of CICS Transaction Server for z/OS, Version 4 Release 1.

Designing and Programming CICS Applications, SR23-9692

CICS Application Migration Aid Guide, SC33-0768

CICS Family: API Structure, SC33-1007

CICS Family: Client/Server Programming, SC33-1435

CICS Family: Interproduct Communication, SC34-6853

CICS Family: Communicating from CICS on System/390, SC34-6854

CICS Transaction Gateway for z/OS Administration, SC34-5528

CICS Family: General Information, GC33-0155

CICS 4.1 Sample Applications Guide, SC33-1173

CICS/ESA 3.3 XRF Guide , SC33-0661

Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully.

You can perform most tasks required to set up, run, and maintain your CICS system in one of these ways:

- using a 3270 emulator logged on to CICS
- using a 3270 emulator logged on to TSO
- using a 3270 emulator as an MVS system console

IBM Personal Communications provides 3270 emulation with accessibility features for people with disabilities. You can use this product to provide the accessibility features you need in your CICS system.

Index

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Readers' Comments — We'd Like to Hear from You

CICS Transaction Server for z/OS
Version 4 Release 1
Recovery and Restart Guide

Publication No. SC34-7012-03

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your 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. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:
- Send your comments to the address on the reverse side of this form.
- Send a fax to the following number: +44 1962 816151
- Send your comments via email to: idrcf@uk.ibm.com

If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	Email address

Readers' Comments — We'd Like to Hear from You
SC34-7012-03

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM United Kingdom Limited
User Technologies Department (MP095)
Hursley Park
Winchester
Hampshire
United Kingdom
SO21 2JN

Fold and Tape | **Please do not staple** | Fold and Tape

SC34-7012-03

Cut or Fold
Along Line

SC34-7012-03