IBM CICS Transaction Affinities Utility MVS/ESA

User's Guide

SC33-1159-01

IBM CICS Transaction Affinities Utility MVS/ESA

User's Guide

SC33-1159-01

Note!

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

First edition (January 1994)

This edition applies to the IBM Customer Information Control System (CICS) program offering, the CICS Transaction Affinities Utility MVS/ESA, program number 5696-582, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Consult the latest edition of the applicable IBM system bibliography for current information on this product.

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

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

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

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

Contents

Notices

The following paragraph does not apply in any country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, 208 Harbor Drive, Stamford, Connecticut 06904, U.S.A.

Trademarks and service marks

The following terms, used in this publication, are trademarks or service marks of IBM Corporation in the United States or other countries:

	CICS	CICS/ESA	CICS/MVS
#	CICSPlex SM	CSP	DB2
	IBM	MVS/ESA	RETAIN

Preface

What this book is about

This book describes the **IBM CICS Transaction Affinities Utility MVS/ESA**. It explains what the utility does, how to install it, and how to run the various components of the utility.

Who this book is for

This book is for CICS system programmers who may be planning to use the CICS **dynamic transaction routing facility** for workload balancing, and need to determine whether any of the transactions in their CICS applications use programming techniques that cause inter-transaction affinity. It can also be used by application programmers to detect whether application programs they are developing are likely to cause inter-transaction affinity.

In particular, this book is of interest to CICS system programmers who are planning to use **CICSPlex System Manager/ESA** (CICSPlex SM) for workload balancing. For more information about CICSPlex SM, see the *CICSPlex System Manager Concepts and Planning* manual, GC33-0786.

It is also of use if you are planning to implement asynchronous processing using CICS function shipping or are planning to use the **transaction isolation** facility of CICS/ESA Version 4 Release 1 (CICS/ESA 4.1).

What you need to know to understand this book

You need to be familiar with the CICS application programing interface (API) and the various programming techniques that are available to CICS application programmers. In particular you should be familiar with those techniques that CICS application programs can use to pass data one to another, such as sharing common storage, and techniques to synchronize their execution.

Chapter 1, “Introduction to transaction affinities” on page 1 gives a brief introduction to the inter-transaction affinity that can be caused by some of these techniques; for a full discussion of transaction affinities, see the *CICS Dynamic Transaction Routing in a CICSplex*, SC33-1012.

How to Use this Book

This book is intended to be read sequentially so that you understand (1) how to install the affinity utility, and (2) how to run the separate components. Later, when you are familiar with the utility, you need only reference the chapter that deals with the particular component that you want to run.

Notes on Terminology

MVS/ESA
The MVS/ESA operating system, generally referred to as **MVS**.

Customer Information Control System/Enterprise System Architecture
CICS/ESA, generally referred to as **CICS**.

Argument zero
When an EXEC CICS command is translated and compiled, it results in an encoded parameter list to be used with a call statement. The first parameter in this list is a constant known as the CICS **argument zero**. The first two bytes of this constant identify the command; for example, X'0A04' identifies it as a READQ TS command.

Summary of changes

The following changes have been made to this book since it was published. The changes are indicated by a hash (#) symbol to the left of the changes.

- Changes affecting CICS/ESA 4.1, for PTF UN61942 and UN71013:
  - For CICS/ESA 4.1, the Detector intercepts the signoff message DFHSN1200 (and the DFHZC3462 and DFHZC5966 messages as for CICS/ESA 3.3.) For more information, see page 11.
  - For a CICS/ESA 4.1 region, you should specify the DSHIPINT=0 system initialization parameter, to ensure that the CICS Transaction Affinities Utility observes terminal logoffs correctly. For more information, see page 27.
- A new section has been added, “Using the IBM Cross System Product” on page 55, describing the use of the Transaction Affinities Utility with programs developed using the IBM Cross System Product.

Chapter 1. Introduction to transaction affinities

This chapter

Provides a brief introduction to the concept of transaction affinities and the associated CICS programming techniques, and highlights the significance of transaction affinities in a dynamic transaction routing environment. For more information about transaction affinities, see the *CICS Dynamic Transaction Routing in a CICSplex* manual, SC33-1012.

CICS/ESA dynamic transaction routing

The IBM Customer Information Control System (CICS) has been handling customers' online transaction processing requirements for over 25 years. In that time, CICS has been extensively enhanced to meet the ever-growing needs of business applications, and to exploit the capabilities of modern computer processors and communication systems. One of the most significant enhancements in recent times is the addition of the dynamic transaction routing facility.

Originally, a full-function CICS ran in a single address space (region) within the MVS environment. Currently, most CICS users use some form of intercommunications to operate multiple, interconnected, CICS regions—a CICSplex. Using the CICS multiregion operation (MRO) facility, a CICSplex typically consists of one or more terminal-owning regions (TOR), and a number of application-owning regions to which the TORs route the incoming transactions for processing.

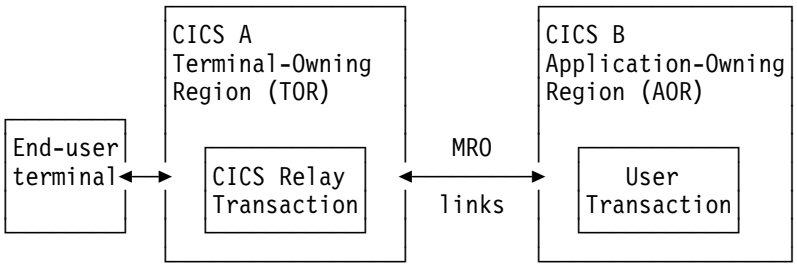


Figure 1. The CICS transaction routing facility

Static versus dynamic transaction routing

Prior to CICS/ESA Version 3, TORs routed transactions to the AORs predefined in transaction resource definitions by the system programmer. This static form of transaction routing adds to the system administration burden of the system programmer, because when transaction workloads have to be rebalanced across the AORs, transaction resource definitions have to be modified accordingly.

The dynamic transaction routing facility introduced in CICS/ESA removes the need to specify the remote system name of a target AOR in the transaction definition. Instead, you let the TOR determine dynamically to which AOR it should route

incoming transactions. Unlike static routing, where there can only ever be one AOR to which the TOR can route a transaction, with dynamic routing you can create several AORs with the capability of processing any given workload, and let the TOR choose the best one from a candidate list.

The benefits of dynamic transaction routing

Being able to route transactions to AORs dynamically offers many benefits in an online transaction processing (OLTP) system. The user can achieve:

- Improved performance
- Improved availability
- Simplified systems management.

What does dynamic routing cost?

Of course, the CICS-supplied code cannot determine where to send a transaction, as this depends on your CICS environment and routing policies. It needs a facility that you can use to specify your routing policies in a way that CICS can use them. This can be a user-written dynamic transaction routing program used to supply the name of a suitable AOR. You can define the name of a dynamic routing program to CICS on the DTRPGM system initialization parameter.

At the basic level, a dynamic routing program simply contains tables of user transaction identifiers, with the matching system identifiers (SYSIDs) of the AORs that can process the transactions. At the highest and most sophisticated level, the dynamic routing program would also be capable of detecting and managing any special factors that might affect transaction routing.

One factor that can affect the otherwise free choice of AOR is the use of particular CICS programming techniques that transactions use to pass data from one to another.

Transaction affinities

CICS transactions use many different techniques to pass data from one to another. Some of these techniques require that the transactions exchanging data must execute in the same CICS region, and therefore impose restrictions on the dynamic routing of transactions. If transactions exchange data in ways that impose such restrictions, there is said to be an affinity between them.

Basically, there are two categories of affinity:

- Inter-transaction affinity
- Transaction-system affinity.

The restrictions on dynamic transaction routing caused by transaction affinities depend on the duration and scope of the affinities. Clearly, the ideal situation for a dynamic transaction routing program is for there to be no transaction affinities at all, which means there are no restrictions in the choice of available AORs for dynamic transaction routing. However even when transaction affinities do exist, there are limits to the scope of these affinities; the scope of the affinity being determined by:

Affinity relation
Determines how the dynamic transaction routing program is to select a target AOR for a transaction instance associated with the affinity.

Affinity lifetime
Determines when the affinity is ended.

Inter-transaction affinity

Inter-transaction affinity is an affinity between two or more CICS transactions. It is caused by the transactions using techniques to pass information between one another, or to synchronize activity between one another, in a way that requires the transactions to execute in the same CICS region. This type of affinity is inter-transaction affinity, where a set of transactions share a common resource and/or coordinate their processing. Inter-transaction affinity, which imposes restrictions on the dynamic routing of transactions, can occur in the following circumstances:

- One transaction terminates, leaving “state data” in a place that a second transaction can access only by running in the same CICS region as the first transaction.
- One transaction creates data that a second transaction accesses while the first transaction is still running. For this to work safely, the first transaction usually waits on some event, which the second transaction posts when it has read the data created by the first transaction. This synchronization technique requires that both transactions are routed to the same CICS region.

Transaction-system affinity

Transaction-system affinity is an affinity between a transaction and a particular CICS region (that is, it is not an affinity between transactions themselves). It is caused by the transaction interrogating or changing the properties of that CICS region.

Transactions with affinity to a particular system, rather than another transaction, are not eligible for dynamic transaction routing. In general, they are transactions that use INQUIRE and SET commands or have some dependency on global user exit programs.

Affinity relations and lifetimes

An affinity relation can be classified as one of the following:

Global A group of transactions where ***all*** instances of all transactions in the group that are initiated from any terminal must execute in the same AOR for the lifetime of the affinity. The affinity lifetime for global relations can be system or permanent.

LUname A group of transactions whose affinity relation is defined as LUname is one where ***all*** instances of all transactions in the group that are initiated from the ***same terminal*** must execute in the same AOR for the lifetime of the affinity. The affinity lifetime for LUname relations can be pseudo-conversation, logon, system, or permanent.

Userid A group of transactions whose affinity relation is defined as userid is one where ***all*** instances of the transactions that are initiated from a terminal and executed on behalf of the **same userid**, must execute in the same AOR for the lifetime of the affinity. The affinity lifetime for userid relations can be pseudo-conversation, signon, system, or permanent.

An affinity lifetime can be classified as one of:

System The affinity lasts for as long as the target AOR exists, and ends whenever the AOR terminates (at a normal, immediate, or abnormal termination). (The resource shared by transactions that take part in the affinity is ***not*** recoverable across CICS restarts.)

Permanent The affinity extends across all CICS restarts. (The resource shared by transactions that take part in the affinity ***is*** recoverable across CICS restarts.) This is the most restrictive of all the inter-transaction affinities.

Pseudo-conversation
The (LUname or userid) affinity lasts for the whole pseudo-conversation, and ends when the pseudo-conversation ends at the terminal.

Logon The (LUname) affinity lasts for as long as the terminal remains logged on to CICS, and ends when the terminal logs off.

Signon The (userid) affinity lasts for as long as the user is signed on, and ends when the user signs off.

Notes:

1. For userid affinities, the pseudo-conversation and signon lifetimes are only possible in those situations where only one user per userid is permitted. Such lifetimes are meaningless if multiple users are permitted to be signed on with the same userid at the same time (at different terminals).

   APAR PN68267

   Applied.

2. If an affinity is both userid and LUname (that is, all instances of all transactions in the group were initiated from the same terminal *and* by the same userid), then LUname takes precedence.

CICS programming techniques for transaction affinity

Associated with transaction affinity, there are three broad categories of CICS programming techniques:

- Safe programming techniques
- Unsafe programming techniques
- Suspect programming techniques.

The following sections outline these programming techniques.

Safe programming techniques

The programming techniques in the generally safe category are the use of:

- The communication area (COMMAREA) on CICS RETURN commands
- A TCT user area (TCTUA) optionally available for each terminal defined to CICS.

Unsafe programming techniques

The programming techniques in the unsafe category are the use of:

- Long-life shared storage:
  - The common work area (CWA)
  - GETMAIN SHARED storage
  - Storage obtained via a LOAD PROGRAM HOLD
- Task-lifetime local storage shared by synchronized tasks.
- Synchronization or serialization of tasks using CICS commands:
  - WAIT EVENT / WAIT EXTERNAL / WAITCICS
  - ENQ / DEQ

Suspect programming techniques

Some programming techniques may, or may not, create affinity depending on exactly how they are implemented. A good example is the use of temporary storage. Application programs using techniques in this category must be checked to determine whether they will work without restrictions in a dynamic transaction routing environment.

The programming techniques in the suspect category are the use of:

- Temporary storage queues with restrictive naming conventions
- Synchronization or serialization of tasks using CICS commands:
  - RETRIEVE WAIT / START
  - START / CANCEL REQID
  - DELAY / CANCEL REQID
  - POST / CANCEL REQID

Avoiding the effects of transaction affinity

In a dynamic transaction routing environment, your dynamic transaction routing program must take account of transaction affinity in order to route transactions effectively. Ideally, you should avoid creating application programs that cause affinity, in which case the problem does not exist. However, where existing applications are concerned it is important that you determine whether they are affected by transaction affinity before using them in a dynamic transaction routing environment. This is the task the CICS Transaction Affinities Utility is designed to help you with.

CICS/ESA 4.1 transaction isolation

The transaction isolation function of CICS/ESA 4.1 offers storage protection between application programs, ensuring that one application does not accidentally overwrite the storage of another application.

Transaction isolation ensures that user-key[1] programs execute in their own subspace, with appropriate access to any shared storage, or to CICS storage. Thus a user transaction is limited to its own ‘view’ of the address space. In

[1] **User key** defines both the storage and execution key for user application programs.

general, transaction isolation ensures that each user-key program is allocated a separate (unique) subspace, with appropriate access to any shared storage or to CICS storage. They do not have any access to user-key task-lifetime storage of other tasks.

Existing applications should run unmodified provided they conform to transaction isolation requirements. However, a minority of applications may need special definition if they:

- Issue MVS macros directly, or
- Modify CICS control blocks, or
- Have a legitimate need for one task to access, or share, another task's storage.

Some existing transactions may share task-lifetime storage in various ways, and this sharing may prevent them running isolated from each other. To allow such transactions to continue to execute, without requiring that they run in the base space (where they could corrupt CICS data or programs) a single common subspace is provided in which all such transactions can execute. They are then isolated from the other transactions in the system that are running in their own subspaces, but able to share each other's data within the common subspace.

You might have some transactions whose application programs access each other's storage in a valid way. One such case is when a task waits on one or more event control blocks (ECBs) that are later posted, either by an MVS POST or 'hand posting', by another task. For example, a task can pass the address of a piece of its own storage to another task (via a temporary storage queue or some other method) and then WAIT for the other task to post an ECB to say that it has updated the storage. Clearly, if the original task is executing in a unique subspace, the posting task will fail when attempting the update and to post the ECB, unless the posting task is executing in CICS key. CICS therefore checks when a WAIT is issued that the ECB is in shared storage, and raises an INVREQ condition if it is not. Storage for the timer-event control area on WAIT EVENT, and storage for event control blocks (ECBs) specified on WAIT EXTERNAL and WAITCICS commands, must reside in shared storage[2] .

You can use the CICS Transaction Affinities Utility to identify those transactions whose programs issue WAIT EVENT, WAIT EXTERNAL, or WAITCICS commands, or MVS POST macros.

What next?

This chapter has briefly summarized the techniques and commands that can cause transaction affinity. Chapter 2, "Overview of the CICS Transaction Affinities Utility" on page 7 gives an overview of the CICS Transaction Affinities Utility and details of all the commands and command sequences that the CICS Transaction Affinities Utility looks for.

[2] Shared storage is allocated from one of the user-key shared dynamic storage areas, below or above the 16MB boundary (SDSA or ESDSA).

Chapter 2. Overview of the CICS Transaction Affinities Utility

This chapter

Gives an overview of the CICS Transaction Affinities Utility, and describes the basic components that make up this program offering.

The CICS Transaction Affinities Utility is designed to detect potential causes of inter-transaction affinity and transaction-system affinity for those users planning to use the CICS dynamic transaction routing facility. It can be used to detect programs that use EXEC CICS commands that may cause transaction affinity. It can also be used to create a file containing combined affinity transaction group definitions, suitable for input to the CICS system management product, CICSPlex SM. The commands that can be detected are listed in “Commands detected by the CICS Transaction Affinities Utility” on page 9. The CICS Transaction Affinities Utility is also of value for those users planning to use either asynchronous processing by CICS function shipping, or the transaction isolation facility of CICS/ESA 4.1.

The CICS Transaction Affinities Utility runs on the following releases of CICS:

- **CICS/ESA Version 4 Release 1**
- **CICS/ESA Version 3 Release 3**
- **CICS/ESA Version 3 Release 2.1**
- **CICS/MVS Version 2 Release 1.2**.

The CICS Transaction Affinities Utility has four main components:

1. Scanner (batch)
2. Detector (real-time)
3. Reporter (batch)
4. Builder (batch).

The CICS Transaction Affinities Utility determines the affinities that apply to a single CICS region; that is, a single pure AOR or single combined TOR/AOR. It can be run against production CICS regions, and is also useful in a test environment, to detect possible affinities introduced by new or changed application suites or packages.

APAR PN68267

Apar applied.

Important note

The CICS Transaction Affinities Utility is only an aid to help you find any affinities in your applications. You must relate the output from the CICS Transaction Affinities Utility to the applications that contain affinities before deciding whether or not the applications are suitable for CICS dynamic transaction routing.

To ensure that you detect as many potential affinities as possible, you should use the CICS Transaction Affinities Utility against all parts of your workload, including rarely-used transactions and out-of-normal situations.

Figure 2 shows the CICS Transaction Affinities Utility. Each of the four components are described in more detail in the following sections.

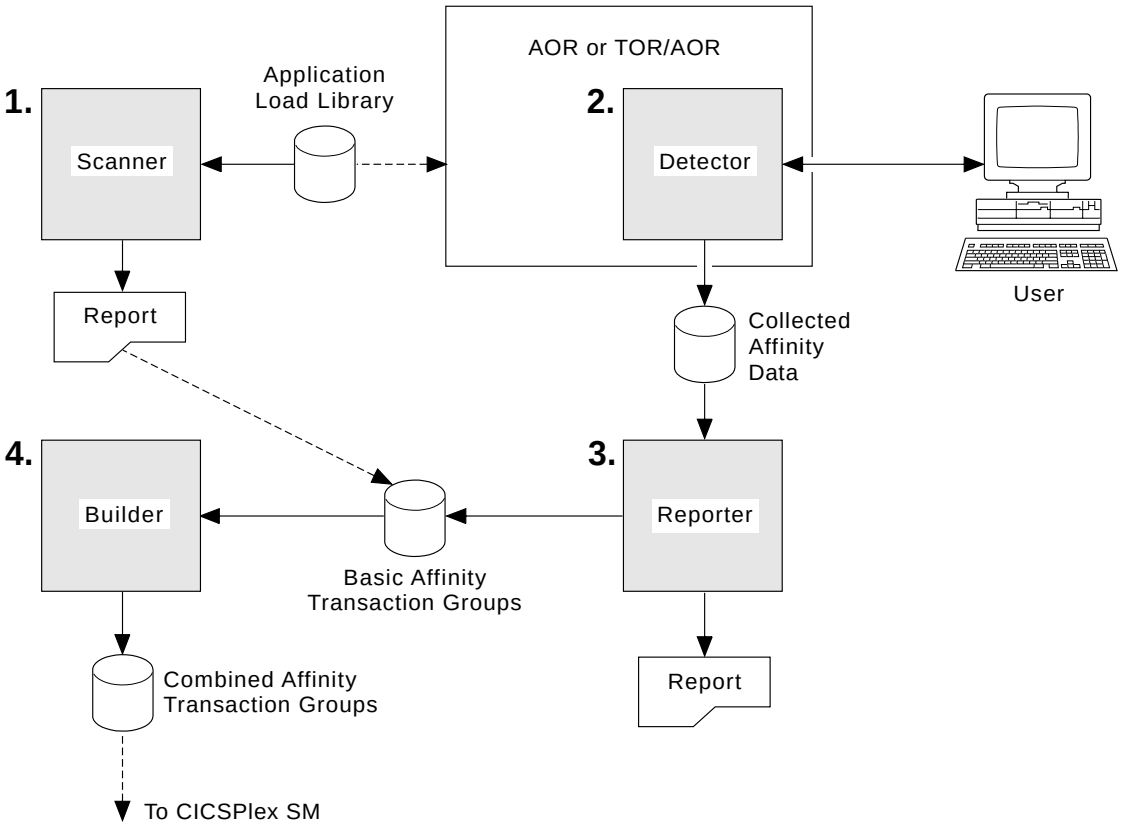


Figure 2. CICS Transaction Affinities Utility components

Commands detected by the CICS Transaction Affinities Utility

The CICS Transaction Affinities Utility can be used to detect instances of the EXEC CICS commands listed in Table 1.

Table 1. Commands detected by the CICS Transaction Affinities Utility

Inter-transaction affinity commands	Transaction-system affinity commands
ENQ	ADDRESS CSA (CICS/MVS 2.1.2 only)
DEQ	COLLECT STATISTICS
READQ TS	ENABLE PROGRAM
WRITEQ TS	DISABLE PROGRAM
DELETEQ TS	EXTRACT EXIT
ADDRESS CWA	INQUIRE
LOAD	SET
RELEASE	PERFORM
GETMAIN SHARED	RESYNC
FREEMAIN	DISCARD (CICS/ESA only)
RETRIEVE WAIT	WAIT EXTERNAL
DELAY	WAIT EVENT
POST	WAITCICS
START	
CANCEL	

Notes:

1. The Scanner may detect some instances of these commands that do not cause an affinity. For example, all FREEMAIN commands are detected but it is only those that are used to free GETMAIN SHARED storage that may cause affinity.
2. The Scanner also detects MVS POST SVC calls and MVS POST LINKAGE=SYSTEM non-SVC calls, because of their tie-up with the various EXEC CICS WAIT commands.
3. The CICS Transaction Affinities Utility does not search for transient data and file control EXEC CICS commands, as they are assumed not to cause affinity, because you can define transient data and file control resources as remote (in which case the request is function-shipped, causing no affinity problem).
4. The Detector ignores commands that target remote resources and are function shipped, because by function shipping the command there is no affinity problem.
5. The Scanner and Detector do not search for commands issued by any program named CAUxxxxx or DFHxxxxx, because CICS Transaction Affinities Utility and CICS programs are not considered part of the workload. Also, the Detector does not search for commands issued from:
   - DB2 and DBCTL task-related user exits
   - User-replaceable modules.
6. There are other ways that transactions can cause affinity with each other, but they are not readily detectable by the CICS Transaction Affinities Utility, because they do not take place via the EXEC CICS API.
7. The Detector lists WAIT commands as transaction-system affinities, because only half of the affinity can be detected. (The Detector does not detect MVS POST calls or hand posting of ECBs.)

Scanner

The Scanner is a batch utility that you can use to scan a load module library to detect those programs in the library that issue the EXEC CICS commands that may cause transaction affinity. It does this by examining the individual object programs looking for patterns that match the **argument zero**[3] format for the commands in question.

The Scanner detects the use of the EXEC CICS commands listed in Table 1 on page 9, and MVS POST requests.

The report produced by the Scanner only indicates that *potential* affinity problems may exist, as it only identifies the programs that issue the commands. It cannot obtain dynamic information about the transactions that use the programs or the names of the resources acted upon. You should use the report in conjunction with with the main report produced by the Reporter (see "Reporter" on page 17).

Notes:

1. The Scanner operation is independent of the language that the scanned program was written in and the release of CICS that the scanned program was translated under.
2. The Scanner may indicate an affinity problem that does not really exist, because the bit pattern found accidentally matches the argument 0 format for an affinity command.
3. The Scanner does not detect CICS macro-level commands.

Detector

The Detector can be used in real-time to detect transaction affinities in a running CICS region, and save details of the affinities in an MVS/ESA data space. This data is subsequently saved to DASD. The Detector consists of:

- A control transaction, CAFF
- An autosave transaction, CAFB
- Some global user exit programs
- A task-related user exit program.

This is shown in Figure 3 on page 11.

The data is collected by the global user exit programs at exit points XEIOUT, XMEOUT (or XTDCOUT for CICS/MVS 2.1.2), and XICEXP, and a task-related user exit at task start and task end. Between them, these exit programs intercept all the EXEC CICS commands and other events (pseudo-conversation end, terminal logoff, user signoff) that are needed to deduce the affinities and their relations and lifetimes. These exit programs coexist with any other exit programs at the same exit points. (They can be placed before or after other exit programs, without any of the exit programs being affected.)

[3] For an explanation of *argument zero*, see "Notes on Terminology" on page x.

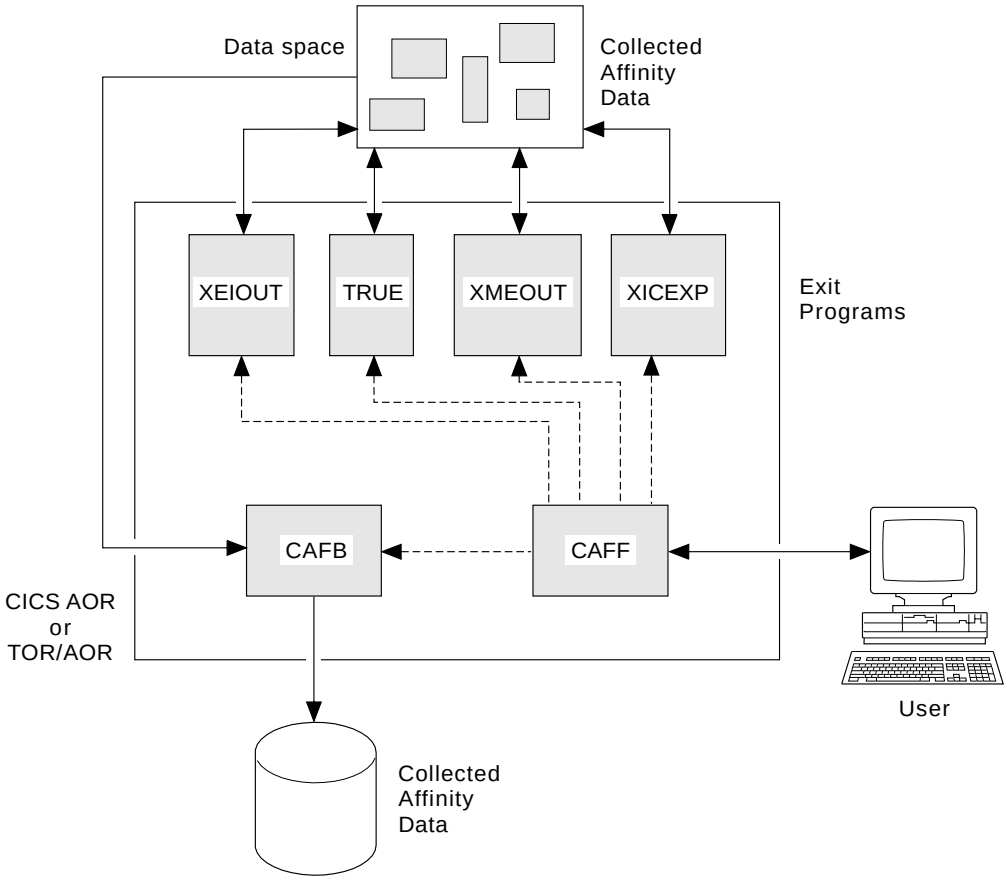


Figure 3. Detector components

You are recommended to run the Detector on stable CICS regions only; that is, maintenance should not be applied to application programs while the Detector is running. This is because maintenance of such programs may introduce or remove affinities, thus rendering collected data inaccurate.

What is detected

The Detector detects the EXEC CICS commands listed in Table 1 on page 9 that can cause transaction affinity. It also detects:

- The end of pseudo-conversations, by detecting when one of the transactions in the pseudo-conversation terminates without issuing an EXEC CICS RETURN TRANSID command with a non-zero transaction identifier. If a pseudo-conversation ends, and the resource shared by transactions that take part in the affinity still exists, then the lifetime of the affinity must be greater than PCONV.
- Logoffs and signoffs, by intercepting the CICS messages listed in Table 2 on page 12.

CICS release	Messages
CICS/ESA 4.1	DFHSN1200 Sign off at netname **portname** by userid **userid** is complete. DFHZC3462 **termid tranid** Node **netname** session terminated. DFHZC5966 **DELETE** started for terminal **termid**.
CICS/ESA V3	DFHSN0200 Signoff at terminal **termid** by userid **userid** is complete. DFHZC3462 **termid tranid** Node **netname** session terminated. DFHZC5966 **DELETE** started for terminal **termid**.
CICS/MVS 2.1.2	DFH3462 **termid tranid** Node **netname** session terminated. DFH5966 Deletion for terminal **termid** successful.

Table 2. Logoff and signoff messages intercepted

For more details about what is detected, see Appendix A, “Details of what is detected” on page 75.

Worsening of transaction affinities relations: In some cases, the Detector may not detect enough occurences (at least 10) of an affinity command to be sure that the affinity is definitely with a terminal (LUNAME) or userid (USERID). In such cases, the Detector records the (worsened) affinity relation as GLOBAL instead of LUNAME or USERID. Such relation worsening is flagged by the Detector, and reported by the Reporter.

Worsening of transaction affinities lifetimes: If a pseudo-conversation ends, and the resource still exists, the Detector deduces that the lifetime is longer than PCONV; that is, one of: LOGON, SIGNON, SYSTEM, or PERMANENT.

If a logoff or signoff occurs, and the resource still exists, the Detector deduces that the lifetime is longer than LOGON or SIGNON; that is, either SYSTEM or PERMANENT.

In some cases, the Detector may not detect a logoff or signoff, so cannot be sure that the affinity lifetime is LOGON or SIGNON. In such cases, the Detector records the (worsened) lifetime as SYSTEM or PERMANENT instead of LOGON or SIGNON. For example, this occurs when the CICS region on which the Detector is running is an AOR, because it is impossible in some cases to detect a logoff or signoff that occurs in a connected TOR. (The Detector cannot detect signoff events on CICS/MVS 2.1.2.)

Such lifetime worsening is flagged by the Detector, and reported by the Reporter.

What is not detected

The Detector does *not* detect EXEC CICS commands in the following situations:

- The Detector is not running
- The issuing program was translated with the SYSEIB option
- The command is an EXEC CICS FEPI command
- The command is not one that can cause transaction affinity
- The program name starts with CAU or DFH
- The program is a DB2 or DBCTL task-related user exit
- The program is a CICS user-replaceable module

- The transaction identifier does not match the prefix, if specified, for transactions to be detected
- The command is not one of the affinity types specified to be detected
- The command is function shipped to a remote CICS region
- The inter-transaction affinity commands are used within the same task.

The Detector does not detect CICS macro-level commands.

The Detector does not detect MVS POST calls or hand posting of ECBs.

If you continue a pseudo-conversation by setting a transid in the TIOA (rather than by using RETURN TRANSID), the Detector cannot detect PCONV lifetimes. In this case, the shortest lifetime detected is LOGON or SIGNON, because it interprets every transaction end as a pseudo-conversation end.

Ideally the CICS Transaction Affinities Utility should ignore commands issued by all task-related user exits and global user exits, because they are not part of applications. However, it cannot distinguish such commands from other commands, and does detect them. If your user exits use commands that can cause transaction affinities, the commands are detected, perhaps making any affinity problem seem worse than it actually is.

In CICS/ESA 4.1, if an exit program at XICEREQ or XTSEREQ modifies the EXEC CICS command then that modification is not visible to the Detector. (It detects the original, unmodified, command.) However, if an XICEREQ, XICEREQC, XEIIN, XTSEREQ, or XTSEREQC exit program (or an XEIOUT exit program invoked earlier) modifies EIBRESP, then the Detector sees the modified value.

APAR PN87388

Applied by Clare Jackson on 2/10/96

CICS Global user exit considerations

Modifications to EXEC CICS commands from exit programs running in XICEREQ
and XTSEREQ are not visible to the Detector because different copies of the
command-level parameter list are involved. . CICS provides a copy of the
command parameter list to the exit programs, addressed via UEPCLPS. This
allows them to modify certain parameters of the EXEC CICS command.

CICS has to provide a copy of the parameter list for such modifications, and not the
original version, since if the original were modified then all subsequent invocations
of it would reference the changed parameters. Also, the parameters to be modified
might be stored as read-only literals within program storage, and be unchangeable.

A program in XTSEREQ could add the SYSID parameter to an EXEC CICS
WRITEQ request, for example, by modifying the copy of the parameter list given to
XTSEREQ. This would result in the request being function-shipped to the remote
CICS system specified by the SYSID option, and not generating an affinity on the
local CICS system.

When the Detector analyses the request (from the XEIOUT global user exit point), it
is referencing the original parameter list, and not the modified copy. This means

that certain affinities may be detected that do not actually exist. The previous example of a temporary storage request being function-shipped due to a change made via XTSEREQ would be invisible to the Detector, and could lead to that request generating affinity information on the local CICS system.

You should take this into account when analyzing the data generated from running the Detector. For further information about the use of CICS global user exit programs, see the *CICS/ESA Customization Guide*.

Controlling the Detector

You can monitor and control the Detector through the CAFF transaction, which enables you to start, pause, continue, and stop the collection of affinity data into the tables in the data space. Using the CAFF transaction, you can also specify for which affinity commands, and for which transactions, data is to be collected.

The options that you specify to control the Detector for a CICS region are preserved in a recoverable VSAM control file. For more information about this file, see "The control record VSAM file" on page 16.

How the affinity data is collected

The Detector uses a number of affinity tables in the data space to hold collected affinity data. The affinity tables are in three categories:

1. There is an affinity table, or set of tables, for each of the following command groups that cause inter-transaction affinity:
   - ENQ and DEQ commands
   - READQ TS, WRITEQ TS, and DELETEQ TS commands
   - LOAD HOLD and RELEASE commands
   - RETRIEVE WAIT and START commands
   - ADDRESS CWA commands
   - GETMAIN SHARED and FREEMAIN commands
   - LOAD and FREEMAIN commands
   - CANCEL, DELAY, POST, and START commands.

   The tables for a particular group have a structure appropriate to that group.
2. There is an affinity table for each of the following command groups that cause transaction-system affinity:
   - INQUIRE, SET, ENABLE, DISABLE, EXTRACT, COLLECT STATS, PERFORM, DISCARD, ADDRESS CSA, and RESYNC commands.
   - WAITCICS, WAIT EVENT, and WAIT EXTERNAL commands.
3. There are two affinity tables that are used as aids to searching some of the other tables.

The affinity tables reside in the data space and are saved to the CICS Transaction Affinities Utility files when you stop the Detector, and optionally at predetermined intervals. (See "Saving affinity data" on page 15.)

Saving affinity data

The affinity data collected by the Detector is saved to the CICS Transaction Affinities Utility VSAM files by the autosave transaction, CAFB. For more information about these files, see “The affinity data VSAM files.”

The CAFB transaction saves affinity data automatically when you stop the Detector. You can also specify that the CAFB transaction save affinity data as follows:

- On a predetermined time/activity basis. That is, data is saved if either more than 300 seconds has passed, or more than 1000 table elements have changed, since the last save.
- When you pause the Detector.

Once the CAFB transaction has saved any data collected, it either becomes dormant until next activated (while the Detector is still running or paused) or terminates (if the Detector has been stopped).

Not all the affinity tables in the data space need to be saved, because some are temporary or are used only as an aid to searching. Further, some tables contain temporary elements, used for recording a possible affinity. Such elements are not saved to the files. They are either deleted when the CICS Transaction Affinities Utility deduces that there is actually no affinity, or are made permanent when it deduces that there really is affinity (in which case they get saved). Also, when data is saved, only those table elements that have been added or changed since the last save are written to the dataset. Timestamps in each table element indicate whether the element has been written already and whether it has changed since the last write. This minimizes the number of writes performed.

To improve performance, each affinity table is browsed and saved in its entirety, before considering the next table.

The affinity table elements are written in such an order that the data on the file is always consistent.

Note: If CICS or the Detector abends, the affinity data may be incomplete. Where possible, the Reporter detects this and issues a message to warn about possible incomplete data.

The affinity data VSAM files

The Detector uses three non-recoverable VSAM KSDS to hold saved affinity data. The files should be big enough to hold the maximum amount of affinity data that might be collected. Three are required because of the wide range of key lengths that the different tables have.

KSDS files are used because the Detector and Reporter need keyed access to the data.

The files are not recoverable because of the large amount of data that needs to be written. The data is written to the files in such a way that it remains consistent.

When the data contained in the tables is saved, each element in each table is a single file record. Records are therefore varying length. Each record has a prefix that contains a one-byte table identifier that identifies the affinity table the record belongs to. The table identifier acts as the first part of the record key. The second part of the key is the key of the table element itself.

Each file contains a header record. This enables both the Detector and Reporter to validate that the files they have been presented with are indeed CICS Transaction Affinities Utility affinity data files. The header record has a key in the same format as the rest of the keys on the file, so a table identifier of zero is used (no real table will have a table identifier of zero). The header record contains the CICS specific applid, thus allowing cross-validation of files.

The control record VSAM file

The CICS Transaction Affinities Utility control file is a recoverable VSAM KSDS file that holds a single control record. This record is used to preserve the Detector options and statistics, so that information is retained across Detector runs, transaction failures, and system failures and restarts. The record is created when the Detector transaction is first run on the CICS region, and is never deleted.

The control record holds the following information.

- CAFF options
- Detector statistics
- History information
  - Reason why STOPPED
  - Userid if STOPPED by user
  - Abend code if STOPPED by abend
  - Userid for last Detector options update
  - Date and time of last Detector options update
  - Specific applid of CICS system.

The record is updated whenever any of the above information changes. Such situations are when the Detector options change, the Detector statistics change, and the Detector state changes to STOPPED.

Note: The supplied definitions of the control record VSAM file makes it recoverable to CICS. If recovery is not required the definition may be changed (see “Defining CICS resources needed” on page 24). This must be done if CICS is being used with no system log, otherwise CAFF will abend AFCP.

Detector performance

The Detector is intended to be run against production CICS regions. However, over the period that the Detector is running, the CICS region suffers a performance degradation (dependent on the workload and number of affinities) equivalent to the performance impact of vendor monitor products that use the same user exits. The Detector is intended to be used over limited periods. To further limit the impact of running the Detector during periods of particularly heavy workloads, you can pause or stop the collection of data, or select specific commands or transactions to search for.

If your naming convention for TS queues uses a unique counter as part of the name, the Detector performance is likely to be worse than described above. This is because if every TS queue created has a unique name, the Detector creates a very large number of affinity records, which adversely affects performance.

The main cause of observed performance degradation is likely to be caused by the detection of TS commands. You may prefer to run the Detector twice; once for TS commands, and once for all other affinity commands.

CICS/MVS 2.1.2 limitations

The following limitations apply when running the Detector on a CICS/MVS 2.1.2 region.

- Signoff events are not detected; therefore, SIGNON lifetimes cannot be deduced.
- Macro-level affinities are not detected (for any CICS release).
- All programs that are the target of LOAD HOLD commands are assumed to be defined as RELOAD(NO); therefore, LOAD/FREEMAIN affinities for programs defined with RELOAD(YES) are not detected.
- You should not modify the original message at exit XTDCOUT, because this interferes with the detection of affinities.

Reporter

The Reporter is a batch utility that you can use to convert the affinity data collected by the Detector into two formats of output:

- A report presenting the affinity data in a readable form. This is intended for use by system programmers and application designers, and indicates the transactions and programs that issue the EXEC CICS commands that cause inter-transaction and transaction-system affinities. This information should give a better understanding of the transactions making up the workload, and should make it obvious what changes need to be made to remove unwanted affinities.
- A file containing a set of basic affinity transaction group definitions in a syntax approximating to the batch API of CICSPlex SM. This is intended as input to the Builder, which can merge these basic groups into the combined groups suitable for input to CICSPlex SM.

The input to the Reporter is the three files of affinity data from only one CICS region. The output from the Reporter therefore describes the affinities detected in a single CICS region.

Note: The output produced may not contain all the transaction affinities in the CICS region concerned, as some affinities may not have been found by the Detector. You may have to supplement the basic affinity transaction groups with information from the Scanner report, or from your own knowledge of your transactions, before using the file as input to the Builder.

Builder

The Builder is a batch utility that you can run against a set of files containing the basic affinity transaction group definitions as created by the Reporter. The Builder outputs a file containing combined affinity transaction group definitions, suitable for input to CICSPlex SM.

The basic groups are combined because of a CICSPlex SM rule that states that a given tranid may only appear in a single transaction group. It is quite possible that a tranid may appear in more than one basic group, and so these must be combined to form larger groups that satisfy CICSPlex SM.

Chapter 3. Installing the affinity utility

This chapter

Describes the steps to follow to install the CICS Transaction Affinities Utility on an MVS system.

The CICS Transaction Affinities Utility is distributed in SMP/E format on either one 9-track 6250bpi magnetic tape or one 3480 cartridge. The tape contains:

- The CICS Transaction Affinities Utility product code, for which the FMID is HQS9110 and the VOLSER is QS9110
- Five jobs, CAUINST1 to CAUINST5, for loading the CICS Transaction Affinities Utility from tape
- Two jobs to create the VSAM files used by the CICS Transaction Affinities Utility
- Two sample DCT source members for the transient data destination used for CICS Transaction Affinities Utility messages
- One sample FCT source member for the files used by the CICS Transaction Affinities Utility
- Four DFHCSDUP jobs for updating a CICS system definition (CSD) file with the resource definitions used by the Detector, and optionally to add those resource definitions to a resource group list to be specified in your CICS startup JCL
- Two jobs to run the Scanner
- A job to run the Reporter
- A job to run the Builder.

Check the program directory

The *Program Directory* is supplied in the distribution package. Read it carefully and use it with this book. If there are differences, follow the *Program Directory*, which contains the latest information.

Requirements for the CICS Transaction Affinities Utility

This section describes those things that you need to be able to use the CICS Transaction Affinities Utility.

- SMP/E 1.5

  To install or service the CICS Transaction Affinities Utility you need System Modification Program Extended (SMP/E) Version 1 Release 5 (5668-949) or later.

- CICS exit XEIOUT

  For CICS/ESA 4.1, CICS/ESA 3.3, and CICS/ESA 3.2.1 this exit is provided as part of the product.

  For CICS/MVS 2.1.2, this exit is provided by PTF UN43826.

- MVS/ESA 3.1.1 or later for the data spaces facility.

  A data space is used, for affinity data, by the Detector and Builder. The amount of storage required depends on the number of affinities discovered, and is a function of the number of different transaction identifiers, the number of terminals, and the number of userids used by the CICS region concerned. For information about calculating the likely data space storage requirement, see "Estimating the size of the MVS data space and files."

- To run the Detector against a CICS region running on MVS/ESA HBB3310 through HBB4430, you should have applied one of the following PTFs for APAR OY61443:

  Table 3. PTFs to fix APAR OY61443

MVS level	PTF
C10	UY91430
C11	UY91431
C13	UY91432
D10	UY91433
D20	UY91434
430	UY91435

  Note: You can still run the CICS Transaction Affinities Utility if you have not applied a PTF for APAR OY61443; see "Tailoring your CICS startup job" on page 26.

- CICS/ESA 4.1, CICS/ESA 3.3, CICS/ESA 3.2.1, or CICS/MVS 2.1.2.

  The CICS Transaction Affinities Utility runs only on CICS/ESA 4.1, CICS/ESA 3.3, CICS/ESA 3.2.1, or CICS/MVS 2.1.2 regions. To run the CICS Transaction Affinities Utility against a CICS/ESA 4.1 region, you must apply PTFs UN61942 and UN71013.

- About 29 cylinders of DASD storage on a 3390 device for the CICS Transaction Affinities Utility members.

  The amount of DASD storage needed for the affinity data files depends on: the number of CICS regions; and for each CICS region, the number of affinities discovered, the number of different transaction identifiers, the number of terminals used. (See "Estimating the size of the MVS data space and files.")

Estimating the size of the MVS data space and files

An MVS data space is used to hold the affinity data collected by the Detector. The amount of storage required depends on the number of affinities discovered, the number of different transaction identifiers, and the number of terminals.

To estimate the amount of storage for the data space (and therefore the size of the VSAM affinity files) that you are likely to need for the Detector, you can use the following algorithm (with storage values in bytes):

```
Dataspace : (#transids * #termids * 250) + 5000 000
CAUAFF1   : (#transids * #termids * 40) + 1 000 000
CAUAFF2   : (#transids * #termids * 150) + 1 000 000
CAUAFF3   : 1 000 000
```

where,

#transids is the number of transaction identifiers in the CICS region.

#termids is the number of terminal identifiers in the CICS region.

Note: The amount of storage needed in the data space for the Builder is about 25% of the storage needed for the Detector.

The algorithm assumes that all affinities are represented, and that all transactions participate in all affinities, and that all transactions run at all terminals (the worst possible scenario). This gives a worst case figure, in bytes.

For example, consider the worst case scenario of a CICS region with 500 different transaction IDs and 1000 terminals. where all transactions issue all affinity commands and all transactions run at all terminals. For this scenario, the storage requirement for the Detector in the data space is:

```
Dataspace : 130 Megabytes
CAUAFF1   : 21  Megabytes
CAUAFF2   : 76  Megabytes
CAUAFF3   : 1   Megabyte
```

The space required for the dataspace is different to that required for the files because:

- Each record has a storage overhead in the dataspace
- Certain tables are not saved to file
- Key length is fixed per file so short keys must be padded out.

Notes:

1. The critical affinity type is temporary storage. The space required for all other affinity types together should be no more than 5 Megabytes.
2. The calculations in this section assume that you do not use unique counters when naming temporary storage queues. If you do use unique counters, the space needed for temporary storage affinity types is much greater. For your calculations with unique counters, replace `#transids * #termids` by the number of unique queues.

Installing the CICS Transaction Affinities Utility

To install the CICS Transaction Affinities Utility, complete the following steps, as described in the following sections:

1. On CICS/MVS 2.1.2, apply PTF UN43826 to install the exits XEIOUT and XEIIN
2. Load the SMP/E installation jobs from tape
3. Tailor the installation jobs
4. Load the CICS Transaction Affinities Utility members from tape
5. Create the CICS Transaction Affinities Utility VSAM files
6. Define the transient data destination for CICS Transaction Affinities Utility messages in your CICS DCT
7. On CICS/ESA Version 3 and Version 4, define the CICS Transaction Affinities Utility transactions, programs, and files in your CSD

8. On CICS/MVS 2.1.2, define the CICS Transaction Affinities Utility transactions and programs in your CSD, and define the CICS Transaction Affinities Utility files in your FCT
9. Tailor your CICS startup job
10. Re-ipl CICS.

Installing exit XEIOUT on CICS/MVS 2.1.2

The CICS Transaction Affinities Utility uses the exit XEIOUT from CICS/ESA Version 3. To enable the CICS Transaction Affinities Utility to work with CICS/MVS 2.1.2, this exit is provided by PTF UN43826. You must apply PTF UN43826 to all CICS/MVS 2.1.2 regions that you intend using the CICS Transaction Affinities Utility on.

Loading the contents of the distribution tape

To load the contents of the distribution tape to DASD:

1. Edit and submit the job in Figure 4, to load file 3 (RELFILE (2)) from the tape into the installation library CICSAFF.XCAUINST. File 3 contains the jobs listed in Table 4 on page 23.

```
//READTAPE JOB etc
//COPYFILE EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
#//IN       DD UNIT=3480,VOL=SER=QS9110,LABEL=(3,SL),
#// DSN=HQS9110.F2,DISP=OLD
#//OUT      DD DSN=CICSAFF.XCAUINST,DISP=(NEW,CATLG),
#// UNIT=SYSDA,VOL=SER=vvvvvv,
// SPACE=(6160,(110,11,2)),
// DCB=(LRECL=80,BLKSIZE=6160,RECFM=FB)
//SYSIN    DD *
  COPY INDD=IN,OUTDD=OUT
/*
```

Figure 4. Sample job to load CICS Transaction Affinities Utility file 3 from tape

HQS9110 is the feature number (FMID) of the Transaction Affinities Utility
tape.

QS9110 is the volume identifier of the Transaction Affinities Utility tape.

Change the following to suit your environment:

vvvvvv The volume identifier of the DASD unit on which you want to load file 3 from the tape.

CICSAFF.XCAUINST
The name of the CICS Transaction Affinities Utility installation library into which you will load file 3 from the tape.

Note: You can later edit the jobs in this library, and run them to load the CICS Transaction Affinities Utility software from the tape, and to tailor the CICS Transaction Affinities Utility environment. If you later apply any service to the copies of these jobs loaded by

SMP/E into the CICSAFF.ACAUINST and CICSAFF.SCAUINST libraries, you should reflect the service in your tailored jobs in CICSAFF.XCAUINST.

Table 4. Members loaded from file 3 of the CICS Transaction Affinities Utility tape

Member	Description
CAUINST1	Installation job 1 – to create the target and distribution libraries, ACAUINST, ACAUMOD, SCAUINST, and SCAULOAD
CAUINST2	Installation job 2 – to create the SMP/E datasets
CAUINST3	Installation job 3 – to initialize the SMP/E datasets
CAUINST4	Installation job 4 – to invoke SMP/E RECEIVE
CAUINST5	Installation job 5 – to invoke SMP/E APPLY and ACCEPT
CAUJCLCA	Job to create VSAM affinity data files
CAUJCLCC	Job to create VSAM control files
CAUJ41CR	Job to define resources in CSD (CICS/ESA 4.1)
CAUJ33CR	Job to define resources in CSD (CICS/ESA 3.3)
CAUJ32CR	Job to define resources in CSD (CICS/ESA 3.2.1)
CAUJ21CR	Job to define non-file resources in CSD (CICS/MVS 2.1.2)
CAUFCT21	Sample file entries for FCT (CICS/MVS 2.1.2)
CAUJCLLS	Job to run the Scanner (Summary report.)
CAUJCLLD	Job to run the Scanner (Detail report.)
CAUJCLRP	Job to run the Reporter
CAUJCLBL	Job to run the Builder
CAUDCTAF	Sample source for CICS Transaction Affinities Utility DCT entry (CICS/ESA V3 and V4)
CAUDCT21	Sample source for CICS Transaction Affinities Utility DCT entry (CICS/MVS 2.1.2)

2. Tailor the installation jobs, CAUINST1 to CAUINST5, installed in CICSAFF.XCAUINST file to suit your requirements. The following values are assumed in those jobs; change them if required. (The jobs have notes within them to help you tailor them.)
   - The target volume is CICAFF.
   - The high-level qualifier of all datasets is CICSAFF.
   - The DASD unit type is 3380.
   - In CAUINST4, the tape unit type is 3480, for unloading from a tape cartridge. If you are unloading from a 6250 tape, change this to 6250.
3. Run the job in member CAUINST1 to create the target and distribution libraries.
4. Run the job in member CAUINST2 to create the SMP/E datasets.
5. Run the job in member CAUINST3 to initialize the SMP/E datasets.
6. Run the job in member CAUINST4 to invoke SMP/E RECEIVE.
7. Run the job in member CAUINST5 to invoke SMP/E APPLY and ACCEPT.

Defining the CICS Transaction Affinities Utility transient data destination

To define the transient data destination to be used for messages that can be issued by the CICS Transaction Affinities Utility, add the sample entry provided into your CICS DCT. The sample DCT entry is installed in the following members of the CICSAFF.XCAUINST library:

- CAUDCTAF (for CICS/ESA Version 3 and Version 4)
- CAUDCT21 (for CICS/MVS 2.1.2)

Creating the VSAM files

The CICS Transaction Affinities Utility uses one copy of each of the following VSAM files for each CICS region it is run against:

Table 5. CICS Transaction Affinities Utility VSAM files and associated jobs

File	Description	Job
CICSAFF.CAUCNTL	A recoverable file used to hold control information.	CAUJCLCC
CICSAFF.CAUAFF1 CICSAFF.CAUAFF2 CICSAFF.CAUAFF3	Non-recoverable files used to hold affinity data with different key sizes (9, 25, and 225 respectively).	CAUJCLCA

To create a set of these files for ***one*** CICS region, edit and run a copy of the associated jobs in the CICSAFF.XCAUINST library.

Note: You must edit and run a copy of the CAUJCLCC and CAUJCLCA jobs for each CICS region against which you are going to use the CICS Transaction Affinities Utility.

Before you run the CAUJCLCC and CAUJCLCA jobs, change the following parameters in the jobs:

1. The JOB accounting parameters
2. The prefix of the files; default CICSAFF.xxxxxx, where xxxxxx is a CICS region qualifier
3. The volume id of the DASD device where the files are to reside; default vvvvvv.

Defining CICS resources needed

To define the CICS files, mapsets, programs, and transactions used by the CICS Transaction Affinities Utility, edit and submit the appropriate sample job, CAUJnnCR, for the release of CICS against which you are going to run the CICS Transaction Affinities Utility. (To define the files for CICS/MVS 2.1.2 you must add the entries from member CAUFCT21 to your FCT.)

The following sample DFHCSDUP jobs are installed in the CICSAFF.XCAUINST library:

Job name	Define resources in CSD groups
CAUJ41CR	AF41GRP and AF41FILE (for CICS/ESA 4.1)
CAUJ33CR	AF33GRP and AF33FILE (for CICS/ESA 3.3)
CAUJ32CR	AF32GRP and AF32FILE (for CICS/ESA 3.2.1)
CAUJ21CR	AF21GRP (for CICS/MVS 2.1.2)

Each CAUJnnCR job adds the resource definitions for the associated release of CICS to the associated resource groups, AFnnGRP and AFnnFILE, in your CICS system definition (CSD) file.

Notes:

1. If you intend running the CICS Transaction Affinities Utility against several CICS regions, and they use separate CSDs, edit and run the appropriate CAUJnnCR job for each CSD.

Before you run the CAUJnnCR job, change the following parameters in the job

1. The JOB accounting parameters.
2. The name of the CSD file; default CICSrel.xxxxxx.DFHCSD, where *rel* is 410, 330, 321, or 212, as appropriate.
3. The prefix of the CICS load library; default CICSrel.SDFHLOAD, where *rel* is 410, 330, 321, or 212, as appropriate.
4. For CICS/ESA Version 3 and Version 4:
   a. The prefixes of the VSAM files, as defined by the CAUJCLCC and CAUJCLCA jobs
   b. The LSRPOOL options; ensure that the LSRPOOLID specified for each file is capable of handling the keylength defined for the file (see Table 5 on page 24).
   c. For file CAUCNTL, if recovery is not required, ensure RECOVERY(NONE) and FWDRECOVLOG(NO) is specified.
5. If you want to add the CICS Transaction Affinities Utility resource group to the group list specified in your CICS startup JCL:
   a. Activate the commands ADD GROUP(AFnnGRP) TO(yyyyyy) and ADD GROUP(AFnnFILE) TO(yyyyyy) by removing the preceding asterisk (*)
   b. Change *yyyyyy* to the name of the group list specified in your CICS startup JCL.

Note: When you first run the CAUJnnCR jobs, they return a code RC=4, because they attempt to delete CSD groups that do not yet exist. You should ignore this return code when you first run the jobs.

To define the VSAM files to be used for CICS/MVS 2.1.2, add the sample entries provided into your CICS FCT. The sample FCT entry is installed in member CAUFCT21 of the CICSAFF.XCAUINST library. Before you add the sample file entries to your FCT, change the following parameters in the CAUFCT21 member:

1. The prefixes of the VSAM files, as defined by the CAUJCLCC and CAUJCLCA jobs
2. The LSRPOOL options; ensure that the LSRPOOLID specified for each file is capable of handling the keylength defined for the file (see Table 5 on page 24).

3. For file CAUCNTL, if recovery is not required, ensure LOG=NO is specified.

Installing the resource group on your CICS

For each CICS region that you intend running the CICS Transaction Affinities Utility against, you must make available the resource definitions needed. You can do this by either of the following methods:

- Add the CICS Transaction Affinities Utility resource group to the group list specified in the startup JCL for your CICS region (for example, when you ran the CAUJnnCR job).
- Use the CEDA transaction to install the CICS Transaction Affinities Utility resource group after you have restarted your CICS region with CICS Transaction Affinities Utility support. This is described at an appropriate stage later in this document.

Tailoring your CICS startup job

To enable the CICS Transaction Affinities Utility to be run against your CICS region, you must specifying the following in the CICS startup job:

- The CICS Transaction Affinities Utility load library, CICSAFF.SCAULOAD in the DFHRPL concatenation:

  ```
  //          DD DSN=CICSAFF.SCAULOAD,DISP=SHR
  ```

- The CICS Transaction Affinities Utility transient data destination on the DD statement:

  ```
  //CAFF DD SYSOUT=*
  ```

- The DCT=xx system initialization parameter, to specify the suffix of the DCT that contains the entry for the CICS Transaction Affinities Utility transient data destination. (See “Defining the CICS Transaction Affinities Utility transient data destination” on page 24.)
- The FCT=xx system initialization parameter, to specify the suffix of the FCT that contains the file entries for CICS/MVS 2.1.2.
- If the CICS region is running on either of the following types of MVS/ESA:
  - MVS/ESA HBB3310 through HBB4430 and you ***have*** applied a PTF for MVS APAR OY61443
  - Another release of MVS/ESA

  specify the ICVR system initialization parameter as at least 10 seconds; that is, ICVR=10000 (or a larger value).

  If the CICS region is running on MVS/ESA HBB3310 through HBB4430 and you ***have not*** applied a PTF for MVS APAR OY61443 specify the ICVR=0 system initialization parameter, to disable the timeout of runaway transactions.

  Notes:

  1. Failure to do this may result in AICA abends of the Detector or your own transactions.
  2. For a list of the PTFs for APAR OY61443, see Table 3 on page 20.
- If the CICS region is running at CICS/MVS 2.1.2, you must specify the EXEC=YES and EXITS=YES system initialization parameters.

- If the CICS region is running at CICS/ESA 4.1, you should specify the DSHIPINT=0 system initialization parameter. (If you specify a DSHIPINT value other than 0, terminal logoffs may be incorrectly observed, and the affinity lifetimes not be determined correctly.)

If the CICS region is an AOR, you must specify the AILDELAY=0 system initialization parameter on ***all TORs*** that route to it. This enables the Detector to see a terminal logoff immediately that it occurs. (If you specify an AILDELAY value other than 0, logoffs may be missed and the affinity lifetimes not be determined correctly.)

Restarting your CICS region

To enable the CICS Transaction Affinities Utility run-time programs, you must restart the CICS region that you intend running the CICS Transaction Affinities Utility against, using a CICS startup job modified for CICS Transaction Affinities Utility support, as described in “Tailoring your CICS startup job” on page 26.

Chapter 4. Running the Scanner

This chapter

Describes how to run the Scanner, which scans load modules for instances of API commands that could cause inter-transaction affinity and transaction-system affinity.

You can run the Scanner to produce either a summary report or a detailed report of modules that contain possible affinity-causing EXEC CICS commands or MVS POST calls.

The recommended way to use the Scanner is to:

1. Produce a summary report and module list to identify suspect modules
2. Produce detailed reports to review modules that the summary report flagged as suspect.

Creating a summary report

You can request a summary report from the Scanner by editing and running the job, CAUJCLLS. This job can also output a list of modules with potential transaction affinities, for input to the CAUJCLLD job for more detailed reporting.

Before running the CAUJCLLS job, change the following as appropriate:

- The JOB accounting parameters.
- The PARM statement:

  ```
  PARM='$SUMMARY[,DETAILMODS]'
  ```

  $SUMMARY
  Specifies that a summary scan (and report) is required for the entire library, except for CICS modules, CICS tables, and those modules that cannot be loaded (due to some error).

  DETAILMODS
  Specifies that the names of those modules that contain at least one possible affinity-causing EXEC CICS command or MVS POST are to be written to the sequential file defined by the AFFMOD DD statement. This file may be used to restrict a subsequent detailed report, by specifying it on the DETAIL DD statement of a detailed report run of the Scanner.

- The STEPLIB DD statement

  Specify the name of the CICS Transaction Affinities Utility load library in which you have installed the Scanner program, CAULMS; default, CICSAFF.SCAULOAD.

- The INPUT DD statement

  Specify the name of the load library to be scanned.

- The SYSPRINT DD statement

  Specify the destination for the summary report.

- The AFFMOD DD statement

  Specify the name of the sequential data set to which the list of modules with potential transaction affinities is to be sent. You can edit the data set to alter the list of modules to be scanned before running the Scanner to produce a detailed report.

- The DETAIL DD statement (dummy)

  Not needed for a summary run.

Each summary report contains:

- A separate line giving the following information about each module in the library:
  - Name
  - Size
  - Language (if determined)
  - Number of possible affinity-causing EXEC CICS commands
  - Number of possible MVS POST commands.

  If a module seems to contain affinity-causing EXEC CICS commands, it is flagged with the message “Possible affinity commands.”. If a module seems to contain MVS POST commands, it is flagged with the message “Possible MVS POSTs.”.

  Note: The language is determined only if at least one affinity-causing EXEC CICS command is detected, and is derived from the EXEC argument zero[4] of the first such command. Therefore, if a load module is created from several source languages, only one language is indicated.

- Library-totals of:
  - Modules in the library
  - Modules scanned
  - CICS modules and tables (not scanned)
  - Modules in error (not scanned)
  - Modules that possibly contain MVS POST commands
  - Modules that possibly contain affinity-causing EXEC CICS commands
  - Assembler modules
  - C/370 modules[5]
  - OS/VS COBOL modules
  - VS COBOL II modules[5]
  - PL/I modules[5].

Figure 5 on page 31 is an example of a summary report produced by the Scanner.

[4] For an explanation of *argument zero*, see “Notes on Terminology” on page x.

[5] This includes programs compiled by a Language Environment/370-enabled compiler.

```
CICS TRANSACTION AFFINITIES UTILITY                                                    1993/12/01  Page   1
LOAD MODULE SCANNER - SUMMARY LISTING OF C330TEST.LOAD

Module     Module     Module      Affinity     MVS POSTs   Comment
Name       Length     Language    Statements
--------   --------   ---------   ----------   ---------   ------------------------------------------------------
ACSA1      00000198   ASSEMBLER          3            0    Possible affinity commands.
AFFYIC     00000308                      0            0
AFFYTS     00000628                      0            0
COBACSA    000008B8   COBOL II           4            0    Possible affinity commands.
CRMRPROG   000001A0                      0            0
DFHSRT1$               CICS TABLE        0            0
DFHTCTSH               CICS TABLE        0            0
DFHTSTEC               CICS TABLE        0            0
PLIACSA    000003C8   PL/I               3            0    Possible affinity commands.
PLTCCC     00000D48   C/370              2            0    Possible affinity commands.
PLTCOB     000009D8   COBOL II           2            0    Possible affinity commands.
PLTPLI     00000570   PL/I               1            0    Possible affinity commands.
SCRATCH    000006D0                      0            0
TRANOUT    000008B8                      0            1    Possible MVS POSTs.


CICS TRANSACTION AFFINITIES UTILITY                                                    1993/12/01  Page   2
LOAD MODULE SCANNER - SUMMARY LISTING OF C330TEST.LOAD

               LOAD LIBRARY STATISTICS
=======================================================
Total modules in library                          =   14
Total modules scanned                             =   11
Total CICS modules/tables (not scanned)           =    3
Total modules in error (not scanned)              =    0
Total modules containing possible MVS POSTs       =    1
Total modules containing possible Affinity commands =  6
  Total ASSEMBLER modules                         =    1
  Total C/370 modules                             =    1
  Total COBOL modules                             =    0
  Total COBOL II modules                          =    2
  Total PL/I modules                              =    2
```

Figure 5. Example of a summary report produced by the Scanner

Creating a detailed report

You can request a detailed report from the Scanner by editing and running the job, CAUJCLLD.

Change the following as appropriate:

- The PARM statement:

  ```
  PARM='$DETAIL[,ALL]'
  ```

  $DETAIL
  Specifies that a detailed scan (and report) is required. The extent of the scan is defined by the either the ALL parameter or the DETAIL DD statement.

  ALL
  Specifies that all modules in the load library are to be scanned for possible affinity-causing EXEC CICS commands and MVS POST commands.

If ALL is omitted, only those modules listed in the file specified on the DETAIL DD statement are to be scanned. This file would normally be from the AFFMOD DD output of an Scanner summary report run, which you can edit before creating a detailed report.

- The STEPLIB DD statement

  Specify the name of the CICS Transaction Affinities Utility load library in which you have installed the Scanner program, CAULMS; default, CICSAFF.SCAULOAD.

- The INPUT DD statement

  Specify the name of the load library to be scanned.

- The SYSPRINT DD statement

  Specify the destination for the detailed report.

- The AFFMOD DD dummy statement

  Not needed for a detailed run.

- The DETAIL DD statement

  Specify the name of the data set containing the list of modules to be scanned. This list may be created initially as the output from a summary run of the Scanner. If you specify ALL on the PARM statement, change the DETAIL DD statement to specify //DETAIL DD DUMMY.

Each detailed report contains:

- A section for each module, that contains:
  - A header line giving the name, size, and entry point of the module.
  - A line for each possible affinity-causing command found, giving:
    - The offset of the command argument zero declaration from the start of the load module.

      Note: This offset is not the same as the offset given by the Reporter; the offset given by the Reporter is for the command itself. (See page 49 and page 81.)

    - The contents of the command argument zero declaration (in hexadecimal).
    - The EDF DEBUG line number, if present. This can provide a useful clue for identifying false affinities. If a section of a load module was translated with the DEBUG option, then EDF DEBUG line numbers are given. For such a module, if no DEBUG line number is given, this may indicate that what was found was not an argument zero.
    - What the command appears to be (for example, WRITEQ TS).
    - Whether the affinity is inter-transaction (Trans), transaction-system (System), or both[6] (Both).

[6] An affinity command can be both inter-transaction and transaction-system only if an ADDRESS command has both CWA and CSA keywords.

– A line for each possible MVS POST command found, giving:
  - The offset of the MVS POST command from the start of the load module
  - The MVS POST instruction
  - The 12 bytes of storage immediately before, and the 10 bytes after, the MVS POST command
  - Whether the MVS POST is SVC or otherwise.
– A summary report of the modules, giving:
  - The total possible affinity commands
  - The total possible MVS post commands.
– Library totals, as for the summary report, but for only those modules selected for the detailed run.

Figure 6 is an example of a detailed report produced by the Scanner.

```
CICS TRANSACTION AFFINITIES UTILITY                                                    1993/12/01  Page     1
LOAD MODULE SCANNER - DETAILED LISTING OF C330TEST.LOAD

Module Name - ACSA1     /  Load Module Length - 00000198  /  Module Entry Point - 00000028
Offset     Storage Content (HEX)                            EDF DEBUG  Possible Command                   Affinity
--------   -----------------------------------------------  ---------  --------------------------------  --------
00000360   0A02E0004900004900                               00000022   WRITEQ TS                          Trans
00000400   0A04E8004900008902                               00000028   READQ TS                           Trans
00000482   0A06E8004900002102                               00000036   DELETEQ TS                         Trans
Total possible Affinity commands =         3
Total possible MVS POSTs         =         0

Module Name - TRANOUT   /  Load Module Length - 000008B8  /  Module Entry Point - 00000028
Offset     Storage Content (HEX)                            EDF DEBUG  Possible Command                   Affinity
--------   -----------------------------------------------  ---------  --------------------------------  --------
00000534   4100411031341B00411010000A021B0041104000A021B00              MVS POST (SVC)                     System
Total possible Affinity commands =         0
Total possible MVS POSTs         =         1


CICS TRANSACTION AFFINITIES UTILITY                                                    1993/12/01  Page     2
LOAD MODULE SCANNER - DETAILED LISTING OF C330TEST.LOAD

               LOAD LIBRARY STATISTICS
============================================================
Total modules in library                          =    2
Total modules scanned                             =    2
Total CICS modules/tables (not scanned)           =    0
Total modules in error (not scanned)              =    0
Total modules containing possible MVS POSTs       =    1
Total modules containing possible Affinity commands =    1
  Total ASSEMBLER modules                         =    1
  Total C/370 modules                             =    0
  Total COBOL modules                             =    0
  Total COBOL II modules                          =    0
  Total PL/I modules                              =    0
```

Figure 6. Example of a detailed report produced by the Scanner

Chapter 5. Running the Detector

This chapter

Describes how to run the Detector, which runs in a CICS region looking for instances of API commands that could cause transaction affinity. The commands looked for are those listed in “Commands detected by the CICS Transaction Affinities Utility” on page 9.

You can run the Detector either at a CICS 3270-type terminal (through interactive screens or single-line commands), from a console, or from an application program. This chapter primarily describes how to use the Detector through the interactive screens at a CICS terminal, but also gives equivalent commands to use at a terminal, at a console, or in an application program.

For an overview of the Detector, see “Detector” on page 10.

You can control the Detector by:

- Changing the state

  This is described in topics “Starting the collection of affinity data” on page 38 through “Stopping the collection of affinity data” on page 40.

- Changing the options

  The options that you can select are shown in Figure 8 on page 41, and described in “Changing the Detector options” on page 40.

You can optimize the performance of the Detector by the manner in which you use it; that is, you should consider doing the following:

- Pausing the collection of data during peak workloads. You can continue the collection of data when the workloads have decreased.
- Collect data for one affinity type at a time. (This can be of particular value for the temporary storage affinity type.)
- Collect data for a restricted set of transaction identifiers, by specifying the prefix of those transactions that you are interested in.

For information about how to specify the affinity types and transaction identifier prefix for which data is to be collected, see “Changing the Detector options” on page 40.

CICS/MVS 2.1.2 considerations

You should note the following considerations when running the Detector on a CICS/MVS 2.1.2 region:

- If the user of the CAFF transaction is not signed on to the CICS/MVS 2.1.2 region, the userid on the CAFF01 and CAFF02 panels and in messages is displayed as ????????.
- If an abend occurs, the program causing the abend is shown as ???????? in messages.

- If CICS shuts down normally and the Detector is not stopped, CICS hangs. You must purge the CAFB transaction to clear the hang.

Displaying the Detector control screen

To display the control screen that you can use to run the Detector at a CICS terminal, first type the transaction identifier CAFF then press Enter. In response, the Detector control screen, CAFF01 (shown in Figure 7) is displayed. You can use this screen to review and change the state of the Detector, or to display the Detector options screen, CAFF02 (shown in Figure 8 on page 41).

To refresh the values displayed on the screen at any time, press Enter.

To exit the Detector control screen, press the F3 (or F12) function key. This does not affect the state of the Detector

```
CAFF01        CICS Transaction Affinities Utility            Applid CICS330

  Press Start key (F5) to start detection.  1
  Press Options key (F4) to modify the CAFF operation options.
  Press Enter to update statistics.

  CAFF state . . . . . : STOPPED by user <userid>  2
  Number of pauses . . . . . : 0     3
  Number of saves. . . . . : 6     3
  Records written last save. : 257   4
  Total records on file. . . : 834   5

  Date/time of last start. . : 06/30/93 09:05:23 (MM/DD/YY HH:MM:SS)  6
  Date/time of last save . . : 06/30/93 11:13:12 (MM/DD/YY HH:MM:SS)
  Date/time of last change . : 06/30/93 11:12:34 (MM/DD/YY HH:MM:SS)

  Total time RUNNING . . . . : 0002:08:12  (HHHH:MM:SS)   7
  Total time PAUSED. . . . . : 0000:00:00  (HHHH:MM:SS)

  Table dataspace name . . . :                 % full  8


  9
F1=Help F3=Exit F4=Options F5=Start F6=Stop F7=Pause F8=Continue F12=Cancel  10
```

Figure 7. Detector control screen, CAFF01

The Detector control screen, CAFF01, shows the following:

1 The functions that you can select from this state of the Detector. For any state of the Detector, only appropriate functions are displayed.

2 The current state of the Detector. If the state is STOPPED, the reason why it was stopped is displayed.

Notes:

1. When you stop the Detector, the CAFF state changes to STOPPED only after the Detector has saved the affinity data.
2. When you pause the Detector, the CAFF state changes to PAUSED *before* the Detector saves the affinity data (to ensure that the Detector pauses immediately). After the state has changed to PAUSED, you can refresh the data displayed by pressing Enter.

3 The number of times that the Detector has been paused, and the number of times data has been saved, since the last time that it was started.

4 The number of new records and updates to existing records written to the
affinity data VSAM files when data was last saved.

5 The total number of affinity records in the affinity data VSAM files. If the Detector was stopped by CICS crashing, and was in the middle of saving affinity data, this figure may be inaccurate. However, the figure is corrected the next time the Detector is started.

6 The date and time of when the Detector was last started, data was saved, and a change was made to an affinity table. The date is given in the format specified by the DATFORM system initialization parameter.

7 The total time that the Detector has been in the running and paused states since it was last started.

8 The name, and current percentage occupied, of the MVS dataspace being used. (Not displayed while in STOPPED state.)

9 The message line used to display diagnostic messages. When the CAFF transaction is first entered, this line displays the copyright notice:

```
5696-582(C) Copyright IBM Corp. 1993.
```

10 The keys that you can use to select functions to affect the operation of the Detector, or to get help information about it. This line displays all possible functions; not all of which are appropriate (or selectable) for a given state of the Detector.

Starting the collection of affinity data

When you can start affinity data collection

You can start the collection of affinity data only when the Detector is currently stopped.

To start the collection of affinity data, use one of the methods shown in Table 6.

Table 6. Methods for starting data collection by the Detector

Where used	Command or function key
Control display, CAFF01	F5 function key 1
3270 terminal	CAFF START
Console	F cicsjob, CAFF START 2
Application program	EXEC CICS START TRANSID('CAFF') FROM('START') 3

Notes:

1 If you press the F5 function key of the CAFF01 screen, you are asked to confirm that you want to start the recording of affinity data.

2 cicsjob is the name of your CICS startup job.

3 You can use this command from a program initiated during the third stage of CICS initialization; that is, a program specified in the second part of the PLTPI list for the CICS region. For more information about using PLTPI programs, see the *CICS/ESA Customization Guide*.

This causes the Detector to record transaction affinities in the CICS region, until you pause or stop the collection of data by the Detector. Data is collected for only the affinity types that you have selected to be detected (by specifying Y for the affinity type on the CAFF02 screen).

Each time the Detector is started, a new data space is created. For information about calculating the likely data space storage requirement, see “Estimating the size of the MVS data space and files” on page 20. You specify this size on the Detector options screen, CAFF02. On the CAFF02 options screen you can also specify that data from affinity data VSAM files (for example, from previous CICS Transaction Affinities Utility runs) is to be loaded into the data space when it is created. For more information about the CAFF02 options screen, see “Changing the Detector options” on page 40.

Note: If there are a large number of data records to be loaded into the data space when it is created (for example, from previous CICS Transaction Affinities Utility runs), the CAFF screen may be frozen for some appreciable time, until the records have been loaded.

Pausing the collection of affinity data

When you can pause affinity data collection

You can pause the collection of affinity data only when the Detector is currently running.

To pause the collection of affinity data, use one of the methods shown in Table 7.

Table 7. Methods for pausing data collection by the Detector

Where used	Command or function key
Control display, CAFF01	F7 function key
3270 terminal	CAFF PAUSE
Console	F cicsjob, CAFF PAUSE
Application program	EXEC CICS START TRANSID('CAFF') FROM('PAUSE')

Note: cicsjob is the name of your CICS startup job.

This causes the Detector to temporarily stop recording any transaction affinities in the CICS region, until you resume the collection of data by the Detector. The data already collected remains in the data space, and can be saved to the affinity data VSAM files when the Detector is paused. (You can specify to save data when the Detector is paused on the CAFF02 options screen, as described in “Changing the Detector options” on page 40.)

Resuming the collection of affinity data

When you can resume the affinity data collection

You can resume the collection of affinity data only when the Detector is currently paused.

To resume the collection of affinity data, use one of the methods shown in Table 8.

Table 8. Methods for resuming data collection by the Detector

Where used	Command or function key
Control display, CAFF01	F8 function key
3270 terminal	CAFF CONTINUE
Console	F cicsjob, CAFF CONTINUE
Application program	EXEC CICS START TRANSID('CAFF') FROM('CONTINUE')

Note: cicsjob is the name of your CICS startup job.

This causes the Detector to continue recording any transaction affinities in the CICS region, until you pause or stop the collection of data by the Detector.

Stopping the collection of affinity data

When you can stop affinity data collection

You can stop the collection of affinity data only when the Detector is currently running or paused.

To stop the collection of affinity data, use one of the methods shown in Table 9.

Table 9. Methods for stopping data collection by the Detector

Where used	Command or function key
Control display, CAFF01	F6 function key 1
3270 terminal	CAFF STOP
Console	F cicsjob, CAFF STOP 2
Application program	EXEC CICS START TRANSID('CAFF') FROM('STOP') 3

Notes:

1 If you press the F6 function key of the CAFF01 screen, you are asked to confirm that you want to stop the recording of affinity data.

2 cicsjob is the name of your CICS startup job.

3 You can use this command from a program initiated during the first quiesce stage of CICS shutdown; that is, programs specified in the first half of the PLT for CICS shutdown. This is recommended, to prevent the Detector delaying CICS shutdown if it the Detector is not in the STOPPED state. For more information about using PLTSD programs, see the *CICS/ESA Customization Guide*.

This causes the Detector to stop recording any transaction affinities in the CICS region, until you next start the collection of data by the Detector. This also destroys the data space, and saves the data collected to the affinity data VSAM files.

Note: If there are a large number of data records to be saved, the CAFF screen may be frozen for some appreciable time, until the records have been saved.

You may want to stop the Detector when it has detected all affinities. This is indicated by the “Date/time of last change” field changing very infrequently and, if the optional periodic saves are performed, the “Records written last save” field being consistently near zero.

If CICS shuts down normally, but the Detector is not stopped, the Detector eventually detects that CICS is not running and terminates cleanly with a save.

Changing the Detector options

You can control the operation of the Detector by changing the options that it uses. Option values are preserved in the CICS Transaction Affinities Utility control file, CAUCNTL, so that they can be used across separate runs of the Detector. For more information about the control file, see “The control record VSAM file” on page 16.

Notes:

1. Generally, all options can be changed while the Detector is stopped. However, some options can also be changed while the Detector is paused, or running. (These options are identified in the descriptions on page 41.)
2. You can change the options even if the Detector is stopped and the Reporter is running (that is, reading the CAUCNTL file).

To review and change the options that the Detector uses, press the F4 function key of the Detector control screen, CAFF01. In response, the Detector options screen, CAFF02 (shown in Figure 8) is displayed.

To update the options screen, press Enter.

To return to the Detector control screen, press the F12 function key.

```
CAFF02            CAFF Operation Options                    Applid CICS330

  Modify the options and press Enter to update, or press Cancel (F12)

  Control options  1
    Perform periodic saves . . . . . Y    (Y=Yes or N=No)
    Restore data on start. . . . . N    (Y=Yes or N=No)
    Multiple signon with same userid . N    (Y=Yes or N=No)
    Size of dataspace. . . . . 16   (10 to 2000 Mbytes)
    Transid prefix (optional). . . . ___  (1 to 3 characters)

  Detect affinity types (Y=Yes or N=No)  2
  Inter-transaction     Transaction-system       Transaction-system
    ENQ, DEQ . . . . Y    INQUIRE, SET . . . Y     WAIT . . . . . Y  3
    TS Queue . . . . Y    ENABLE, DISABLE. . Y     DISCARD. . . . . Y  5
    ADDRESS CWA. . . Y    EXTRACT. . . . . Y
    RETRIEVE WAIT. . Y    COLLECT STATS. . . Y
    LOAD . . . . . Y    PERFORM. . . . . Y
    GETMAIN SHARED . Y    RESYNC . . . . . Y
    CANCEL . . . . . Y

Last updated by <userid>  on 03/26/93 09:04:35  4

F1=Help  6                                                  F12=Cancel
```

Figure 8. Detector options screen

The Detector options screen, CAFF02, shows the following options:

Note: Each option can be changed only while the Detector is stopped, unless otherwise stated below.

1 The control options:

- Whether or not the affinity data collected is to be saved to the affinity data VSAM files if either:
  - More than 300 seconds has passed, or more than 1000 table elements have changed, since the last save.
  - You paused the Detector.

  (The autosave transaction, CAFB, writes the affinity data to the affinity data VSAM files automatically when you stop the Detector.)

Note: This option can be changed while the Detector is stopped, paused, or running.

- Whether or not to restore the affinity data from the affinity data VSAM files when the Detector is started. This enables the affinity data collected on a run of the Detector to be added to the data collected from previous runs of the Detector. This should be used if you are gathering data for either one set of transaction identifiers at a time or for one set of commands at a time, or if the Detector is being run at varying times. It is also of particular value if the Detector terminates unexpectedly, because you do not have to start collecting affinity data all over again; you can start from the last time data was saved.

  Notes:

  1. The data restored is only for those affinity types that you have selected to be detected (by specifying Y for the affinity type on the CAFF02 screen).

     Note: Data is kept on file for all commands, but only the data for the commands that you have selected to be detected may change, and therefore is restored.

  2. This option can be changed while the Detector is stopped, paused, or running.

- Whether or not your conventions allow for more than one user to be signed on to CICS with the ***same userid*** at the same time. If so, set the *multiple signon with same userid* to Y; otherwise, the Detector may incorrectly deduce some affinity lifetimes and create erroneous affinity transaction groups (also known as *affinity groups*). This includes conventions where more than one user is simultaneously ***not*** signed on; that is they all take the default userid CICSUSER (for CICS/ESA Version 3 and Version 4) or blanks (for CICS/MVS 2.1.2).

  Also, if you are running the Detector in an AOR, then the userids examined depend on whether the userid is propagated from the TOR, or derived from the SESSION and CONNECTION resource definitions. In the last case, you should set multiple signon option to Y if your conventions allow the same AOR userids to be signed on to CICS at the same time.

  Notes:

  1. ***It is very important that this option is set correctly.***
  2. If you are about to start a new run of the Detector, and intend restoring data from the affinity data VSAM files, you should ensure that this option is the same as used in the previous run of the Detector (for which affinity data is to be restored).

- The size that you want to use for the dataspace to store the affinity data collected. The size of the data space is fixed for a run of the Detector. For information about estimating the size of the dataspace, see “Estimating the size of the MVS data space and files” on page 20.

  If the dataspace becomes full when running, the Detector terminates with abend code AUXB.

  Note: If affinity data is being saved, there may be a delay from the time that the dataspace became full until the Detector actually terminates.

- The prefix, 0 through 3 characters, of the transactions that you want to gather affinity data for. If you do not specify any characters, affinity data is collected for all transactions. If you specify a valid prefix (for example, AB_), affinity data is collected for only those transactions whose identifiers start with the prefix.

  Note: Leading and trailing blanks are ignored, but embedded blanks are treated as an error.

2 The affinity types to be detected. Whether or not the Detector is to detect the types of affinities listed.

For more information about restrictions affecting the detection of affinity commands, see "Detector" on page 10 and Appendix A, "Details of what is detected" on page 75.

3 WAIT commands are listed as transaction-system affinities, because only half of the affinity can be detected. (The other half, an MVS POST command or hand-posting, cannot be detected.) You should investigate such affinities further, and if needed change the output from the Reporter to create basic affinity transaction groups for them.

4 The date and time when the options were last updated, and the userid of the user who made the updates.

Note: The date is displayed in the format defined by the DATFORM system initialization parameter.

5 DISCARD is shown for CICS/ESA Version 3 and Version 4 only. For CICS/MVS 2.1.2, ADDRESS CSA is shown.

6 Pressing PF1 for help does not save any changes made; you must press the Enter key to save changes.

Detector errors

If the CAFF transaction, CAFB transaction, or an exit program encounters a serious error, the Detector is stopped by abending CAFF and CAFB. The abend code, in the AUxx range, is accompanied by messages on the CAFF transient data queue that indicate the cause of the error. (For details about the messages, see Appendix D, "Messages and Codes" on page 87.)

If the abend code is not in the AUxx range, it is presumably a CICS transaction abend code, and should be resolved using the appropriate CICS *Messages and Codes* manual.

Note: If the CAFF transaction abends with code ATCH, ATNI, or AKCT, the Detector continues to collect affinity data. For all other abends of either the CAFF or CAFB transaction, the Detector is stopped. The Detector should stop cleanly, so after performing the actions suggested by the message explanations, you should be able to restart the Detector.

If a program check or MVS abend occurs in an exit program, it results in an abend of the transaction that caused the exit to be invoked. This probably indicates a problem with the Detector. You should use the CAFF transaction to stop the Detector, and contact IBM support if the evidence points to the Detector being at fault. Unless running under CICS/ESA 4.1, it is not possible to rerun the Detector until the CICS region has been restarted.

If the abend code is AICA, this may be caused by the Detector scanning the table of affinity data when the amount of affinity data is very large. You can prevent such AICA abends by increasing the value of the ICVR system initialization parameter.

Chapter 6. Running the Reporter

This chapter

Describes how to run the Reporter, which runs as a batch job to produce a report of the affinities found by the Detector. The commands reported on are those listed in “Commands detected by the CICS Transaction Affinities Utility” on page 9. For information about interpreting the report output by the Reporter, see “Using the affinity report” on page 52.

You can run the Reporter to produce a report of the affinities found in your CICS region and definitions for the basic affinity transaction groups that correspond to the report. The definitions for the basic affinity transaction groups are suitable for input to the Builder.

You can request a report from the Reporter by editing and running the job, CAUJCLRP. Before running the CAUJCLRP job, change the following as appropriate:

- The JOB accounting parameters
- The STEPLIB DD statement

  Specify the name of the CICS Transaction Affinities Utility load library in which you have installed the Reporter program, CAUREP.
- The CAUAFF1, CAUAFF2, and CAUAFF3 DD statements

  Specify the names of your affinity data VSAM files for this CICS region.

  Note: Each of the CAUAFF1, CAUAFF2, and CAUAFF3 files has a header record that specifies the applid of the CICS region that created the record. The Reporter checks these applids with the applid recorded in the CAUCNTL file, and proceeds only if all four applids are the same.
- The CAUCNTL DD statements

  Specify the name of your CICS Transaction Affinities Utility control VSAM file for this CICS region.
- The CMDGRPS DD statement

  Specify the affinity (command) types to be output in the report. Only those affinity types listed on this DD statement are shown in the report. (The types correspond exactly to the type options on the CAFF02 screen.) You can specify any of the following affinity types, with each type on a separate line, starting in column one:

  ```
  CANCEL      ENABLE      LOAD
  COLLECT     ENQ         PERFORM
  CSA         EXTRACT     RESYNC
  CWA         GETMAIN     RETRIEVE
  DISCARD     INQUIRE     TS
  ```

  If you do not specify any affinity types on the CMDGRPS DD statement, or specify CMDGRPS DD DUMMY, ***all*** affinity types are selected for reporting.

  The first part of the report lists the affinity types selected.

- The TRANGRPS DD statement

  Specify the name of the sequential data set to which the basic affinity transaction groups are to be sent.

- SYSPRINT DD statement

  Specify the destination for the report output by the Reporter.

Note: The Reporter cannot read files from the CAUAFF1, CAUAFF2, and CAUAFF3 VSAM files while the Detector has those files opened for update. Therefore, you should not run the Reporter at the same time as you are running the Detector.

Output from the Reporter

For each of the affinity types specified on the CMDGRPS DD statement, the Reporter outputs each individual affinity in both report format and as definitions for basic affinity transaction groups suitable for input to the Builder.

Notes:

1. The Reporter outputs basic affinity transaction group definitions for inter-transaction affinity transaction groups only.
2. Transactions that were not initiated from a terminal do not appear in a basic affinity transaction group. If none of the transactions in an inter-transaction affinity group were initiated from a terminal, a special reporting affinity relation of **Background** is used, no basic affinity transaction group is created, and you should ignore the affinity lifetime. This is because dynamic transaction routing programs (for example, that provided by CICSPlex SM) are only interested in transactions initiated from a terminal, as they are the only transactions that can be dynamically routed.

Affinity report

Figure 9 shows an example report for two affinities, a TS queue affinity and a CWA affinity. These were the only affinity types selected, as shown.

```
CICS TRANSACTION AFFINITIES UTILITY                                          1993/06/28  Page  1
AFFINITY TYPE REPORTING OPTIONS                                                   Applid=CICS330

Affinity Type       Reporting      Message
---------_---       ---------      ---------------------------------------
1
  Inter-Transaction Affinities  2
  ----------------------------

  ADDRESS             Yes
  CANCEL              No
  ENQ                 No
  GETMAIN             No
  LOAD                No
  RETRIEVE            No
  TS                  Yes
  CSA                 No

  Transaction-System Affinities
  ----------------------------

  COLLECT             No
  DISCARD             No
  ENABLE              No
  EXTRACT             No
  INQUIRE             No
  PERFORM             No
  RESYNC              No
  WAIT                No

CICS TRANSACTION AFFINITIES UTILITY                                  1993/06/28  Page  2  3
INTER-TRANSACTION AFFINITIES REPORT FOR ADDRESS CWA                               Applid=CICS330

Trangroup    :  CW.00000001
Affinity     :  GLOBAL
Lifetime     :  SYSTEM

Tranid    Program      Offset      Usage    Terminal
------    --------    --------    ------    --------
 AUXX     AUXXTST     000000CC         1       Yes
 CWA1     AUCWA       FFFFFFFF         2       Yes
          Total Transactions      :       2
          Total Programs          :       2
```

Figure 9 (Part 1 of 2). Example report output by the Reporter

```
CICS TRANSACTION AFFINITIES UTILITY                                    1993/06/28  Page  3 3
INTER-TRANSACTION AFFINITIES REPORT FOR TEMPORARY STORAGE COMMANDS           Applid=CICS330

   Trangroup    :  TS.00000001
   Affinity     :  LUNAME
   Lifetime     :  PCONV
   Queue        :  LOCA1         (D3D6C3C1F1404040)
   Recoverable  :  No            (MAIN)
#  Terminal Id  :  V102          (E5F1F0F2)

   Tranid    Program      Offset     Command      Usage    Terminal
   ------    --------    --------    --------    ------    --------
    AFTD     AFFYTSD     0000012E    DELETEQ         43       Yes
    AFTR     AFFYTSR     000002BE    READQ           43       Yes
    AFTW     AFFYTSW     00000260    WRITEQ          43       Yes
             Total Transactions      :       3
             Total Programs          :       3

   Trangroup    :  TS.00000002
   Affinity     :  LUNAME
   Lifetime     :  PCONV
   Queue        :  LOCA2         (D3D6C3C1F2404040)
   Recoverable  :  No            (MAIN)
#  Terminal Id  :  V102          (E5F1F0F2)

   Tranid    Program      Offset     Command      Usage    Terminal
   ------    --------    --------    --------    ------    --------
    AFTD     AFFYTSD     0000012E    DELETEQ         39       Yes
    AFTR     AFFYTSR     000002BE    READQ           39       Yes
    AFTW     AFFYTSW     00000260    WRITEQ          39       Yes
             Total Transactions      :       3
             Total Programs          :       3
```

Figure 9 (Part 2 of 2). Example report output by the Reporter

Notes for Figure 9:

1 Incorrect affinity types

This lists any affinity types that were specified incorrectly on the CMDGRPS DD statement of the CAUJCLRP job.

2 Affinity types reported

This lists any affinity types that were selected for reporting; that is, those affinity types specified correctly on the CMDGRPS DD statement of the CAUJCLRP job. The affinity types are listed under their associated affinity category; inter-transaction affinity and transaction-system affinity.

3 Affinities reports

For each affinity transaction group, this lists appropriate characteristics of the affinities, as given in the following notes.

Trangroup

The name of the affinity transaction group, assigned by the Reporter. This name is only used to cross-reference the group in the report to the corresponding affinity transaction group in the data set specified on the TRANGRPS DD statement for this run of the Reporter.

Note: The Trangroup value for a affinity transaction group may vary for one run to another of the Detector or Reporter.

Affinity The affinity relation. If appropriate, this also indicates if the relation was worsened from a less restrictive relation. For more information about worsened relations, see “Worsening of transaction affinities relations” on page 12.

Lifetime The affinity lifetime. If appropriate, this also indicates if the lifetime was worsened from a less restrictive lifetime. For more information about worsened lifetimes, see “Worsening of transaction affinities lifetimes” on page 12.

Queue (resource)
The resource causing the affinity. This may be the name of the resource (for example, `Queue : LOCA1 (D3D6C3C1F1404040)` as shown) or the address of the resource, depending on the type of affinity.

Note: Unprintable characters appear as a period (.).

Recoverable
Whether or not the resource is recoverable. For TS queues, this also indicates whether the queue is in auxiliary or main temporary storage.

Terminal Id
The Id of the terminal that the transactions which took part in the affinity
were initiated from. This information is only available for TS queue
affinity and is only meaningful if the affinity is LUNAME or worsened
from LUNAME to GLOBAL. Therefore, the terminal Id is included in the
report only in these cases.

Tranid The identifier of each transaction that took part in the affinity. It is possible for an affinity transaction group to contain only one tranid. An example of such a situation is where each part of a pseudo-conversation accesses a TS queue and each part runs under the same tranid.

Program The name of each program that took part in the affinity.

Offset The offset from the load point of the program of the BALR instruction of the EXEC CICS command causing the affinity.

#
APAR PN68267

APAR applied

The Reporter outputs a negative offset (X'FFFFxxxx') if it could not determine an offset; that is, either of the following is true:

- The program was defined with RELOAD(YES) before CICS/ESA 4.1
- The offset calculated is not within the program. This may indicate that the program (or perhaps language run-time code) has passed control to another program by using a non-CICS mechanism (for example, VS COBOL II dynamic call).

Notes:

1. This offset is ***not*** the same as the offset given by the Scanner; the offset given by the Scanner is the offset of the command argument 0 declaration from the start of the load module. (See page 81.)

#
APAR PN68267

APAR applied

2. If a negative offset (X'FFFFxxxx') is used, it is not possible to directly locate individual affinity commands within a program. The program must be scanned for all instances of the affinity command, as there may be more than one instance of the command.

Command
The EXEC CICS command causing the affinity.

Usage The number of times that this particular EXEC CICS command (with the transaction, program, and offset values reported) took part in the affinity, up to a limit of 5000.

Note: The usage count is used as an indication of the relative importance of the affinity, not as a completely accurate usage count. For performance reasons, when the usage count is incremented by the Detector, the "save to file" flag is not necessarily set to indicate that the record needs to be saved to the data file. The save flag is set as follows:

```
0 =< usage count < 10,  save flag set every increment
10 <= usage count < 100,  save flag set every 10 increments
100 <= usage count < 5000,  save flag set every 100 increments
5000 <= usage count, no increment nor save flag set
```

PN 79260

Apar added

If the usage count is **1+**, then at least one example of the affinity was seen, but the total number is unknown.

Terminal Whether this particular EXEC CICS command (with the transaction, program, and offset values reported) was ever issued by a transaction initiated from a terminal; that is, started as a result of terminal input or for an EXEC CICS RETURN TRANSID command. (This does not include ATI-started transactions.)

APAR PQ12612

Documentation for Apar pq12612 added 07/04/98

The word **Mix** in this column is used to indicate that a particular EXEC CICS command was both issued by a transaction that was initiated from a terminal and also by the transaction when it was initiated with no associated terminal.

Total Transactions
The total number of different transactions in the affinity transaction group.

Total Programs
The total number of different programs in the affinity transaction group.

Affinity transaction group definitions

The Reporter outputs affinity transaction group definitions suitable for input to the Builder (but *not* CICSPlex SM). Each definition consists of a unique transaction group name, a relation, a lifetime and a set of tranids.

Not everything that appears in the report appears as an affinity transaction group. In particular, transaction-system affinities do not appear because they are not of interest to a dynamic routing program; nor do transactions that were not initiated from a terminal (for the same reason).

Figure 10 on page 52 shows an example set of definitions to match the report in Figure 9 on page 47.

Notes:

1. The transaction group name is ***not*** a valid CICSPlex SM transaction group name, as the latter must be eight characters; it is used only as a cross-reference to the report.
2. MATCH or STATE attributes are ***not*** generated on CREATE TRANGRP commands, because those attributes are relevant only to the combined affinity transaction groups. For more information about MATCH and STATE attributes, see page 59.
3. The HEADER statement is generated so that the Builder can detect a new data set in its input concatenation. It also gives the CICS applid, and the date and time of the last Detector save, all obtained from the CAUCNTL control file. For more information about HEADER statements, see “HEADER statements” on page 63.

```
* HEADER APPLID(CICS330)  SAVEDATE(93/06/27)  SAVETIME(10:11:45);
*
* Generated by the CICS Transaction Affinities Utility (Reporter) on 1993/06/28
* Note: NOT suitable for input to CICSPlex SM
*
CREATE TRANGRP NAME(CW.00000001) AFFINITY(GLOBAL) AFFLIFE(SYSTEM   )
               DESC(ADDRESS CWA                  );
  CREATE DTRINGRP TRANGRP(CW.00000001) TRANID(AUXX);
  CREATE DTRINGRP TRANGRP(CW.00000001) TRANID(CWA1);
*
CREATE TRANGRP NAME(TS.00000001) AFFINITY(LUNAME) AFFLIFE(PCONV    )
               DESC(TS.LOCA1     D3D6C3C1F1404040 );
  CREATE DTRINGRP TRANGRP(TS.00000001) TRANID(AFTD);
  CREATE DTRINGRP TRANGRP(TS.00000001) TRANID(AFTR);
  CREATE DTRINGRP TRANGRP(TS.00000001) TRANID(AFTW);
*
CREATE TRANGRP NAME(TS.00000002) AFFINITY(LUNAME) AFFLIFE(PCONV    )
               DESC(TS.LOCA2     D3D6C3C1F2404040 );
  CREATE DTRINGRP TRANGRP(TS.00000002) TRANID(AFTD);
  CREATE DTRINGRP TRANGRP(TS.00000002) TRANID(AFTR);
  CREATE DTRINGRP TRANGRP(TS.00000002) TRANID(AFTW);
```

Figure 10. Example basic affinity transaction group definitions

Once these definitions have been created, you can edit them to add extra definitions for affinities that the Detector could not detect, or to modify definitions in the light of further knowledge about the affinity (for example, correct a worsened lifetime). The report output from the Scanner may be particularly useful at this stage. (See “Using the affinity report.”)

Using the affinity report

The affinity report has the following two main purposes:

1. To help you to understand the affinities present in the CICS region concerned.
2. To help you modify the affinity transaction group definitions before they are input to the Builder, if such modification is required.

You need to be able to:

Notes:

1. Assume that the affinity information is complete.
2. Assume that any worsening of affinity relation or affinity lifetime by the Detector does not create too pervasive an affinity (this inhibits the effectiveness of dynamic transaction routing).
3. Investigate whether any application changes would be appropriate to reduce the amount of affinity.

Understanding the affinities

The inter-transaction affinities listed in the report highlight those transactions that have affinities with other transactions.

Understanding the affinities present in the CICS region enables you to determine which of the affinities are most pervasive. If you decide that it is worth changing your application programs, it is generally more cost-effective to remove the most pervasive affinities, as those affinities most restrict dynamic transaction routing. The most pervasive affinities are those with a relation of GLOBAL, or a lifetime of SYSTEM or PERMANENT, and are heavily used.

The transaction-system affinities listed in the report highlight those transactions that use system programmer type commands. It may not be appropriate to dynamically route such transactions, as their action may be tied to a particular CICS region (as opposed to a particular set of other transactions).

The affinity report also lists affinities that occur between transactions that were not initiated from a terminal, the special background relation. This information is really for completeness, as such transactions cannot be dynamically routed.

To establish complete affinity information, you should use as many code paths as possible while running the Detector, because it can find affinities only if the commands that cause the affinity have been executed. However, the Scanner detects all instances of affinity commands in the corresponding load library. So a comparison of the Reporter and Scanner output is very useful when establishing complete affinity information.

APAR PN68267

APAR applied.

Important note

Both the Reporter and Scanner may identify commands that, on closer examination, do not cause real affinities. You must relate the output from the Reporter and Scanner to your knowledge of your applications, to distinguish between such commands and those commands that cause real affinities that impact CICS dynamic transaction routing.

For more information about interpreting information in the affinity report, see Appendix C, “Useful tips when analysing CICS Transaction Affinities Utility reports” on page 85.

Modification of affinity transaction groups

You may need to make the following modifications to the affinity transaction groups before they are input to the Builder:

- Remove false affinities.

  False affinities may arise because the sharing of a resource is done on a read-only basis, so it is possible for the resource to be replicated across cloned CICS regions. The prime example of this is a read-only CWA, where the CWA is set up at CICS startup (for example, from a PLTPI program), and only ever read thereafter. (An alternative way to remove this false affinity is to prohibit detection of ADDRESS CWA by the Detector using the CAFF options.)

- Remove affinity relation worsening.

  An affinity that has a relation of LUNAME or USERID may be worsened to GLOBAL because the Detector has not seen enough examples of the affinity to be convinced that the affinity is related to a terminal or userid. You should change this to LUNAME or USERID (and correct the lifetime) if you know that the affinity really is terminal- or userid-related.

- Remove affinity lifetime worsening.

  An LUNAME affinity that has a lifetime of LOGON, or a USERID affinity that has a lifetime of SIGNON, may be worsened to SYSTEM or PERMANENT because there are limitations in the ability of the Detector to observe logoffs or signoffs. You should change this to LOGON or SIGNON if you know that really is the correct lifetime.

  # **APAR PN68267**

  # APAR applied

- # Change LUNAME affinity relation to USERID

  # An LUNAME affinity group may be *both* LUNAME *and* USERID, because all
  # instances of all transactions in the group were initiated from the same terminal
  # by the same userid. This appears in the report as LUNAME, because
  # LUNAME takes precedence. If you know that the affinity is primarily
  # userid-related, you should change the affinity to USERID. (This may be
  # indicated by other, similar, affinity groups appearing in the report with USERID.)

- Add WAIT affinities

  The Reporter reports the use of WAIT EVENT, WAITCICS and WAIT EXTERNAL commands as transaction-system affinity, because the Detector is unable to detect the corresponding posting of the ECBs being waited upon. The posting transactions need to be identified and affinity transaction groups created to describe the affinities. The output from the Scanner may be particularly useful here, as it finds programs that issue MVS POST.

- Add other affinities

  Scanner output or your knowledge of your applications may identify additional affinities. You should create affinity transaction groups to describe any such extra affinities.

- Add GETMAIN storage sharers

  The Detector cannot detect transactions that share storage other than via EXEC CICS commands. Although it detects GETMAIN SHARED/FREEMAIN

affinities, the address of the storage may have been passed to a third transaction. You should add such transactions to the affinity transaction group.

- Correct LOAD HOLD affinities on CICS/MVS 2.1.2

  If you run the Detector with CICS/MVS 2.1.2, it assumes that all programs loaded with the HOLD option are defined as RELOAD(NO). If a program was defined as RELOAD(YES), there may be another transaction that freed the program, which the Detector was unable to detect. You should add such transactions to the appropriate affinity transaction group.

Temporary storage compression

If your temporary storage queue names contain a unique counter or a termid, a very large number of basic affinity transaction groups may be created, for what may seem to be a small number of logical queues. For example, consider the queues ABCD0001 through ABCD1000, whose names comprise a fixed part (ABCD) and a counter (0001 through 1000). This might cause 1000 basic affinity transaction groups, each with relation, LUNAME, lifetime PCONV, and transactions ABCD and ABCE. This is one logical queue, ABCD*, which causes an affinity that may be described by ***one*** affinity transaction group. However, the result is 1000 basic affinity transaction groups.

The affinity data may be more readable if compressed to its logical form. You can use the Builder to do this, because it combines all affinity transaction groups that contain the same transaction IDs. The Builder output for the previous example is one affinity transaction group with relation LUNAME, lifetime PCONV, and transactions ABCD and ABCE.

Using the IBM Cross System Product

The following information about the IBM Cross System Product (CSP) concentrates on tests carried out running CSP 3.3, but in general the information also applies to later releases of CSP.

If you use the IBM Cross System Product to develop your applications and, in particular, use CSP/AE (Application Environment) as the run-time environment for the applications, the Reporter report will contain a large number of transaction groups. These groups are created because of the way that CSP/AE uses EXEC CICS commands, and in many cases do not cause real affinities.

Affinity analysis for a CICS region containing CSP 3.3 applications

This section concentrates on version 3.3 of IBM's CSP 4GL application generator. There are two components to CSP:

- CSP/AD (Application Development) is used to develop the applications
- CSP/AE (Application Environment) is the run-time environment for application execution.

When CSP 3.3 is used to develop and execute CICS pseudo-conversational applications, the main CSP affinity is LUNAME/PCONV TS queue affinity, which can be dealt with either by CICSPlex SM or by a queue-owning region (QOR). The only other real affinity likely to be encountered is the use of non-read-only CSP shared tables, and the scope of this affinity depends on the tables and applications involved.

CSP internally uses these CICS resources and commands in the following ways. They can cause transaction affinities, and these appear in the Transaction Affinities Utility report.

ENQUEUEs/DEQUEUEs
are used to serialize the loading of CSP tables and applications from VSAM files called ALFs (application load files). They are also used to serialize writing messages to TD destination CSMT.

Shared storage
When a CSP application or table or map from an ALF has completed loading, it is copied to shared storage. Note that some of these tables may be defined by the application developer as SHARED and made resident by the CSP utility program ALFUTIL. Such tables may be shared between applications, and may be updated.

Temporary storage queues
CSP allows division of applications into 'segments'. This is just another name for a pseudo-conversational application. CSP uses TS to save state data between transactions in the pseudo-conversation, building the TS queue name from the termid to ensure uniqueness.

SPI commands
are used to inquire on system attributes such as the version and release of CICS in use, to set up and share a user exit global work area (GWA), and to obtain file characteristics of the ALFs.

Detailed affinity analysis

Each of the above command scenarios is dealt with below. A description of how the use of the command appears in the Transaction Affinities Utility Reporter report is given, followed by an assessment of the affinity problem it causes, if any. However, it is helpful first to expand on the structure of a CSP segmented application.

The default CICS transaction identifier that CSP provides for applications is XSPS, although this is normally replaced by a unique transid for the application concerned. CSP transactions are defined so that the initial program is DCBINIT or DCBRINIT, the former for the first segment (that is, the first transaction in a pseudo-conversation), the latter for subsequent segments. These two CSP programs ensure that the correct environment is built for the application, including loading of programs and tables and saving and restoring of state data. DCBINIT and DCBRINIT branch to other CSP programs, but these other programs are not known to CICS. This means the Transaction Affinities Utility Reporter report shows DCBINIT or DCBRINIT as the program containing the affinity command, but the command offset is the generic x'FFFFFFFF'. In fact, the CSP program that issues most EXEC CICS commands is DCBMODS.

It is very important to note that a single report transaction/program/offset entry can conceal several affinity commands. Although the Transaction Affinities Utility Detector has correctly logged, and deduced information from, all the commands, it is only the first one encountered that is described in full in the report. So the Transaction Affinities Utility may report DCBINIT issuing only ENQUEUEs, but in reality DCBMODS is issuing both ENQUEUEs and DEQUEUEs. Similarly, the Transaction Affinities Utility may report that DCBINIT is issuing only WRITEQ TS commands, but in reality DCBMODS is issuing READQ TS and DELETEQ TS as

well. (The Transaction Affinities Utility Scanner shows that this is indeed the case when it is run against the CSP/AE load library.)

Note that there is a unique generic offset for each different command type within an affinity group. The generic offset is zero minus the group/function code for the command. So, for example, ENQUEUEs appear with x'FFFFEDFC', and DEQUEUEs appear with x'FFFFEDFA'. This is also the case for other command types.

ENQUEUE/DEQUEUE

There is an EQ affinity group in the report for each table or application or map that is loaded. The resource used in each case starts with 'FZE' and contains the name of the load module concerned. Other resources that may appear are 'FZELOAD' and 'FZETUTRI'. The programs involved are DCBINIT and DCBRINIT.

Upon analysis, this use of ENQUEUE/DEQUEUE does not cause affinity. Here, the ENQUEUE/DEQUEUE is being used to serialize a browse on an ALF, so that another CSP application in the same CICS region does not interfere with the loading process. If multiple CICS regions were cloned, each cloned CICS region must perform this same loading process, but this has no effect on any of the other CICS regions. So the ENQUEUE/DEQUEUE is not CICSplex wide and does not cause affinity. All that is required is to ensure that each CICS has access to the ALFs. Because these are used read-only by CSP/AE, the ALFs may be shared without resorting to the overhead of function shipping.

There may also be an EQ affinity group in the report with a resource name of 'CSMT when CSP serializes writing of information to TD destination CSMT. This does not cause affinity because each cloned CICS has its own CSMT.

GETMAIN SHARED

There is a GM affinity group in the report for each pair of transactions that were observed performing GETMAINs and FREEMAINs on shared storage. The programs involved are DCBINIT and DCBRINIT.

Upon analysis, this use of GETMAIN SHARED may cause affinity. It depends on the application. **If the storage obtained is for a CSP shared table that is updated by applications, there will be affinity**. Otherwise, there should not be affinity, because the program or table concerned is read-only and therefore duplicate copies are loaded by each cloned CICS region in the CICSplex.

The scope of any affinity depends on the use of the shared table by the application(s) concerned. The application developer decides this use. But it is extremely important to note that the Transaction Affinities Utility can detect only the transaction issuing the GETMAIN and the transaction issuing the FREEMAIN, and **not** intermediate transactions sharing the storage.

Note that unmatched GETMAIN SHARED commands may also appear in the report. This means that the Transaction Affinities Utility has seen a GETMAIN SHARED but as yet no matching FREEMAIN. The discussion in the rest of this section applies in this case also.

Temporary storage queues

There are temporary storage (TS) affinity groups in the report for each terminal that participated in a pseudo-conversation where the transactions involved are developed using CSP. The TS queue names are all of the form 'EZExtttt' where x is either A, C, R or T, and tttt is the termid of the terminal concerned. The programs involved are DCBINIT and DCBRINIT. The affinity group should be LUNAME/PCONV.

Upon analysis, this use of TS does cause affinity. Here, CSP is saving state data between transactions in the pseudo-conversation. But, because the TS queue contains the termid of the terminal, the affinity must be LUNAME; that is, terminal oriented. And because this technique is applicable only to pseudo-conversational applications, and the TS queue is deleted by CSP at the end of the pseudo-conversation, the lifetime must be PCONV. **Therefore, there is LUNAME/PCONV affinity**.

This affinity may be dealt with by either defining the affinity as LUNAME/PCONV to CICSPlex SM (which still permits good workload balancing if the pseudo-conversations are not excessively long) or, alternatively, by creating a queue-owning region (QOR) to which all TS queue requests from all cloned CICS regions are function shipped.

There is an interesting point to note here. Unrelated transactions may appear in the same affinity group; that is, it looks as though different applications have shared the same TS queue. In fact they have not; they have simply reused the TS queue name. This occurs because the TS queue name 'EZExtttt' is not flexible enough to incorporate a unique application identifier. The probable result of this is that the Transaction Affinities Utility Builder combines all transactions in all pseudo-conversational applications into a **single** affinity group for CICSPlex SM with an affinity of LUNAME and a lifetime of PCONV. The presence of one group rather than a group for each application is actually not important. When dynamic routing, the affinity still ends when the current pseudo-conversation ends, so the effect is exactly the same.

It is useful to have applied the PTF for CSP APAR PN45100, because this adds deletion of 'EZExtttt' TS queues after a transaction abend.

SPI commands

There are transaction-system affinities in the report indicating that commands such as EXTRACT EXIT and INQUIRE SYSTEM have been used. These are concerned with establishing the correct environment for a transaction. The programs involved are DCBINIT and DCBRINIT.

Transaction-system affinity can mean that a transaction is tied to a particular CICS, or more likely, as in this case, that the transaction should be run on all cloned CICS regions. Because it happens automatically there is no affinity problem. If multiple CICS regions are cloned, each cloned CICS region must perform this processing and will do so, but this will have no effect on any of the other CICS regions.

Chapter 7. Running the Builder

This chapter

Describes how to run the Builder, which runs as a batch job to build affinity transaction groups suitable for input to the CICS system management product, CICSPlex SM.

You can run the Builder to build affinity transaction groups suitable for input to the CICS system management product, CICSPlex SM. The Builder takes as input a set of files containing basic affinity transaction groups, combines those groups, and outputs a file containing combined affinity transaction groups.

Note: You can use the Reporter to produce files of basic transaction affinity groups for input to the Builder. The files can be from several runs of the Detector (for example, against a production CICS region and a test CICS region), but must be for the same workload.

You need to create combined affinity transaction groups for CICSPlex SM, because CICSPlex SM requires that a transaction identifier be in one transaction group only. (The Builder satisfies this by combining groups that contain the same transaction identifier.)

You can run the Builder by editing and running the job, CAUJCLBL. Before running the CAUJCLBL job, change the following as appropriate:

- The JOB accounting parameters
- The PARM parameter of the EXEC statement; for example:

```
//BUILD    EXEC PGM=CAUBLD,
// PARM=('STATE=ACTIVE,MATCH=LUNAME,DSPSIZE=16',
//       'CONTEXT=CICPLEX1')
```

[DSPSIZE={16|number}]
Specify the size, in the range 2 through 2000 (MB), of the dataspace created internally by the Builder to store the group tables. The default value is 16 (MB).

[MATCH={LUNAME|USERID}]
Specify the filter that CICSPlex SM is to use for workload separation, and which applies to all combined affinity groups produced by the Builder.

[STATE={ACTIVE|DORMANT}]
Specify whether the combined affinity groups are to be defined as active or dormant to CICSPlex SM.

[CONTEXT=plexname]
Specify the name, one through eight characters, of a CICSplex. If you specify this parameter, the Builder generates a CICSPlex SM CONTEXT statement, which enables CICSPlex SM to associate the combined affinity transaction groups with a particular CICSplex that it is managing. The default is to *not* generate a CONTEXT statement; in which case, CICSPlex SM assumes the local CMAS.

For more information about defining transaction groups to CICSPlex SM, see the *CICSPlex System Manager Setup and Administration* manual, SC33-0784.

- The STEPLIB DD statement

  Specify the name of the CICS Transaction Affinities Utility load library in which you have installed the Builder program, CAUBLD.

- The REPGRPS DD statement

  Specify the (concatenation of) names of the sequential data sets containing the basic affinity transaction groups to be input to the Builder. The Builder reads each line of the input data sets, and checks them for syntax and logic errors. For information about the valid syntax, see "Syntax for input to the Builder."

- The AFFGRPS DD statement

  Specify the name of the sequential data set to which the combined affinity transaction groups are to be sent. This data set is suitable for input to CICSPlex SM.

- SYSPRINT DD statement

  Specify the destination for the report output by the Builder.

Syntax for input to the Builder

The syntax in the sequential data sets input to the Builder is similar (but not identical) to that allowed by CICSPlex SM. (For more information, see the *CICSPlex System Manager Setup and Administration* manual.) The ***differences*** are given in the following list:

1. Only CREATE statements for TRANGRPs and DTRINGRPs, REMOVE statements for TRANGRPs, CONTEXT statements, and line comments are allowed. (A line comment is a line that starts with an asterisk (*) in column 1). Additionally, the HEADER statement is allowed for the Builder, but *is not* a CICSPlex SM statement.
2. Block comments delimited by '/*' and '*/' are not recognized.
3. Transaction group names of up to 11 characters are allowed. (CICSPlex SM only allows 8 characters.)
4. A CREATE TRANGRP statement must have exactly one NAME, one AFFINITY and one AFFLIFE value. MATCH and STATE values are optional and ignored (they are overridden by the values on the PARM statement or the default). A DESC value is optional and ignored. Any other keywords are reported as errors.
5. A CREATE DTRINGRP statement must have exactly one TRANGRP and one TRANID value. Any other keywords are reported as errors.
6. REMOVE TRANGRP statements are optional and are ignored by the Builder. However, if a REMOVE TRANGRP statement appears in an input data set, it must have exactly one NAME value. Any other keywords are treated as errors.
7. CONTEXT statements in the input data set are optional and are ignored by the Builder. They are overriden by the CONTEXT operand of the PARM statement (if specified) or the default.
8. A HEADER statement requires no other keywords. APPLID, SAVEDATE and SAVETIME are all optional, and if specified their values are not validated. The

HEADER statement must end in a semi-colon and should not span lines. Each input data set must start with a HEADER statement. (See “HEADER statements” on page 63.)

9. If a line comment contains the characters HEADER anywhere in it, it is not treated as a comment, and is parsed like any non-comment line, in case it is a HEADER statement. Otherwise comment lines are thrown away.
10. The only valid values for AFFINITY are GLOBAL, LUNAME, USERID. NONE is not allowed.
11. Keywords and values (including surrounding brackets) must not be split across input lines.
12. Nested brackets are not allowed within values.
13. The Builder is case sensitive. This applies to both keywords and their values. (Keywords must be in upper case.)

Any syntax error causes an error message to be issued. Logic errors are also possible; for example, CREATE DTRINGRP before CREATE TRANGRP, and also cause error messages to be issued.

Any such errors do not cause the Builder to terminate immediately, but normally cause a skip to either the next keyword or the next statement, depending on the error. The Builder terminates with return code of 8 when EOF is finally reached. An error report lists all errors encountered. For each error, the line containing the error is output, plus up to 4 preceding lines for the same statement to put the error in context, plus the error message.

```
input_statement  = {create_statement |
                    remove_statement |
                    header_statement |
                    context_statement |
                    comment}

create_statement = CREATE
                   {create_trangrp |
                    create_dtringrp}
                   ;

create_trangrp   = TRANGRP
                   NAME     (trangroup)
                   AFFINITY ({GLOBAL|LUNAME|USERID})
                   AFFLIFE  ({PERMANENT|SYSTEM|LOGON|SIGNON|PCONV})
                   [DESC    (string)]
                   [MATCH   ({LUNAME|USERID})]
                   [STATE   ({ACTIVE|DORMANT})]

create_dtringrp  = DTRINGRP
                   TRANGRP (trangroup)
                   TRANID  (tranid)

remove_statement = REMOVE
                   TRANGRP
                   NAME     (trangroup)
                   ;

context_statement = CONTEXT
                    [plexname]
                    ;

header_statement = HEADER
                   [APPLID   (applid)]
                   [SAVEDATE (date)]
                   [SAVETIME (time)]
                   ;

comment          = '*'
                   [string |
                    header_statement]
```

Figure 11. Builder input syntax

HEADER statements

The HEADER statement is specific to the Builder, and is not a CICSPlex SM statement. It is produced by the Reporter, and is needed by the Builder to create unique transaction group names.

The Reporter generates temporary transaction group names (for example, CW.00000001 and TS.00000001) while it is running, and stores these names in the output data set for that run. However, the Builder can take several Reporter data sets as input, and may therefore get the same transaction group name from different input data sets (describing different affinity transaction groups).

To ensure that the transaction group names are unique, the input transaction group names are qualified by the input data set name. To do this, when the Builder reads a HEADER statement (the first line of an input data set), it obtains the data set name from MVS/ESA. The HEADER statement is *vital*, because without it the Builder cannot detect the change from one input data set to another.

Warning: If you omit a HEADER statement, the Builder may generate error messages or add transactions to the wrong group, and gives incorrect line numbers in the error report and an incomplete report of data sets processed.

Output from the Builder

The Builder outputs a file containing a set of definitions for combined affinity transaction groups, and a report listing the combinations that occurred.

Combined affinity transaction group definitions

Before each definition of a combined group in the output file, the Builder adds a commented out REMOVE command for that group. If you already have combined groups of the same name, you should check that it is appropriate to delete them, before uncommenting the REMOVE command.

The name of each combined affinity transaction group is derived from the first (alphanumerically) transaction identifier in the combined group. For example, if ABCD was first, the transaction group name would be ABCDGRP.

Note: For CICSPlex SM, the name of each combined affinity transaction group must be unique.

For example, Figure 12 on page 64 shows the set of combined definitions that correspond to the basic definitions in Figure 10 on page 52, assuming that a MATCH filter of LUNAME, a STATE of ACTIVE, and a CONTEXT of CICPLEX1 was specified on the PARM statement.

```
* HEADER  APPLID(BUILDER )  SAVEDATE(93/06/27)  SAVETIME(12:00:51);              1
*
* Generated by CICS Transaction Affinities Utility (Builder) on 1993/06/28
* Note: Suitable for input to CICSPlex SM
*
CONTEXT CICPLEX1;
*
* REMOVE TRANGRP NAME(AFF1GRP );
CREATE TRANGRP NAME(AFF1GRP ) AFFINITY(LUNAME) AFFLIFE(SYSTEM   )
               MATCH(LUNAME) STATE(DORMANT);
  CREATE DTRINGRP TRANGRP(AFF1GRP ) TRANID(AFF1);
  CREATE DTRINGRP TRANGRP(AFF1GRP ) TRANID(AFF2);
  CREATE DTRINGRP TRANGRP(AFF1GRP ) TRANID(AFF3);
  CREATE DTRINGRP TRANGRP(AFF1GRP ) TRANID(AFF4);
  CREATE DTRINGRP TRANGRP(AFF1GRP ) TRANID(AFF5);
  CREATE DTRINGRP TRANGRP(AFF1GRP ) TRANID(AFF6);
  CREATE DTRINGRP TRANGRP(AFF1GRP ) TRANID(AFF7);
  CREATE DTRINGRP TRANGRP(AFF1GRP ) TRANID(AFF8);
*
* REMOVE TRANGRP NAME(AFTDGRP );
CREATE TRANGRP NAME(AFTDGRP ) AFFINITY(LUNAME) AFFLIFE(PCONV    )
               MATCH(LUNAME) STATE(DORMANT);
  CREATE DTRINGRP TRANGRP(AFTDGRP ) TRANID(AFTD);
  CREATE DTRINGRP TRANGRP(AFTDGRP ) TRANID(AFTR);
  CREATE DTRINGRP TRANGRP(AFTDGRP ) TRANID(AFTW);
*
* REMOVE TRANGRP NAME(AUXXGRP );
CREATE TRANGRP NAME(AUXXGRP ) AFFINITY(GLOBAL) AFFLIFE(SYSTEM   )
               MATCH(LUNAME) STATE(DORMANT);
  CREATE DTRINGRP TRANGRP(AUXXGRP ) TRANID(AUXX);
  CREATE DTRINGRP TRANGRP(AUXXGRP ) TRANID(CWA1);
```

Figure 12. Example definitions for combined affinity transaction groups

Notes:

1. The values of the SAVEDATE and SAVETIME fields in the HEADER statement give the latest save date and save time from all the input data sets. (See Figure 12 (**1**) and Figure 13 on page 66.)
2. The combined transaction groups can be input again to the Builder. For example, you may decide to:
   a. Use the Reporter then the Builder to produce combined groups for temporary storage affinities
   b. Use the Reporter then the Builder to produce combined groups for all other affinity command types
   c. Merge the two files output by the Builder in steps 2a and 2b, by inputting those files to the Builder together
   d. Input to CICSPlex SM the file output by the Builder in step 2c.

   This facility is particularly useful when dealing with the temporary storage compression problem, described in “Temporary storage compression” on page 55.

Combining basic affinity transaction groups

When the Builder combines two basic affinity transaction groups, it assigns relations and lifetimes to the combined group based on the relations and lifetimes derived from the basic groups. This may cause some worsening of the relations and lifetimes. For example, LUNAME combined with USERID gives GLOBAL. Table 10 through Table 13 on page 65 show the relations and lifetimes that result from the combination of basic affinity transaction groups.

To help you analyse the effect of combining basic transaction affinity groups, the Builder produces a report that lists the combinations that occurred.

Table 10. Resultant affinity relations

Relation A	**Relation B**	**Resultant relation C**
GLOBAL	Any relation	GLOBAL
LUNAME	LUNAME	LUNAME
LUNAME	USERID	GLOBAL
USERID	USERID	USERID

Table 11. Resultant affinity lifetimes (LUNAME relation)

Lifetime X	**Lifetime Y**	**Resultant lifetime Z**
PERMANENT	Any lifetime	PERMANENT
SYSTEM	SYSTEM	SYSTEM
SYSTEM	LOGON	SYSTEM
SYSTEM	PCONV	SYSTEM
LOGON	LOGON	LOGON
LOGON	PCONV	LOGON
PCONV	PCONV	PCONV

Table 12. Resultant affinity lifetimes (USERID relation)

Lifetime X	**Lifetime Y**	**Resultant lifetime Z**
PERMANENT	Any lifetime	PERMANENT
SYSTEM	SYSTEM	SYSTEM
SYSTEM	SIGNON	SYSTEM
SYSTEM	PCONV	SYSTEM
SIGNON	SIGNON	SIGNON
SIGNON	PCONV	SIGNON
PCONV	PCONV	PCONV

Table 13. Resultant affinity lifetimes (GLOBAL relation)

Lifetime X	**Lifetime Y**	**Resultant lifetime Z**
PERMANENT	Any lifetime	PERMANENT
Any other lifetime combination		SYSTEM

Data sets processed report

This report gives the names of all the input data sets (specified on the REPGRPS DD statement) that were read. This is produced even if errors occur in the input data sets.

Note: Only the data sets that contain a HEADER statement appear in the report.

```
CICS TRANSACTION AFFINITIES UTILITY                                                    1993/06/28    Page    1
BUILDER REPGRPS DATASETS PROCESSED REPORT

               Dataset Name                      CICS    Detector Last   Detector Last
                                                 APPLID    Save Date       Save Time
-------------------------------------------   --------  --------------  --------------
CICS330.TRANGRPS.MERGE1                       CICS330     93/06/25        09:05:09
CICS330.TRANGRPS.MERGE2                       CICS330     93/06/26        15:22:34
CICS330.TRANGRPS.ONE                          CICS330     93/06/27        12:00:51
```

Figure 13. Example data sets processed report

Empty transaction groups report

This report (for example, see Figure 14) gives all basic transaction groups (trangroups) that were defined, but contained no transactions. It is produced only if no errors occur in the input data sets. An empty trangroup probably indicates that you have made a mistake. (The Reporter cannot produce empty trangroups, so you must have created the input by hand, and probably omitted some corresponding CREATE DTRINGRP statements.)

```
CICS TRANSACTION AFFINITIES UTILITY                                                    1993/06/28    Page    2
BUILDER EMPTY TRANGROUP DEFINITIONS

      CICS330.TRANGRPS.EMPTY1

        G1           (GLOBAL SYSTEM   )     G2           (GLOBAL PERMANENT)     G3           (GLOBAL SYSTEM   )

      CICS330.TRANGRPS.EMPTY2

        L2           (LUNAME PERMANENT)     L3           (LUNAME LOGON    )     L4           (LUNAME PCONV    )
```

Figure 14. Example empty trangroups report

Group merge report

For each combined group, this report (for example, see Figure 15 on page 67) gives the constituent transactions and basic groups that went to comprise it. (This is a kind of audit trail.) It is produced only if there are no errors in the input data sets. It is very useful when establishing which basic group has caused the severe worsening of a lifetime. For example, in Figure 15 on page 67, four groups were merged, three were LUNAME and PCONV, one was LUNAME and SYSTEM; the latter caused the lifetime worsening.

```
CICS TRANSACTION AFFINITIES UTILITY                                          1993/06/28    Page    3
BUILDER GROUP MERGE REPORT

Trangroup   : AFF1GRP
Affinity    : LUNAME
Lifetime    : SYSTEM
Match       : LUNAME
State       : DORMANT

    Consists of Transactions

        AFF1  AFF2  AFF3  AFF4  AFF5  AFF6  AFF7  AFF8

    Consists of groups merged from

      CICS330.TRANGRPS.MERGE1

        TS.00000001  (LUNAME PCONV    )    TS.00000002  (LUNAME PCONV    )

      CICS330.TRANGRPS.MERGE2

        TS.00000001  (LUNAME SYSTEM   )    TS.00000002  (LUNAME PCONV    )

Trangroup   : AFTDGRP
Affinity    : LUNAME
Lifetime    : PCONV
Match       : LUNAME
State       : DORMANT

    Consists of Transactions

        AFTD  AFTR  AFTW

    Consists of groups merged from

      CICS330.TRANGRPS.ONE

        TS.00000001  (LUNAME PCONV    )    TS.00000002  (LUNAME PCONV    )

Trangroup   : AUXXGRP
Affinity    : GLOBAL
Lifetime    : SYSTEM
Match       : LUNAME
State       : DORMANT

    Consists of Transactions

        AUXX  CWA1

    Consists of groups merged from

      CICS330.TRANGRPS.ONE

        CW.00000001  (GLOBAL SYSTEM   )
```

Figure 15. Example group merge report

Error report

This report (for example, see Figure 16) gives any errors that were detected in the processing of the input files. These may be syntax errors or logic errors. Each error is accompanied by a message. (See Appendix D, “Messages and Codes” on page 87 for details of the messages.)

```
CICS TRANSACTION AFFINITIES UTILITY                                                    1993/09/08    Page    1
BUILDER REPGRPS ERROR REPORT

Dataset = CICS330.TRANGRPS.ERR1

          Line Number  Statement in error
          -----------  ------------------------------------------------------------------------------
                   5   CREATE TRANGRP NAME(G3         ) AFFINITY(GLOBAL) AFFLIFE(LOGON    );
                       CAU5038E Invalid AFFLIFE for AFFINITY
                   6   CREATE TRANGRP NAME(G4         ) AFFINITY(GLOBAL) AFFLIFE(SIGNON   );
                       CAU5038E Invalid AFFLIFE for AFFINITY
                   7   CREATE TRANGRP NAME(G5         ) AFFINITY(GLOBAL) AFFLIFE(PCONV    );
                       CAU5038E Invalid AFFLIFE for AFFINITY

Dataset = CICS330.TRANGRPS.ERR2

          Line Number  Statement in error
          -----------  ------------------------------------------------------------------------------
                  11   CREATE TRANGRP NAME(L4         ) AFFINITY(LUNAME) AFFLIFE(SIGNON   );
                       CAU5038E Invalid AFFLIFE for AFFINITY
                  15   CREATE TRANGRP NAME(U3         )
                  16          AFFINITY(USERID) AFFLIFE(LOGON    );
                       CAU5038E Invalid AFFLIFE for AFFINITY
```

Figure 16. Example error report

Chapter 8. Service

This chapter

Describes how you can get help from IBM for any problems that you encounter with the CICS Transaction Affinities Utility. You can get assistance by telephoning your local Support Center.

This chapter helps you decide when to contact the Support Center, and what information you need to collect before contacting the Center. It also gives you an understanding of the way in which IBM Program Support works.

When to contact the Support Center

Before contacting the Support Center, try to ensure that the problem belongs with the Center. Don't worry if you can't be sure that the problem is due to the CICS Transaction Affinities Utility itself. How sure you are depends on the complexity of your installation, the experience and skill levels of your systems staff, and the symptoms that you have been getting.

In practice, many errors reported to Program Support turn out to be user errors, or they cannot be reproduced, or they need to be dealt with by other parts of IBM such as Hardware CE or Systems Engineering. This indicates just how difficult it can be to determine the precise cause of a problem.

Dealing with the Support Center

Your first contact with the Support Center is the call receipt operator, who takes initial details and puts your problem on a queue. You are subsequently contacted by a Support Center representative, and your problem is taken from there.

The Support Center needs to know as much as possible about your problem, and you should have the information ready before making your first call. It is a good idea to put the information down on a problem reporting sheet, such as the one shown in Table 14 on page 70.

Table 14. Sample problem reporting sheet

PROBLEM REPORTING SHEET			
Date	Severity	Problem No.	Incident No.
Problem/Enquiry (See note.)			
MVS Rel	MVS Lvl	CAU Rel	CAU Lvl
CICS/ESA 4.1	CICS/ESA 3.3	CICS/ESA 3.2.1	CICS/MVS 2.1.2
Scanner	Detector	Reporter	Builder
Module	Abend/Prog CK	Incorrout	Wait
Loop	Message	Performance	Other
Documentation available			
Message	Abend	Transaction dump	System dump
Trace	Program output	Translator output	Compiler output
Symptom string	Other		
Actions			
Date	Name	Activity	
Resolution			
APAR	PTF	Other	

Note: Information given should be specific to the environment in which you were running the CICS Transaction Affinities Utility when you encountered a problem. For example the CICS version is for the CICS region against which you were using the CICS Transaction Affinities Utility, and the MVS release and level are those that the CICS region were running under.

There are two advantages of using a problem reporting sheet:

1. You will be communicating with the IBM Support Center by telephone. So, with all your findings before you on a sheet of paper, you are prepared to respond to the questions that you may be asked.
2. You should maintain your own in-house tracking system to record all problems. This information can then be used for planning, organizing, communicating, and establishing priorities for controlling and solving these problems.

What the Support Center needs to know

When you contact the Support Center, you need to give the operator the name of your organization and your access code. Your access code is a unique code authorizing you to use IBM Software Services, and you provide it every time you contact the Center. Using this information, the operator accesses your customer profile, which contains details of your address, relevant contact names, telephone numbers, and details of the IBM products at your installation.

The Support Center operator asks you if this is a new problem, or a further call on an existing one. If it is new, you are assigned a unique incident number. A problem management record (PMR) is opened on the RETAIN system, where all

activity associated with your problem is recorded. The problem remains “open” until it is solved.

Make a note of the incident number on your own problem reporting sheet. The Center expects you to quote the incident number in all future calls connected with this problem.

If the problem is new to you, the operator asks you for the source of the problem within your system software—that is, the program that seems to be the cause of the problem. As you are reading this book, it is likely that you have already identified the CICS Transaction Affinities Utility as the problem source.

You need to give a severity level for the problem. Severity levels can be 1, 2, or 3. They have the following meanings:

- Severity level 1 indicates that you are unable to use a program, resulting in a critical condition that needs immediate attention.
- Severity level 2 indicates that you are able to use the program, but that operation is severely restricted.
- Severity level 3 indicates that you are able to use the program, with limited functions, but the problem is not critical to your overall operation.

When deciding the severity of the problem, take care neither to understate it nor to overstate it. The Support Center procedures depend on the severity level so that the most appropriate use can be made of the Center’s skills and resources.

Finally, the call receipt operator asks you for a brief description of the problem, and may prompt you to quote the CICS symptom string, or to give any keywords associated with the problem. The primary keywords are ABEND, WAIT, LOOP, PERFORMANCE, and INCORROUT, corresponding exactly with the problem classification types used in the earlier parts of this book. Strings containing other keywords are also useful. These are not predefined, and might include such items as a message or message number, an abend code, and any parameters known to be associated with the problem.

The keywords are subsequently used as search arguments on the RETAIN database, to see if your problem is a known one that has already been the subject of an authorized program analysis report (APAR).

The following information is needed when problems with the CICS Transaction Affinities Utility are diagnosed by IBM. You should provide as much of this information as you can, so that it can be documented in the PMR for future reference.

- The CICS release for the CICS region against which you were running the CICS Transaction Affinities Utility: CICS/ESA 4.1, CICS/ESA 3.3, CICS/ESA 3.2.1, or CICS/MVS 2.1.2
- Which component of the CICS Transaction Affinities Utility is the PMR related to: Scanner, Detector, Reporter, or Builder
- The exact symptoms of the problem, or the nature of the query
- Any message numbers involved
- How the problem began

- If the problem began suddenly, any changes made before the onset of the problem
- Whether the problem is solid or intermittent
- Whether the problem can be recreated
- The name of any specific transaction, CICS region, or load library related to the problem
- A list of documentation available: auxiliary trace, internal trace, dumps, and so on
- Any other relevant information that you think may be valid.

You should keep all the information relevant to the problem, and any available documentation such as dumps, traces, and translator, compiler, and program output.

How your problem is subsequently progressed depends on its nature. The representative who handles the problem gives you guidance about what is required from you. The possibilities are described in the next section.

What happens next

Details of your call are passed using the RETAIN problem management system to the appropriate support group. Your problem, assuming it is associated with the CICS Transaction Affinities Utility, is put on the CICS queue. The problems are dealt with in order of severity level.

At first, a Level-1 representative uses the keywords that you have provided to search the RETAIN database. If your problem is found to be one already known to IBM, and a fix has been devised for it, a Program Temporary Fix (PTF) can quickly be dispatched to you.

If the RETAIN search is unsuccessful, the problem is passed to the CICS change team for further investigation. They try to respond in the PMR with their answer to the problem.

CICS Transaction Affinities Utility queries, and problems that turn out to be user errors, set up difficulties, or other non-code defects should be resolved in this way.

If any reported problems need further investigation, the CICS change team will contact you by telephone to discuss the problems in greater detail.

You might be asked to give values from a formatted dump or trace table. You might also be asked to carry out some special activity, for example to set a trap, or to use trace with a certain type of selectivity, and then to report on the results.

It might be necessary to have several follow-up telephone calls, depending on the complexity of the symptoms and your system environment. In every case, the actions taken by you and the Support Center are entered in the PMR. The representative can then be acquainted with the full history of the problem before any follow-up call.

The result of the investigations determines whether your problem is a new one, or one that is already known. If it is already known, and a fix has been developed, the fix is sent to you.

If the problem is new, an APAR may be submitted. This is dealt with by the CICS change team.

Appendix A. Details of what is detected

This appendix

Describes what is detected by the Detector and Reporter for each affinity type. Additionally, it highlights the differences, if any, with what the Scanner detects. (In general, the Scanner always detects more, because it covers paths that may not get exercised by the Detector, and because it can not see beyond the command argument 0 to eliminate commands that do not actually cause affinity.)

This information adds details to the general description in “What is detected” on page 11.

ENQ/DEQ

- The affinity here is between all transactions that ENQ or DEQ on a given resource. The match is made on the resource.
- It is possible for the ENQ/DEQ resource to be either a character string of length 1 to 255 bytes, or an address (which has an implied length of 4 bytes).
- It is possible for the affinity relation to be GLOBAL or USERID.
- The only possible lifetime is SYSTEM.
- Commands that result in a LENGERR condition are grouped together and treated as a resource of 'LENGERR'. All other conditions result in a valid resource and so are treated the same as a NORMAL command.
- Because of affinity record size limitations, character string resources of greater than 207 bytes in length are compressed to 207 bytes. The compression is achieved by removing bytes from the middle of the string (these are probably less significant). This means that such resources may be flagged as being the same when they are not, if the only variation was in the removed bytes. Please check all such compressed resources to see if that is the case. The Reporter flags such compression, and pads the resource back out to the correct length for the report, by inserting '?' characters.

Scanner differences: None.

TS commands

- The affinity here is between all transactions that use the same TS queue. It applies to both MAIN and AUXILIARY TS. The match is made on the name of the TS queue.
- It is possible for the affinity relation to be GLOBAL, LUNAME or USERID.
- The possible lifetimes are PCONV, LOGON, SIGNON, SYSTEM and PERMANENT. Note that a MAIN queue is always non-recoverable regardless of definition, so cannot cause PERMANENT.
- If a TS queue is defined as remote or a remote SYSID was specified on the TS command so that the command is function shipped to a remote CICS region, then no data is collected.

- Commands in error are treated the same as commands that give a NORMAL response, so data is collected.
- If a TS queue is created and deleted within the same task then no data is collected.

Scanner differences: Scanner detects all instances of TS commands.

LOAD HOLD/RELEASE

- The affinity here is between all transactions that LOAD HOLD and RELEASE the same program (or more probably table). The match is made on the program name.
- The LOAD and RELEASE protocol only applies to programs that are defined with RELOAD(NO). If the Detector can not establish the RELOAD attribute for some reason, then RELOAD(NO) is assumed.
- Once a LOAD HOLD has occurred for a program, any subsequent LOAD (with or without HOLD) or RELEASE is part of the affinity.
- It is only possible for the affinity relation to be GLOBAL.
- The only possible lifetime is SYSTEM.
- Commands in error are treated the same as commands that give a NORMAL response, so data is collected.
- LOAD with no HOLD for programs defined as RESIDENT are not treated as an affinity, because relying on residency for sharing is inherently unsafe (the program can be replaced by SET PROG() NEWCOPY).
- The incorrect use of RELEASE for a program defined with RELOAD(YES) is not detected.

Scanner differences: Scanner detects all instances of LOAD, not just LOAD HOLD, and all instances of RELEASE.

RETRIEVE WAIT/START

- The affinity here is between all the transactions that issue START commands for a particular transaction at a terminal, where that transaction issues RETRIEVE WAIT. The transaction that issues the RETRIEVE WAIT is also part of the affinity. The match is made on the transid.
- It is possible for the affinity relation to be GLOBAL or USERID.
- The possible lifetimes are SYSTEM or PERMANENT. PERMANENT is assumed if PROTECT is specified on any START.
- If the transaction to be STARTed is defined as remote or a remote SYSID was specified on the START command so that the command is function shipped to a remote CICS region, then no data is collected.
- Commands in error are treated the same as commands that give a NORMAL response, so data is collected.

Scanner differences: Scanner finds all instances of RETRIEVE WAIT, and all instances of START that either specify TERMID or omit NOCHECK or specify REQID (because of CANCEL affinity).

ADDRESS CWA

- The affinity here is between all transactions that issue ADDRESS CWA.
- The only affinity relation is GLOBAL.
- Lifetime is always SYSTEM.

Scanner differences: None.

GETMAIN SHARED/FREEMAIN

- The affinity here is between the transaction that obtains storage via GETMAIN SHARED and the transaction that frees the same piece of storage via FREEMAIN. Both transactions must be seen for there to be affinity. The match is made on storage address.
- However, the situation is complicated by the fact that the storage address may be passed to other transactions; and if they access the storage, ***they cannot be detected***, because the storage access does not take place through the CICS API.
- The affinity relation may be GLOBAL, LUNAME or USERID.
- The possible lifetimes are PCONV, LOGON, SIGNON and SYSTEM. However, the Detector always worsens LOGON and SIGNON to SYSTEM, because of limitations in the way that this affinity is detected.
- Commands in error are ignored, as there is no address on which to match GETMAIN with FREEMAIN, so no data is collected.
- A GETMAIN/FREEMAIN affinity is considered to be initiated from a terminal if the GETMAIN is initiated from a terminal. Whether the FREEMAIN was or not is irrelevant.
- Any unmatched GETMAIN SHAREDs are also reported if they have never matched by the time a Detector stop occurs. They are output in a separate report section. Note that on a start with restore data, they are not restored and are deleted from the affinity file.

Scanner differences: Scanner finds all instances of GETMAIN SHARED and all instances of FREEMAIN.

LOAD/FREEMAIN

- The affinity here is between the transaction that loads the program via LOAD and the transaction that releases the same program via FREEMAIN. The match is made on load point address.
- However, the situation is complicated by the fact that the load point address may be passed to other transactions (for example, the program is actually a table); and if they access the program, ***they cannot be detected***. This is analogous to storage address passing with GETMAIN SHARED/FREEMAIN.
- The LOAD and FREEMAIN protocol only applies to programs defined as RELOAD(YES). Note that HOLD is irrelevant, as CICS Program Control never sees the FREEMAIN nor knows the storage location of the individual task's copy, and so can not release the program at task end. This implies that all LOADs must be examined as they are all effectively LOAD HOLDs.
- The affinity relation may be GLOBAL, LUNAME or USERID.

- The possible lifetimes are PCONV, LOGON, SIGNON and SYSTEM. However, the Detector always worsens LOGON and SIGNON to SYSTEM, because of limitations in the way that this affinity is detected.
- Commands in error are ignored, because there is no load address on which to match LOAD with FREEMAIN, so no data is collected.

  LOADs with no SET option are ignored as no load address is returned, so no data is collected.
- A LOAD/FREEMAIN affinity is considered to be initiated from a terminal if the LOAD is initiated from a terminal. Whether the FREEMAIN was or not is irrelevant.
- Any unmatched LOADs are also reported if they have never matched by the time a Detector stop occurs. They are output in a separate report section. Note that on a start with restore data, they are not restored and are deleted from the affinity file.
- If you are using the CICS Transaction Affinities Utility on CICS/MVS 2.1.2, LOAD HOLD for programs defined with RELOAD(YES) appears under LOAD/RELEASE, because the Detector cannot determine the attributes of a program loaded with the HOLD option and assumes RELOAD(NO).

Scanner differences: Scanner finds all instances of LOAD and all instances of FREEMAIN.

CANCEL/DELAY/POST/START

- The affinity here is between the transaction that issues the DELAY, POST or START command and the transaction that issues the CANCEL command via REQID. The match is on REQID.
- In order for **another** task to CANCEL a DELAY, REQID must be explicitly specified on the DELAY command. If no REQID is specified on a DELAY command, it can not be cancelled. So only those DELAY commands that specify REQID can be detected.

  In order for another task to CANCEL a START or POST, it is not necessary to specify REQID on the command as CICS supplies a unique REQID that may be used (unless START specifies NOCHECK). So only START commands that do not both specify NOCHECK and omit REQID, and all POST commands, are detected.
- Further, data is not collected for commands that expire on entry to Interval Control, because they cannot be cancelled (because an ICE is not created). DELAY and POST commands get an EXPIRED response. For START commands there is no such response; so expired on entry is deduced if INTERVAL(0) was specified. This detects most expired on entry STARTs, but not all.
- START, DELAY and POST commands in error are ignored, so no data is collected.
- CANCEL commands that omit REQID are ignored, because they can not cancel another task, or that return a NOTFND response, as the ICE must have expired so the CANCEL failed. No data is collected for these.

- REQIDs are assumed to be unique; that is, there are not simultaneous pairs of START/CANCEL using the same REQID. Doing this violates CICS programming guidelines, and the results from CICS are unpredictable.
- The affinity relation for START may be GLOBAL, LUNAME or USERID. The possible lifetimes are PCONV, LOGON, SIGNON, SYSTEM and PERMANENT. The PROTECT option determines whether SYSTEM or PERMANENT would be used. However, the Detector always worsens LOGON and SIGNON to SYSTEM or PERMANENT, because of limitations in the way that this affinity is detected.

  The affinity relation for DELAY and POST may be GLOBAL, LUNAME or USERID. The possible lifetimes are only SYSTEM or PCONV. If LUNAME or USERID, it must be PCONV because neither DELAY nor POST exist beyond task termination.
- If the transaction specified on a START or CANCEL command is defined as remote or a remote SYSID was specified on the command so that the command is function shipped to a remote CICS region, then no data is collected. (It is not possible to function ship POST or DELAY commands.)
- A CANCEL affinity is considered to be initiated from a terminal if the START, DELAY or POST is initiated from a terminal. Whether the CANCEL was or not is irrelevant.

Scanner differences: Scanner detects all instances of POST, all instances of DELAY REQID, all instances of CANCEL REQID, and all instances of START that either omit NOCHECK or specify REQID or specify TERMID (because of RETRIEVE WAIT affinity).

SPI commands

- The commands included here are INQUIRE, SET, DISCARD, ENABLE, DISABLE, EXTRACT EXIT, COLLECT STATISTICS, PERFORM, RESYNC and ADDRESS CSA.
- The affinity here is not an affinity between transactions, but rather an affinity with the system that the command was issued on; that is, a transaction-system affinity. Such affinities do not generate transaction affinity groups, because it does not generally make sense to dynamically route such transactions.
- The use of these commands does require reporting however, because the system programmer should be aware of the transactions and programs that issue such commands.
- Use of ADDRESS CSA is only detected when running under CICS/MVS 2.1.2.

Scanner differences: None.

WAIT commands

- The affinity here is really an inter-transaction affinity between the issuer of the WAIT EVENT, WAIT EXTERNAL or WAITCICS command, and one or more posters. However, the poster of the ECB(s) associated with the WAIT command cannot be detected, because this is not performed via the CICS API. Only half of the affinity can be detected.
- This means affinity transaction groups cannot be created, because the affinity degenerates to an affinity with the system that the WAIT command was issued on; that is, a transaction-system affinity.
- The use of WAIT commands does require reporting however, because the system programmer should be aware of the transactions and programs that issue such commands, and should attempt to locate the posters and so create the correct inter-transaction affinity groups.

Scanner differences: None.

Appendix B. Correlating Scanner and Reporter output to source

This appendix

Describes how to match the EXEC CICS command in the Reporter report and/or the Scanner detail report with the actual program source code. It also gives some examples of the procedures described.

Reporter output

The reported offset of a command is the offset from the start of the load module of the BALR to the CICS stub. To get the offset from the start of the program, you should subtract the length of the CICS stub from the offset reported. (You may also need to subtract the lengths of any additional preceding CSECTs.) You can then use the compiler listing to find the command.

Scanner output

The reported offset of a command is the offset from the start of the load module of the command CICS argument zero[7] . This is a constant and therefore is located in the literal pool for the program. As with the Reporter, you should subtract the length of the CICS stub and preceding CSECTS to get the offset from the start of the program. You should then be able to locate the argument zero in the compiler listing. Next, you should match the argument zero to the command, which involves finding the instruction which referenced the argument zero, using the compiler listing.

Examples

This section gives some examples of the procedures for the Scanner.

Example 1–Assembler-language

Before the BALR to the CICS stub, the CICS translator generates an LA instruction with the argument zero as source, for example:

```
LA     14,=X'0202800008070000000000000000000000000000000000'
```

To locate the EXEC CICS command, you can match the argument zero in the literal pool with the same argument zero in the LA instruction.

[7] For an explanation of *argument zero*, see “Notes on Terminology” on page x.

Example 2–VS COBOL II

The literal pool in VS COBOL II is part of the CGT. Having calculated the offset from the start of the program, you should subtract the start of the CGT from your calculated offset to get the offset within the CGT. In the listing, there is an MVC instruction with the argument zero as the source, of the form:

```
MVC   D1(L,R1),D2(R2)       DFHEIV0             PGMLIT AT ...
```

where:

DFHEIV0 is the slot in working storage into which the argument zero is copied before the BALR to the CICS stub

D2 is the decimal offset of the argument zero within the CGT (which you have just calculated)

R2 is the CGT base register.

Once you know the offset of the argument zero within the CGT, you can find the MVC and hence the EXEC CICS command.

An example of finding an EXEC CICS command is given in Figure 17 on page 83.

For this, the calculations are:

```
Scanner offset     = X'7A6'
CICS stub length   = X'28'
Offset of CGT      = X'B8'
CGT base register  = GPR 10
Offset within CGT  = X'7A6' - X'28' - X'B8' = X'6C6' = 1734 (decimal)

MVC instruction looks like:
MVC   d(l,r),1734(10)       DFHEIV0             PGMLIT AT ...
```

To determine the EXEC CICS command:

1. Look at the Assembler-language for

   ```
   MVC   d(l,r),1734(10)       DFHEIV0             PGMLIT AT ...
   ```

   which occurs for the first MOVE

   ```
   001126  MOVE
   ```

2. Look at the COBOL source for the MOVE at line 001126. This is for the EXEC CICS WRITEQ TS command starting on line 001124.

For the Scanner output:

```
CICS TRANSACTION AFFINITIES UTILITY                                                    1993/10/19  Page     1
LOAD MODULE SCANNER - DETAILED LISTING OF CICS.DEVR330.LOCLLOAD

Module Name - ACCT04    /  Load Module Length - 000159D0  /  Module Entry Point - 00000028
Offset    Storage Content (HEX)                              EDF DEBUG  Possible Command                      Affinity
--------  -------------------------------------------------  ---------  -----------------------------------  --------
000007A6  0A02E0000700004100                                    00669   WRITEQ TS                             Trans
Total possible Affinity commands =         1
Total possible MVS POSTs         =         0
```

The COBOL source after translation was:

```
001123
001124               *EXEC CICS WRITEQ TS QUEUE('ACERLOG') FROM(ACCTERRO)
001125               *    LENGTH(ERR-LNG) END-EXEC.
001126                     MOVE '  \      00669   ' TO DFHEIV0                          97800000  1057
001127                     MOVE 'ACERLOG' TO DFHC0080                                             1034
001128                    CALL 'DFHEI1' USING DFHEIV0  DFHC0080 ACCTERRO ERR-LNG.                 EXT 1057 1034 380 861
```

The equivalent Assembler-language is:

```
 001126  MOVE
     002764  D210 8558 A6C6          MVC  1368(17,8),1734(10)     DFHEIV0                          PGMLIT AT +1718
     00276A  9240 8569               MVI  1385(8),X'40'           DFHEIV0+17
     00276E  D232 856A 8569          MVC  1386(51,8),1385(8)      DFHEIV0+18                       DFHEIV0+17
 001127  MOVE
     002774  D207 8340 ACEA          MVC  832(8,8),3306(10)       DFHC0080                         PGMLIT AT +3290
 001128  CALL
     00277A  4130 8558               LA   3,1368(0,8)             DFHEIV0
     00277E  5030 D1B0               ST   3,432(0,13)             TS2=0
     002782  4130 8340               LA   3,832(0,8)              DFHC0080
     002786  5030 D1B4               ST   3,436(0,13)             TS2=4
     00278A  4130 75A8               LA   3,1448(0,7)             ACCTERRO
     00278E  5030 D1B8               ST   3,440(0,13)             TS2=8
     002792  4130 9A0E               LA   3,2574(0,9)             ERR-LNG
     002796  5030 D1BC               ST   3,444(0,13)             TS2=12
     00279A  9680 D1BC               OI   444(13),X'80'           TS2=12
     00279E  4110 D1B0               LA   1,432(0,13)             TS2=0
     0027A2  4100 D150               LA   0,336(0,13)             CLLE@=2
     0027A6  0530                    BALR 3,0
     0027A8  5030 D158               ST   3,344(0,13)             TGT FDMP/TEST-INFO. AREA +0
     0027AC  58F0 A000               L    15,0(0,10)              V(DFHEI1  )
     0027B0  05EF                    BALR 14,15
     0027B2  50F0 D078               ST   15,120(0,13)            TGTFIXD+120
     0027B6  BF38 D089               ICM  3,8,137(13)             TGTFIXD+137
     0027BA  0430                    SPM  3,0
```

Figure 17. Example for finding an EXEC CICS command from the argument zero

Appendix C. Useful tips when analysing CICS Transaction Affinities Utility reports

APAR PN 68267

APAR applied

Sometimes the report produced by the Reporter from data gathered from the Detector can contain some results that appear odd at first glance. This appendix gives tips on how to resolve such results.

COBOL Affinities

If an application program is invoked using the native CALL statement, CICS COBOL run-time code issues an EXEC CICS LOAD HOLD for the program and branches to it directly. This only causes affinity if the program is not reentrant; that is, it modifies itself. Otherwise, there is no affinity.

CICS COBOL run-time code writes data to a TS queue if an abend occurs. The TS queue name used is “CEBR” plus the termid of the terminal, or blanks if no terminal. This does not cause affinity.

LOGON or SYSTEM when PCONV expected

When dealing with an application that is known to use TS queues within a pseudo-conversation, but never beyond, there may be occurrences in the report of affinity groups that appear as LUNAME/SYSTEM or LUNAME/LOGON, instead of the expected LUNAME/PCONV.

- A SYSTEM lifetime can be explained if the installation uses a session manager that logs users off after a pre-determined quiet time. When the log off occurs in the middle of a pseudo-conversation, the CICS Transaction Affinities Utility notices that the TS queue still exists and increases the lifetime to SYSTEM.
- A LOGON lifetime can be explained by the user switching off the terminal in the middle of a pseudo-conversation and causing a VTAM line error. This causes an error transaction to be attached internally at the terminal, and the Transaction Affinities Utility notices that the TS queue exists at the end of that transaction and increases the lifetime to LOGON.

In both these circumstances the real lifetime is PCONV, because although the TS queue exists at the end of the pseudo-conversation the data in it is never be used again. Normally the first action of a new pseudo-conversation is to delete the contents of all such TS queues for that terminal to ensure that everything is tidy.

Unrecognised Transids

Transids that consist of garbage data are reported in the CICS Transaction Affinities Utility report. Such transids are not known to CICS, and most contain the same hexadecimal data. This is probably caused by a bug in an application that causes the EIB to be overwritten.

Appendix D. Messages and Codes

This appendix

Describes the messages that the Builder, Detector, Reporter and Scanner can issue, and the transaction abend codes that the Detector can produce.

The messages are described in alphanumeric order, starting on this page. The abend codes are described in alphabetic order, starting on page 112.

As an aid to problem determination, this chapter also lists the meaning for each possible value of the call parameters that are included in the error messages issued if an error occurs on a call to the:

- Detector and Builder table manager, CAUTABM. (See page 118.)
- Detector CAFB request queue manager, CAUCAFP. (See page 120.)
- CICS Transaction Affinities Utility date formatter, CAUCAFDT. (See page 120.)

CAU2101W CAFF is already in use elsewhere

Explanation: The CAFF transaction is already being used when an attempt is made to start another CAFF transaction.

System Action: The second CAFF transaction is terminated, as only one instance of CAFF is permitted.

User Response: Check where CAFF is currently in use. Only one user should be attempting to run the Detector at a given time.

Destination: Terminal end user or CAFF TD queue.

Modules: CAUCAFF1

CAU2102W Detector is not *state*

Explanation: An attempt was made to Start, Stop, Pause, or Continue the Detector from CAFF. However, the Detector was not currently in an appropriate *state* to make the change.

System Action: The Detector state is not changed.

User Response: Check why the Detector is currently in that state.

Destination: Terminal end user or CAFF TD queue.

Modules: CAUCAFF1

CAU2103W Detector is already STOPPED

Explanation: An attempt was made to Stop the Detector from CAFF when the Detector was already STOPPED.

System Action: The Detector state is not changed.

User Response: Check why the Detector is currently in that state.

Destination: Terminal end user or CAFF TD queue.

Modules: CAUCAFF1

CAU2104W Invalid key pressed

Explanation: The terminal operator has pressed a function key in response to a screen displayed by the CAFF transaction, but the function key was not valid for that screen.

System Action: The function key is ignored.

User Response: Use the correct function key.

Destination: Terminal end user.

Modules: CAUCAFF1, CAUCAFF2, CAUCAFF7

CAU2105I CAFF session ended

Explanation: The transaction CAFF has ended.

System Action: The state of the Detector is unchanged.

User Response: None.

Destination: Terminal end user and CAFF TD queue.

Modules: CAUCAFF1

CAU2106W Options must be Y(Yes) or N(No)

Explanation: The only valid values for this CAFF operation option are 'Y' and 'N'.

System Action: The CAFF operation options will not be changed unless all of the input is correct.

User Response: Correct the invalid input.

Destination: Terminal end user.

Modules: CAUCAFF2

CAU2107W Size must be integer (10 to 2000 Mb)

Explanation: The only valid value for the Detector dataspace size is an integer in the range 10 to 2000. This gives the size in megabytes.

System Action: The CAFF operation options will not be changed unless all of the input is correct.

User Response: Correct the invalid input.

Destination: Terminal end user.

Modules: CAUCAFF2

CAU2110I No amendments were entered

Explanation: The Enter key was pressed on the CAFF operation options screen (CAFF02). However, no changes had been input.

System Action: No options are changed.

User Response: Enter any changes and then press Enter again.

Destination: Terminal end user.

Modules: CAUCAFF2

CAU2111I CAFF options updated

Explanation: The CAFF transaction has successfully amended the operation options.

System Action: The operation options held on VSAM control file CAUCNTL are updated.

User Response: None.

Destination: Terminal end user and CAFF TD queue.

Modules: CAUCAFF2

CAU2114I Records restored

Explanation: The Detector is being started with the restore data option set to Y. Records from the previous Detector run are retained on the affinity data files (CAUAFF1, CAUAFF2, CAUAFF3).

Records for those affinity command types that are being detected on this Detector run were found on the files and were read into the dataspace.

System Action: The Detector is started with any records from a previous run retained on the affinity data files and read into the dataspace.

User Response: None.

Destination: Terminal end user.

Modules: CAUCAFF3

CAU2115I Affinity files emptied

Explanation: The Detector is being started with the restore data option set to N. All existing affinity records were deleted from the affinity data files (CAUAFF1, CAUAFF2, CAUAFF3).

System Action: The Detector is started with empty affinity data files.

User Response: None.

Destination: Terminal end user and CAFF TD queue.

Modules: CAUCAFF3

CAU2116W Dataspace too large - no storage available

Explanation: The Detector is being started and it received a response from the table manager (CAUTABM) that it was unable to obtain the amount of storage requested for the MVS/ESA dataspace, because the MVS Real Storage Manager does not have enough resources.

System Action: The Detector start is aborted and the Detector is stopped.

User Response: Decrease the dataspace size via the CAFF operation options.

Destination: Terminal end user.

Modules: CAUCAFF3

CAU2117W Dataspace too large - IEFUSI limit reached

Explanation: The Detector is being started and it received a response from the table manager (CAUTABM) that it was unable to obtain the amount of storage requested for the MVS/ESA dataspace, because MVS exit IEFUSI has imposed a limit on address space size.

System Action: The Detector start is aborted and the Detector is stopped.

User Response: Decrease the dataspace size via the CAFF operation options or else ask the MVS system programmer to increase the IEFUSI limit.

Destination: Terminal end user.

Modules: CAUCAFF3

CAU2118I No records restored

Explanation: The Detector is being started with the restore data option set to Y. Records from the previous Detector run are retained on the affinity data files (CAUAFF1, CAUAFF2, CAUAFF3).

No records for those affinity command types that are being detected on this Detector run were found on the files, so none were read into the dataspace.

System Action: The Detector is started with any records from a previous run retained on the affinity data files.

User Response: None.

Destination: Terminal end user.

Modules: CAUCAFF3

CAU2119I CICS is terminating

Explanation: The CAFF transaction has ended because it has found that the CICS region has entered quiesce state before shutdown.

System Action: The CAFF transaction is terminated. If the Detector state was other than STOPPED then the CAFB transaction will stop the Detector.

User Response: None.

Destination: Terminal end user and CAFF TD queue.

Modules: CAUCAFF1

CAU2120I F5 to confirm Start with data restore

Explanation: Confirmation is required that the Detector is to be started with the restore data option set to Y.

System Action: If F5 is pressed then the Detector will be started with affinity data retained on the affinity data files and read into the dataspace. Otherwise, if any other key is pressed the Detector will not be started.

User Response: Press F5 to start the Detector.

Destination: Terminal end user.

Modules: CAUCAFF1

CAU2121I F5 to confirm Start without restore

Explanation: Confirmation is required that the Detector is to be started with the restore data option set to N.

System Action: If F5 is pressed then the Detector will be started with all of the data on the affinity data files deleted. Otherwise, if any other key is pressed the Detector will not be started.

User Response: Press F5 to start the Detector.

Destination: Terminal end user.

Modules: CAUCAFF1

CAU2122I F6 to confirm Stop

Explanation: Confirmation is required that the Detector is to be stopped. The affinity data in the dataspace will be saved to the affinity data files.

System Action: If F6 is pressed then the Detector will be stopped and any changes made to the affinity data in the dataspace since the last save will be saved to the affinity data files. If any other key is pressed, the Detector will not be stopped.

User Response: Press F6 to stop the Detector.

Destination: Terminal end user.

Modules: CAUCAFF1

CAU2125E Must supply CAFF action

Explanation: The transid CAFF is being entered at a console device. It is mandatory to supply a Detector action with CAFF at a console device.

System Action: The CAFF transaction is not initiated.

User Response: Ensure that an action (one of START, STOP, PAUSE, CONTINUE) is entered after the transid CAFF.

Destination: Console.

Modules: CAUCAFF1

CAU2127W Transid prefix is invalid

Explanation: The transid prefix input on screen CAFF02 is invalid. The input was rejected because it contained an embedded blank space character.

System Action: The CAFF options will not be changed unless all of the input is correct.

User Response: Correct the invalid input.

Destination: Terminal end user.

Modules: CAUCAFF2

CAU2201S CICS *command data* **failed RESP=***eibresp* **RESP2=***eibresp2* **RCODE=***eibrcode*

Explanation: Transaction CAFF or CAFB received an invalid response when issuing EXEC CICS *command*. The response is in *eibresp*, *eibresp2* and *eibrcode*. Other *data*, if present, may give the object operated on by the *command*.

System Action: The transaction is terminated by abending AUZA and the Detector is stopped.

User Response: For further details of the exception *eibresp* refer to the *command* in the *CICS/ESA Application Programming Reference* manual or the *CICS/ESA System Programming Reference* manual.

For further information on how to determine system problems refer to the *CICS/ESA Problem Determination Guide*. If you are still unable to resolve the problem, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFF1, CAUCAFF2, CAUCAFF3, CAUCAFF4, CAUCAFF5, CAUCAFF6, CAUCAFF7, CAUCAFB1, CAUCAFB2, CAUCAF41

CAU2202S VSAM *filetype* **file** *filename command* **failed RESP=***eibresp* **RESP2=***eibresp2*

Explanation: Transaction CAFF or CAFB received an invalid response when issuing EXEC CICS *command* on VSAM *filetype* file *filename*. The response is in *eibresp* and *eibresp2*.

System Action: The transaction is terminated by abending AUZB and the Detector is stopped.

User Response: For further details of the exception *eibresp* refer to the *command* in the *CICS/ESA Application Programming Reference* manual or the *CICS/ESA System Programming Reference* manual.

For further information on how to determine system problems refer to the *CICS/ESA Problem Determination Guide*. If you are still unable to resolve the problem, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFF1, CAUCAFF2, CAUCAFF3, CAUCAFF4, CAUCAFF5, CAUCAFF6, CAUCAFB2

CAU2203S Internal error determining state in *transaction program*

Explanation: Detector program *program* in transaction *transaction* was unable to determine the Detector state from either the VSAM control file record or from the Detector exit GWA.

System Action: The transaction is terminated by abending AUZC and the Detector is stopped.

User Response: Attempt to re-enter transaction CAFF. If you are still unable to use CAFF, run the VSAM job to delete and define the VSAM control file CAUCNTL. If you are still unable to use CAFF, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFF1, CAUCAFF2

CAU2204I CAFF being used by *userid*

Explanation: User *userid* is using the CAFF transaction.

System Action: This user has exclusive use of the CAFF transaction.

User Response: None.

Destination: CAFF TD queue.

Modules: CAUCAFF1

CAU2205S File *filename* **not for CICS system** *applid*

Explanation: File *filename* was first used by a CICS system with a different CICS specific APPLID, and is therefore not useable by a CICS system with specific APPLID *applid*.

System Action: The transaction is terminated by abending AUZD and the Detector is stopped.

User Response: Ensure that the file *filename* is the correct file for the CICS system in question. If the correct file can not be located, run the VSAM job to delete and define file *filename*. If *filename* was the control file (CAUCNTL), then the CAFF operation options may need to be amended via CAFF, as the file is recreated with default option values. If *filename* was an affinity data file (one of CAUAFF1, CAUAFF2, CAUAFF3), then when the Detector is next started via CAFF, the restore data option must be set to N.

Destination: CAFF TD queue.

Modules: CAUCAFF1, CAUCAFF3

CAU2206S CICS *command* **PROGRAM** *program* **failed RESP=***eibresp* **RCODE=***eibrcode*

Explanation: Transaction CAFF or CAFB received an invalid response when issuing command EXEC CICS *command* for Detector user exit program *program*.

System Action: The transaction is terminated by abending AUZF and the Detector is stopped.

User Response: For further details of the exception *eibresp* refer to the *command* in the *CICS/ESA System Programming Reference* manual.

For further information on how to determine system problems refer to the *CICS/ESA Problem Determination Guide*. If you are still unable to resolve the problem, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFF1, CAUCAFF3, CAUCAFF4, CAUCAFF5, CAUCAFF6, CAUCAFB1

CAU2210S CAUTABM create dataspace failed REASON=*reason code* **ERROR=***error code*

Explanation: The CAFF transaction received an invalid response when issuing a call to the table manager (CAUTABM) to create the MVS/ESA dataspace when starting the Detector.

System Action: The transaction is terminated by abending AUZH and the Detector is stopped.

User Response: The *reason code* value may be looked up in “Detector table manager diagnostics” on page 118. If it is AUTM_DSPSERV_CREATE_ERROR then *error code* is the value of GPR 0 after the MVS DSPSERV CREATE call. If it is AUTM_ALESERV_ADD_ERROR then *error code* is the value of GPR 15 after the MVS ALESERV ADD call. Use the appropriate MVS manual to find out the meaning of the error code.

If you are still unable to resolve the problem, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFF3

CAU2211S CAUTABM create table failed REASON=*reason code* **TABLE=***table number*

Explanation: The CAFF transaction received an invalid response when issuing a create table call to the table manager (CAUTABM) for table *table number*.

System Action: The transaction is terminated by abending AUZI and the Detector is stopped.

User Response: The *reason code* and *table number* values may be looked up in “Detector table manager diagnostics” on page 118. If you are unable to resolve the problem, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFF3, CAUCAFF6

CAU2212S CAUTABM add element error REASON=*reason code* **TABLE=***table number*

Explanation: The transaction CAFF received an invalid response when issuing an add element call to the table manager (CAUTABM) for table *table number*.

System Action: The transaction is terminated by abending AUZJ and the Detector is stopped.

User Response: The *reason code* and *table number* values may be looked up in “Detector table manager diagnostics” on page 118. If you are unable to resolve the problem, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFF3

CAU2214I Detector is now *state*

Explanation: The CAFF transaction has changed the Detector state to *state*.

System Action: The state of the Detector is now *state*.

User Response: None.

Destination: CAFF TD queue.

Modules: CAUCAFF1, CAUCAFF3, CAUCAFF4, CAUCAFF5, CAUCAFF6

CAU2216S CAUTABM destroy pool failed REASON=*reason code* **ERROR=***error code*

Explanation: The transaction CAFF received an invalid response when issuing a call to the table manager (CAUTABM) to destroy the MVS/ESA dataspace when stopping the Detector.

System Action: The transaction is terminated by abending AUZN and the Detector is stopped.

User Response: The *reason code* value may be looked up in “Detector table manager diagnostics” on page 118. If it is AUTM_DSPSERV_DELETE_ERROR then *error code* is the value of GPR 0 after the MVS DSPSERV DELETE call. If it is AUTM_ALESERV_DELETE_ERROR then *error code* is the value of GPR 15 after the MVS ALESERV DELETE call. Use the appropriate MVS manual to find out the meaning of the error code.

If you are still unable to resolve the problem, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFF4

CAU2217S CAUTABM destroy table failed REASON=*reason code* **TABLE=***table number*

Explanation: The transaction CAFF received an invalid response when issuing a destroy table call to the table manager (CAUTABM) for table *table number*.

System Action: The transaction is terminated by abending AUZO and the Detector is stopped.

User Response: The *reason code* and *table number* values may be looked up in “Detector table manager diagnostics” on page 118. If you are unable to resolve the problem, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFF6

CAU2218S CAFF has abended *abend code* **in program** *program*

Explanation: This message is issued when the CAFF HANDLE ABEND exit program is driven to handle a transaction abend that occurred within the CAFF transaction. The abend code is given by *abend code*, and the failing program is given by *program*. Note that abends are issued by CAFF (codes AUxx) as well as by CICS.

System Action: The transaction is terminated with a transaction dump, with a dumpcode of *abend code*, and the Detector is stopped.

User Response: If the original abend was issued by CAFF, then there will be a preceding message on the CAFF TD queue describing the abend. If so, refer to the description for that message. Otherwise, the abend was issued by CICS (eg, ASRA), so refer to the *CICS/ESA Messages and Codes*.

Destination: CAFF TD queue and console.

Modules: CAUCAFFE

CAU2220S CAUCAFP create CPOOL failed REASON=*reason code*

Explanation: The CAFF transaction received an invalid response when issuing a call to the CAFB request queue manager (CAUCAFP) to create its storage in the MVS CPOOL.

System Action: The transaction is terminated by abending AUZQ and the Detector is stopped.

User Response: The *reason code* value may be looked up in “Detector CAFB request queue manager diagnostics” on page 120. Check that there was sufficient storage in your system for at least 4K bytes of MVS CPOOL. If you are unable to resolve the problem, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFF3

CAU2221S CAUCAFP call failed FUNC=*function code* **REASON=***reason code*

Explanation: Transaction CAFF or CAFB received an invalid response when issuing a call to the CAFB request queue manager (CAUCAFP) to perform function *function code*.

System Action: The transaction is terminated by abending AUZR and the Detector is stopped.

User Response: The *reason code* and *function code* values may be looked up in “Detector CAFB request queue manager diagnostics” on page 120. If you are unable to resolve the problem, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFF4, CAUCAFF5, CAUCAFB1

CAU2222S CAUCAFP destroy CPOOL call failed REASON=*reason code*

Explanation: The CAFF transaction received an invalid response when issuing a call to the CAFB request queue manager (CAUCAFP) to destroy its MVS CPOOL storage.

System Action: The transaction is terminated by abending AUZS and the Detector is stopped.

User Response: The *reason code* value may be looked up in “Detector CAFB request queue manager diagnostics” on page 120. If you are unable to resolve the problem, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFF4

CAU2224S Error calculating space utilisation

Explanation: An error occurred during the calculation of the percentage of the dataspace currently occupied by affinity data.

System Action: The transaction is terminated by abending AUZU and the Detector is stopped.

User Response: Contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFF1

CAU2225E Unsupported type of CAFF task initiation

Explanation: The CAFF transaction has been initiated in a way which is not allowed. The only valid ways to initiate a CAFF transaction are:

- From a terminal
- From a console
- By issuing EXEC CICS START TRANSID('CAFF') from another task.

System Action: The transaction is terminated by abending AUZV and the Detector is stopped.

User Response: Use one of the methods above to initiate CAFF.

Destination: CAFF TD queue.

Modules: CAUCAFF1

CAU2226E Incorrect CAFF action

Explanation: The CAFF transaction received an incorrect value for <action> when it was started by another task using EXEC CICS START TRANSID('CAFF') FROM(<action>), or started from a terminal by entering CAFF <action>. The only acceptable <action> values are:

- START
- STOP
- PAUSE
- CONTINUE

System Action: The CAFF transaction is terminated.

User Response: Correct the <action> passed to CAFF to be one of those above.

Destination: CAFF TD queue and terminal end user, if started from a terminal.

Modules: CAUCAFF1

CAU2227I Non-terminal CAFF task initiating

Explanation: Transaction CAFF has been initiated as a background non-terminal task.

System Action: CAFF runs in the background.

User Response: None.

Destination: CAFF TD queue.

Modules: CAUCAFF1

CAU2228S CAUTABM replace element failed REASON=*reason code* **TABLE=***table number*

Explanation: Transaction CAFF or CAFB received an invalid response when issuing a call to the table manager (CAUTABM) to replace a table element for table *table number*.

System Action: The transaction is terminated by abending AUZY and the Detector is stopped.

User Response: The *reason code* and *table number* values may be looked up in “Detector table manager diagnostics” on page 118. If you are unable to resolve the problem, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFF3, CAUCAFB2

CAU2229S UT/TT table update failed FUNC=*function code* **REASON=***reason code*
TABLE=*table number*

Explanation: Transaction CAFF received an invalid response when issuing a call to the table manager (CAUTABM) to update either the userid table (UT) or the termid table (TT).

System Action: The transaction is terminated by abending AUZZ and the Detector is stopped.

User Response: The *reason code*, *table number* and *function code* values may be looked up in “Detector table manager diagnostics” on page 118. If you are unable to resolve the problem, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFF3, CAUCAFF6

CAU2230S VSAM affinity file *filename* **header READ failed RESP=***eibresp*
RESP2=*eibresp2*

Explanation: Transaction CAFF received an invalid response when issuing an EXEC CICS READ command for the header record on VSAM affinity data file *filename*, when the Detector was starting with the restore data option set to Y. Either an incorrect file has been allocated for the affinity data file *filename* or the file is empty.

System Action: The transaction is terminated by abending AUZ1 and the Detector is stopped.

User Response: Check that the correct file is allocated, and that the Detector has previously been started with the same files.

The first time the Detector is started, the CAFF restore data option must be N. If the restore data option is initially set to Y, then CAFF will abend with this message, because initially the VSAM affinity data files will be completely empty. The affinity file header records are added by CAFF after the Detector has been started for the first time.

For further details of the exceptions *eibresp* and *eibresp2* refer to the READ command in the *CICS/ESA Application Programming Reference* manual.

For further information on how to determine system problems refer to the *CICS/ESA Problem Determination Guide*. If you are still unable to resolve the problem, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFF3

CAU2231I Number of records restored = *count*

Explanation: The Detector has been started with the restore option set to Y. The message gives the number of affinity records that were restored from the VSAM affinity data files to the MVS/ESA dataspace.

System Action: The Detector is now RUNNING.

User Response: None.

Destination: CAFF TD queue.

Modules: CAUCAFF3

CAU2232E CICS release *release* **is not supported by the Detector**

Explanation: Transaction CAFF or CAFB has been initiated on a version/release/modification of CICS which the Detector does not support.

System Action: The transaction is terminated by abending AUZ3 and the Detector is stopped.

User Response: The Detector cannot be run on this CICS release.

Destination: CAFF TD queue.

Modules: CAUCAFF1, CAUCAFB1

CAU2233W CAUCAFDT call failed REASON=*reason code*

Explanation: The CAU date formatter program (CAUCAFDT) was unable to format the packed Julian date passed to it by its caller.

System Action: Question marks are used for the date instead.

User Response: The *reason code* value may be looked up in “Date formatter diagnostics” on page 120. If the problem persists, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFF1, CAULMS, CAUREPPM

CAU3301I CAFB task initiating

Explanation: The CAFB transaction has been initiated by CAFF. CAFB saves affinity data from the dataspace to the affinity data files.

System Action: The Detector continues RUNNING.

User Response: None.

Destination: CAFF TD queue.

Modules: CAUCAFB1

CAU3302S CAFB received an unrecognizable request

Explanation: Transaction CAFB received an invalid request to perform one of its functions from another component of the Detector (CAFF or a Detector exit program).

System Action: The transaction is terminated by abending AUYA and the Detector is stopped.

User Response: Contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFB1

CAU3303S CAFB has abended *abend code* **in program** *program*

Explanation: This message is issued when the CAFB HANDLE ABEND exit program is driven to handle a transaction abend that occurred within the CAFB transaction. The abend code is given by *abend code*, and the failing program is given by *program*. Note that abends are issued by CAFB (codes AUxx) as well as by CICS.

System Action: The transaction is terminated with a transaction dump, with a dumpcode of *abend code*, and the Detector is stopped.

User Response: If the original abend was issued by CAFB, then there will be a preceding message on the CAFF TD queue describing the abend. If so, refer to the description for that message. Otherwise, the abend was issued by CICS (eg, ASRA), so refer to the *CICS/ESA Messages and Codes*.

Destination: CAFF TD queue and console.

Modules: CAUCAFBE

CAU3304S CAFB abending - system error elsewhere in Detector

Explanation: The transaction CAFB received a request to abend from another component of the Detector. The request was sent because of an error encountered elsewhere, either in CAFF or a Detector exit program.

System Action: The transaction is terminated by abending AUYC and the Detector is stopped.

User Response: Refer to any earlier messages on the CAFF TD queue for the cause of the error.

Destination: CAFF TD queue.

Modules: CAUCAFB1

CAU3305I CAFB save started - because of *reason*

Explanation: Transaction CAFB is commencing a scan of the affinity tables in the dataspace to write any data changed since the previous save to the affinity data files. The save may have been started for one of four possible *reasons*:

- Detector stopped (STOP)
- Save interval reached (TIME)
- Activity count reached (TRIGGER)
- Detector paused (PAUSE)

Note that the latter three reasons can only occur when the CAFF perform periodic saves option is set to Y.

System Action: CAFB saves changed data elements from the dataspace to the affinity data files.

User Response: None.

Destination: CAFF TD queue.

Modules: CAUCAFB2

CAU3306I CAFB save ended - *count* **records saved**

Explanation: Transaction CAFB has finished the scan of the affinity tables in the dataspace and wrote *count* records to the affinity data files.

System Action: The number of records given by *count* were saved to the affinity data files.

User Response: If the CAFF perform periodic saves option is set to Y, and *count* has been consistently near zero for the past few saves, this may indicate that the Detector has detected all the affinities it can and the Detector may be stopped.

Destination: CAFF TD queue.

Modules: CAUCAFB2

CAU3307I CAFB terminated - Detector stopping

Explanation: Transaction CAFB received a request to terminate, from CAFF, because the Detector is stopping.

System Action: The transaction is terminated and the Detector is stopped.

User Response: None.

Destination: CAFF TD queue.

Modules: CAUCAFB1

CAU3308I Message received from program *program*

Explanation: Transaction CAFB received a message from program *program* to write to CAFF TD queue.

System Action: The associated message is written to CAFF. This is the only mechanism available to the Detector exit programs when they wish to issue a message.

User Response: Examine the following message on the CAFF TD queue.

Destination: CAFF TD queue.

Modules: CAUCAFB1

CAU3310S Invalid file number for table in GWA

Explanation: Transaction CAFB found an incorrect value (the affinity file number) in an internal array in the Detector GWA. This suggests that the GWA has been corrupted.

System Action: The transaction is terminated by abending AUYE and the Detector is stopped.

User Response: Attempt to find out the cause of the corruption. It could be due to an application accidentally overwriting the GWA. If you are unable to resolve the problem, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFB2, CAUCAFF3

CAU3311E Transaction CAFB must be initiated by transaction CAFF

Explanation: Transaction CAFB could not have been initiated by CAFF, as its CICS startcode indicates something other than EXEC CICS START with no data.

System Action: The transaction is terminated by abending AUYF and the Detector is stopped.

User Response: The CAFB transaction can only be started by the Detector control transaction CAFF.

Destination: CAFF TD queue.

Modules: CAUCAFB1

CAU3312S CAFB abending - CICS is terminating

Explanation: Transaction CAFB found that CICS had entered quiesce state before shutdown and the Detector was still RUNNING.

System Action: The transaction is terminated by abending AUYG and the Detector is stopped.

User Response: None. To avoid this, always stop the Detector before shutting down CICS. The CAFF STOP action could be submitted from a CICS Shutdown PLT program, via EXEC CICS START.

Destination: CAFF TD queue.

Modules: CAUCAFB1

CAU3313S Invalid address for *program* **in the GWA**

Explanation: Transaction CAFF or CAFB found that the address of Detector program *program* in the GWA was a null value. This suggests that the GWA has been corrupted.

System Action: The transaction is terminated by abending AUYH and the Detector is stopped.

User Response: Attempt to find out the cause of the corruption. It could be due to an application accidentally overwriting the GWA. If you are unable to resolve the problem, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFF4, CAUCAFF5, CAUCAFB1

CAU3314S CAUTABM call failed FUNC=*function code* **REASON=***reason code* **TABLE=***table number*

Explanation: Transaction CAFF or CAFB received an invalid response when issuing a call to the table manager (CAUTABM) to access a table element for table *table number*.

System Action: The transaction is terminated by abending AUYI and the Detector is stopped.

User Response: The *reason code*, *table number* and *function code* values may be looked up in “Detector table manager diagnostics” on page 118. If you are unable to resolve the problem, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFB2, CAUCAFF6

CAU3315S File *filename* **full**

Explanation: Transaction CAFB received a NOSPACE response when issuing EXEC CICS WRITE for VSAM affinity file *filename*. The file has filled up.

System Action: The transaction is terminated by abending AUYJ and the Detector is stopped.

User Response: Allocate more space to the file and rerun the Detector. For guidance on how much space to allocate refer to “Estimating the size of the MVS data space and files” on page 20.

Destination: CAFF TD queue.

Modules: CAUCAFB2

CAU4100S CAUTABM call failed FUNC=*function code* **REASON=***reason code* **TABLE=***table number*

Explanation: One of the Detector exit programs received an unexpected response when issuing a call to the table manager (CAUTABM).

Note that the message is issued by transaction CAFB on behalf of the exit program.

System Action: The CAFB transaction is terminated with abend code AUXA and the Detector is stopped.

User Response: The *reason code*, *table number* and *function code* values may be looked up in “Detector table manager diagnostics” on page 118. If you are unable to resolve the problem, contact IBM support.

Destination: CAFF TD queue.

Modules: CAUCAFB1

CAU4200S Dataspace full

Explanation: Transaction CAFF or a Detector exit program received a reason code of AUTM_NO_STORAGE when issuing a call to the table manager (CAUTABM) to either create a new table or add an element to a table. The dataspace has filled up.

Note that if this situation was encountered by a Detector exit program, the message is issued by transaction CAFB on its behalf.

System Action: The CAFF or CAFB transaction is terminated by abending AUXB and the Detector is stopped.

User Response: Increase the dataspace size via the CAFF options and rerun the Detector. For guidance on how much space to allocate refer to section “Estimating the size of the MVS data space and files” on page 20.

Destination: CAFF TD queue.

Modules: CAUCAFF3, CAUCAFF6, CAUCAFB1

CAU5000S Function call *num* **invalid for** *module*

Explanation: A module of the Reporter or Builder has been called with an invalid function number. This indicates an internal logic error in the Reporter or Builder.

System Action: The Reporter or Builder is terminated.

User Response: Contact IBM support.

Destination: Console.

Modules: CAUREPFM, CAUREPPM, CAUBLD

CAU5001E *filename* **does not contain a control record. Run aborted**

Explanation: If *filename* is CAUCNTL, then it does not contain the expected control record. If *filename* is one of CAUAFF1, CAUAFF2 or CAUAFF3, then it has not been initialized with a header record or one or more of the files are not affinity data files.

System Action: The Reporter is terminated.

User Response: Check that the correct files are being input into the Reporter. If the files are correct check that the Detector has been run at least once. These files are initialized by the Detector the first time it is run.

Destination: Console.

Modules: CAUREP

CAU5002E Affinity files not for same CICS APPLID. Run aborted

Explanation: Either one or more of CAUCNTL, CAUAFF1, CAUAFF2 or CAUAFF3 were not initialized by the same CICS system.

System Action: The Reporter is terminated.

User Response: The most likely cause of this error is one or more incorrect filenames in the JCL to run the Reporter. Correct the problem and rerun the job.

Destination: Console.

Modules: CAUREP

CAU5003E Unable to open *filename* **RC=***return code***, REASON=***reason code*

Explanation: The Reporter, Builder or Scanner has been unable to open *filename*.

System Action: The Reporter, Builder, or Scanner is terminated.

User Response: If the codes are zero then it was a non-VSAM file that failed to open.

Otherwise the *return code* is as returned by VSAM in GPR 15, and the *reason code* is the result of a subsequent SHOWCB ACB FIELDS=(ERROR) macro call. Check these in the VSAM messages and codes manual to determine the cause of the error.

The most likely cause of this message will be missing or incorrect filenames in the JCL to run the job. Correct the problem and rerun the job.

Destination: Console.

Modules: CAUREPFM, CAULMS

CAU5004E GENCB failed for *filename* **CB=***control block* **RC=***return code* **REASON=***reason code*

Explanation: The generation of a VSAM *control block* failed for Reporter file *filename*.

System Action: The Reporter is terminated.

User Response: The *return code* and *reason code* are as returned by VSAM in GPR 15 and GPR 0 respectively. Check the VSAM messages and codes manual to determine the cause of the error. Correct the problem and rerun the job. If the problem persists, contact IBM support.

Destination: Console.

Modules: CAUREPFM

CAU5005S File number *filenum* **invalid**

Explanation: The file manager module (CAUREPFM) used by the Reporter and the Builder has been called with an invalid file number *filenum*.

System Action: The Reporter or Builder is terminated.

User Response: Rerun the job. If the problem persists, contact IBM support.

Destination: Console.

Modules: CAUREPFM

CAU5006S Attempting to *action filename***. File is** *type*

Explanation: The Reporter or Builder attempted to read from the output file or write to the input file specified by *filename*.

System Action: The Reporter or Builder is terminated.

User Response: Rerun the job. If the problem persists, contact IBM support.

Destination: Console.

Modules: CAUREPFM

CAU5007S RPL number *rplnum* **invalid for** *filename*

Explanation: The RPL number *rplnum* is invalid for Reporter file *filename*.

System Action: The Reporter is terminated.

User Response: Rerun the job. If the problem persists, contact IBM support.

Destination: Console.

Modules: CAUREPFM

CAU5008S Table number *table number* **invalid**

Explanation: A request by the Reporter to read a table from the affinity data files was being processed but *table number* was not valid.

System Action: The Reporter is terminated.

User Response: Rerun the job. If the problem persists, contact IBM support.

Destination: Console.

Modules: CAUREPFM

CAU5009E *command* **failed for** *filename* **RPL=***rplnum* **RC=***return code* **REASON=***reason code*

Explanation: The VSAM *command* has failed to execute for Reporter file *filename*.

System Action: The Reporter is terminated.

User Response: The *return code* is as returned by VSAM in GPR 15, and the *reason code* is the result of a subsequent SHOWCB RPL FIELDS=(FDBK) macro call. Check these in the VSAM messages and codes manual, correct the problem and rerun the job. If the problem persists, contact IBM support.

Destination: Console.

Modules: CAUREPFM

CAU5010S Attempting to position a non-VSAM file

Explanation: The Reporter has attempted file positioning on a non-VSAM file. This indicates a potential internal logic error in the Reporter.

System Action: The Reporter is terminated.

User Response: Rerun the job. If the problem persists, contact IBM support.

Destination: Console.

Modules: CAUREPFM

CAU5011W No transaction entries for trangroup *trangroup***. Report incomplete**

Explanation: The Reporter could find no transaction entries in the affinity data files for transaction group *trangroup*. This means, for example, that a record was found for a named temporary storage queue, but no records were found detailing the transactions that used that queue. This will probably be caused by a Detector abend or a CICS crash occurring in the middle of a Detector save, so the data saved to the affinity files is only partly complete. It could also be caused by incorrect files.

System Action: The Reporter continues.

User Response: Check the validity of the affinity files as far as is possible. For instance check that the correct set of files is being input and that one is not from a different run of the Detector. Ascertain if the Detector did indeed abend or if CICS crashed; this will be indicated by message CAU5035W appearing at the start of the Reporter output. If CAU5011W occurs frequently, the Detector should be rerun to produce a more complete set of data.

Destination: Console and report.

Modules: CAUREPRM

CAU5012I Invalid PARM specified. $SUMMARY assumed

Explanation: When invoking the Scanner, a PARM field has been specified on the EXEC that does not contain valid information.

System Action: The Scanner uses the default PARM of $SUMMARY.

User Response: Correct the PARM information and rerun if required.

Destination: Console.

Modules: CAULMS

CAU5013S *macro* **failed. Return code=***return code*

Explanation: A Scanner or Builder MVS GETMAIN or MVS FREEMAIN macro has failed to execute successfully.

System Action: The Scanner or Builder is terminated.

User Response: Check the appropriate MVS manual for the meaning of *return code*, which is the value of GPR 15 after the macro call. Correct the problem and rerun the job. If the problem persists, contact IBM support.

Destination: Console.

Modules: CAULMS, CAUBLD, CAUBLDMR

CAU5014E Invalid PARM keyword specified. Correct and rerun

Explanation: When invoking the Builder, a PARM field has been specified on the EXEC that contains an invalid keyword. The allowable keywords are MATCH, STATE, CONTEXT, and DSPSIZE.

System Action: The Builder is terminated.

User Response: Correct the PARM information and rerun the job.

Destination: Console.

Modules: CAUBLD

CAU5015E Invalid *keyword* **value specified. Correct and rerun**

Explanation: When invoking the Builder, a PARM field has been specified on the EXEC that contains an invalid value for *keyword*.

Keyword	Allowed values
MATCH	LUNAME and USERID
STATE	ACTIVE and DORMANT
CONTEXT	plexname, one through eight characters

System Action: The Builder is terminated.

User Response: Correct the PARM information and rerun the job.

Destination: Console.

Modules: CAUBLD

CAU5016E DSPSIZE value is not numeric. Correct and rerun

Explanation: When invoking the Builder, a PARM field has been specified on the EXEC that contains an invalid value for keyword DSPSIZE. A character other than a digit in the range 0..9 was encountered. The value must be an integer in the range 2 to 2000 Mb.

System Action: The Builder is terminated.

User Response: Correct the PARM information and rerun the job.

Destination: Console.

Modules: CAUBLD

CAU5017E DSPSIZE is invalid. Must be between 2 and 2000

Explanation: When invoking the Builder, a PARM field has been specified on the EXEC that contains an invalid value for keyword DSPSIZE. The value must be an integer in the range 2 to 2000 Mb.

System Action: The Builder is terminated.

User Response: Correct the PARM information and rerun the job.

Destination: Console.

Modules: CAUBLD

CAU5018E Load of CAUTABM has failed AC=*abend code* **RC=***reason code*

Explanation: The Builder attempted to load the table manager module (CAUTABM) but the MVS LOAD macro failed. The *abend code* is that returned in GPR 1 and is the abend code that would have resulted had the task abended. The *reason code* is that returned in GPR 15 and is the reason code associated with the abend.

System Action: The Builder is terminated.

User Response: The MVS *abend code* and *reason code* should indicate the cause of the error. Correct the problem and rerun the job. If the problem persists, contact IBM support.

Destination: Console.

Modules: CAUBLD

CAU5019E Dataspace too large - no storage available

Explanation: The Builder received a response from the table manager (CAUTABM) that it was unable to obtain the amount of storage requested for the MVS/ESA dataspace, because the MVS Real Storage Manager does not have enough resources.

System Action: The Builder is terminated.

User Response: Decrease the dataspace size specified on the PARM field of the EXEC statement in the job, and rerun the job.

Destination: Console.

Modules: CAUBLD

CAU5020E Dataspace too large - IEFUSI limit reached

Explanation: The Builder received a response from the table manager (CAUTABM) that it was unable to obtain the amount of storage requested for the MVS/ESA dataspace, because MVS exit IEFUSI has imposed a limit on address space size.

System Action: The Builder is terminated.

User Response: Decrease the dataspace size specified on the PARM field of the EXEC statement in the job, or else ask the MVS system programmer to increase the IEFUSI limit. Rerun the job.

Destination: Console.

Modules: CAUBLD

CAU5021E CAUTABM create dataspace failed REASON=*reason code* **ERROR=***error code*

Explanation: The Builder received an invalid response when issuing a call to the table manager (CAUTABM) to create the MVS/ESA dataspace.

System Action: The Builder is terminated.

User Response: The *reason code* value may be looked up in “Detector table manager diagnostics” on page 118. If it is AUTM_DSPSERV_CREATE_ERROR then *error code* is the value of GPR 0 after the MVS DSPSERV CREATE call. If it is AUTM_ALESERV_ADD_ERROR then *error code* is the value of GPR 15 after the MVS ALESERV ADD call. Use the appropriate MVS manual to find out the meaning of the error code.

If you are still unable to resolve the problem, contact IBM support.

Destination: Console.

Modules: CAUBLD

CAU5022E CAUTABM create table failed REASON=*reason code* **TABLE=***table number*

Explanation: The Builder received an invalid response when issuing a create table call to the table manager (CAUTABM) for table *table number*.

System Action: The Builder is terminated.

User Response: The *reason code* and *table number* values may be looked up in “Detector table manager diagnostics” on page 118. If you are unable to resolve the problem, contact IBM support.

Destination: Console.

Modules: CAUBLD

CAU5023E Keyword *keyword* **is invalid or unexpected**

Explanation: When reading statements from its REPGRPS input stream the Builder encountered a keyword it did not recognise, or a keyword that occurred in an unexpected place.

System Action: The Builder skips to the next input statement and continues, but will terminate when the end of the input is encountered.

User Response: Refer to “Syntax for input to the Builder” on page 60 to correct the statement, and rerun the job. A likely cause of this message is the omission of the terminating semicolon from the preceding statement.

Destination: Error report.

Modules: CAUBLDIN

CAU5024E Keyword *keyword* is missing

Explanation: When reading statements from its REPGRPS input stream the Builder encountered a statement where required keyword *keyword* was missing.

System Action: The Builder skips to the next input statement and continues, but will terminate when the end of the input is encountered.

User Response: Refer to “Syntax for input to the Builder” on page 60 to correct the statement, and rerun the job.

Destination: Error report.

Modules: CAUBLDIN

CAU5025E A value of *value* is invalid for keyword *keyword*

Explanation: When reading statements from its REPGRPS input stream the Builder encountered an invalid *value* for keyword *keyword*.

System Action: The Builder skips to the next keyword and continues, but will terminate when the end of the input is encountered.

User Response: Refer to “Syntax for input to the Builder” on page 60 to correct the statement, and rerun the job.

Destination: Error report.

Modules: CAUBLDIN

CAU5026E Invalid CREATE type

Explanation: When reading statements from its REPGRPS input stream the Builder encountered a CREATE statement. The only allowable keywords that may immediately follow the keyword CREATE are TRANGRP and DTRINGRP.

System Action: The Builder skips to the next input statement and continues, but will terminate when the end of the input is encountered.

User Response: Refer to “Syntax for input to the Builder” on page 60 to correct the statement, and rerun the job.

Destination: Error report.

Modules: CAUBLDIN

CAU5027E Incorrect number of brackets

Explanation: When reading statements from its REPGRPS input stream the Builder encountered a keyword for which a corresponding value was expected. Either the value did not start with a bracket, or did not end with a bracket, or contained a bracket in the middle, or spanned input lines.

System Action: The Builder skips to the next input statement and continues, but will terminate when the end of the input is encountered.

User Response: Refer to “Syntax for input to the Builder” on page 60 to correct the statement, and rerun the job. Values must start and end with a bracket, must not contain other brackets, and must not span lines.

Destination: Error report.

Modules: CAUBLDIN

CAU5028E Missing semicolon

Explanation: When reading statements from its REPGRPS input stream the Builder encountered the end of the input in the middle of a statement, implying that a terminating semicolon is missing from the last statement.

System Action: The Builder is terminated.

User Response: Refer to “Syntax for input to the Builder” on page 60 to correct the statement, and rerun the job.

Destination: Error report.

Modules: CAUBLDIN

CAU5029E Keyword *keyword* **is duplicated**

Explanation: When reading statements from its REPGRPS input stream the Builder encountered a keyword that occurred more than once in the same statement. Duplicate keywords are not allowed.

System Action: The Builder continues, but will terminate when the end of the input is encountered.

User Response: Refer to “Syntax for input to the Builder” on page 60 to correct the statement, and rerun the job.

Destination: Error report.

Modules: CAUBLDIN

CAU5030E No valid statements in REPGRPS - processing terminated

Explanation: When reading statements from its REPGRPS input stream the Builder did not find a single complete statement.

System Action: The Builder is terminated.

User Response: The most likely cause of this message is an empty input file. Correct the problem and rerun the job.

Destination: Console.

Modules: CAUBLDMR

CAU5031E CAUTABM error *function* **element TABLE=***table number* **REASON=***reason code* **MODULE=***progname*

Explanation: Builder module *progname* received an unexpected response when issuing a call to the table manager (CAUTABM).

System Action: The Builder is terminated.

User Response: The *function* gives the operation being performed. The *reason code* and *table number* may be looked up in “Detector table manager diagnostics” on page 118. If you are unable to resolve the problem, contact IBM support.

Destination: Console.

Modules: CAUBLDMR, CAUBLDOT

CAU5032E No HEADER record found - statement ignored

Explanation: When reading statements from its REPGRPS input stream the Builder encountered a CREATE statement. However, no HEADER statement had been encountered first. The HEADER statement is compulsory, and must be the first statement in each dataset in the REPGRPS concatenation.

System Action: The Builder skips to the next input statement and continues, but will terminate when the end of the input is encountered.

User Response: Ensure that each dataset in the REPGRPS concatenation has a HEADER statement as the first statement in the dataset. Correct the problem and rerun the job.

Destination: Error report.

Modules: CAUBLDMR

CAU5033E Duplicate TRANGRP name

Explanation: When reading statements from its REPGRPS input stream the Builder encountered a CREATE TRANGRP statement. However, the trangroup name supplied in the statement is not unique within the current input dataset. Duplicate trangroup names are not allowed.

System Action: The Builder skips to the next input statement and continues, but will terminate when the end of the input is encountered.

User Response: Ensure that each CREATE TRANGRP statement within the dataset specifies a unique trangroup name. If this is already the case, then the probable cause of this message is that the HEADER statement is missing from the dataset. Correct the problem and rerun the job.

Destination: Error report.

Modules: CAUBLDMR

CAU5034E Trangroup has not been previously created

Explanation: When reading statements from its REPGRPS input stream the Builder encountered a CREATE DTRINGRP statement. However, no corresponding valid CREATE TRANGRP statement for the trangroup in question had been encountered.

System Action: The Builder skips to the next input statement and continues, but will terminate when the end of the input is encountered.

User Response: Either the CREATE TRANGRP statement was missing or was in error in some way. Correct the problem and rerun the job.

Destination: Error report.

Modules: CAUBLDMR

CAU5035W Affinity data may be incomplete because of *abend type* **abend**

Explanation: The control record on affinity control file CAUCNTL indicates that the Detector did not stop cleanly. Either CICS crashed or the Detector abended, as indicated by *abend type*. If the termination occurred during a Detector save, it is possible that the data on the affinity files may be incomplete.

System Action: The Reporter continues.

User Response: Incomplete data may be indicated by frequent appearance of message CAU5011W. If so, the Detector should be rerun to produce a more complete set of data.

Destination: Console.

Modules: CAUREP

CAU5036E Dataspace full

Explanation: A Builder module received a reason code of AUTM_NO_STORAGE when issuing a call to the table manager (CAUTABM) to add an element to a table. The dataspace has filled up.

System Action: The Builder is terminated.

User Response: Increase the dataspace size specified on the PARM field of the EXEC statement in the job, and rerun the job.

Destination: Console.

Modules: CAUBLDMR

CAU5037E No valid transids in REPGRPS - processing terminated

Explanation: When reading statements from its REPGRPS input stream the Builder did not find a single valid CREATE DTRINGRP statement.

System Action: The Builder is terminated.

User Response: The most likely cause of this message is an input stream that contains valid CREATE TRANGRP statements, but no CREATE DTRINGRP statements. Correct the problem and rerun the job.

Destination: Console.

Modules: CAUBLDMR

CAU5038E Invalid AFFLIFE for AFFINITY

Explanation: When reading statements from its REPGRPS input stream the Builder encountered a CREATE TRANGRP statement. However, the value specified for AFFLIFE is not one of those permitted given the value specified for AFFINITY.

System Action: The Builder skips to the next input statement and continues, but will terminate when the end of the input is encountered.

User Response: Refer to “Syntax for input to the Builder” on page 60 to correct the statement, and rerun the job.

Destination: Error report.

Modules: CAUBLDMR

CAU5039E PARM keyword is duplicated. Correct and rerun

Explanation: When invoking the Builder, a PARM field has been specified on the EXEC that contains a duplicate keyword. Duplicate keywords are not allowed.

System Action: The Builder is terminated.

User Response: Correct the PARM information and rerun the job.

Destination: Console.

Modules: CAUBLD

CAU5040E Invalid REMOVE type

Explanation: When reading statements from its REPGRPS input stream the Builder encountered a REMOVE statement. The only allowable keyword that may immediately follow the keyword REMOVE is TRANGRP.

System Action: The Builder skips to the next input statement and continues, but will terminate when the end of the input is encountered.

User Response: Refer to “Syntax for input to the Builder” on page 60 to correct the statement, and rerun the job. Alternatively comment out the statement.

Destination: Error report.

Modules: CAUBLDIN

AUXA

Explanation: An unexpected error occurred when calling Detector program CAUTABM, from one of the Detector exit programs. Transaction CAFB issues this abend on behalf of the exit program.

System Action: The Detector is stopped.

User Response: Refer to message CAU4100S.

Modules: CAUCAFB1

AUXB

Explanation: The dataspace has filled up. If the situation was detected by a Detector exit program, transaction CAFB issues this abend on its behalf.

System Action: The Detector is stopped.

User Response: Refer to message CAU4200S.

Modules: CAUCAFF3, CAUCAFF6, CAUCAFB1

AUYA

Explanation: Transaction CAFB received an unrecognisable request from another Detector component (CAFF or a Detector exit program).

System Action: The Detector is stopped.

User Response: Refer to message CAU3302S.

Modules: CAUCAFB1

AUYC

Explanation: Transaction CAFB received a request from another Detector component (CAFF or an exit program) to abend because of an unexpected error.

System Action: The Detector is stopped.

User Response: Refer to message CAU3304S.

Modules: CAUCAFB1

AUYE

Explanation: A Detector program found an invalid affinity file number in an internal array in the Detector GWA.

System Action: The Detector is stopped.

User Response: Refer to message CAU3310S.

Modules: CAUCAFB2, CAUCAFF3

AUYF

Explanation: Transaction CAFB was not started by transaction CAFF.

System Action: The Detector is stopped.

User Response: Refer to message CAU3311S.

Modules: CAUCAFB1

AUYG

Explanation: Transaction CAFB was still running at CICS termination.

System Action: The Detector is stopped.

User Response: Refer to message CAU3312S.

Modules: CAUCAFB1

AUYH

Explanation: A Detector program found that the address held in the Detector GWA for one of the Detector internal modules was invalid.

System Action: The Detector is stopped.

User Response: Refer to message CAU3313S.

Modules: CAUCAFF4, CAUCAFF5, CAUCAFB1

AUYI

Explanation: An unexpected error occurred when calling Detector program CAUTABM to access affinity table data in the dataspace, from transaction CAFF or CAFB.

System Action: The Detector is stopped.

User Response: Refer to message CAU3314S.

Modules: CAUCAFB2, CAUCAFF6

AUYJ

Explanation: One of the affinity data files has filled up.

System Action: The Detector is stopped.

User Response: Refer to message CAU3315S.

Modules: CAUCAFB2

AUZA

Explanation: An unexpected error occurred when issuing an EXEC CICS command, by a program from transaction CAFF or CAFB.

System Action: The Detector is stopped.

User Response: Refer to message CAU2201S.

Modules: CAUCAFF1, CAUCAFF2, CAUCAFF3, CAUCAFF4, CAUCAFF5, CAUCAFF6, CAUCAFF7, CAUCAFB1, CAUCAFB2, CAUCAF41

AUZB

Explanation: An unexpected error error occurred when issuing a VSAM file control EXEC CICS command, by a program from transaction CAFF or CAFB.

System Action: The Detector is stopped.

User Response: Refer to message CAU2202S.

Modules: CAUCAFF1, CAUCAFF2, CAUCAFF3, CAUCAFF4, CAUCAFF5, CAUCAFF6, CAUCAFB1, CAUCAFB2

AUZC

Explanation: The internal field holding the Detector state has an invalid value.

System Action: The Detector is stopped.

User Response: Refer to message CAU2203S.

Modules: CAUCAFF1, CAUCAFF2

AUZD

Explanation: One of the files contains a CICS APPLID that does not match the APPLID of the CICS system.

System Action: The Detector is stopped.

User Response: Refer to message CAU2205S.

Modules: CAUCAFF1, CAUCAFF2

AUZF

Explanation: An unexpected error error occurred when issuing a Detector user exit related EXEC CICS command, by a program from transaction CAFF or CAFB. The command is one of ENABLE, DISABLE or EXTRACT EXIT.

System Action: The Detector is stopped.

User Response: Refer to message CAU2206S.

Modules: CAUCAFF1, CAUCAFF2, CAUCAFF3, CAUCAFF4, CAUCAFF5, CAUCAFF6, CAUCAFB1

AUZH

Explanation: An unexpected error occurred when calling Detector program CAUTABM to create the MVS/ESA dataspace to hold the affinity data, from transaction CAFF.

System Action: The Detector is stopped.

User Response: Refer to message CAU2210S.

Modules: CAUCAFF3

AUZI

Explanation: An unexpected error occurred when calling Detector program CAUTABM to create an affinity table in the dataspace, from transaction CAFF.

System Action: The Detector is stopped.

User Response: Refer to message CAU2211S.

Modules: CAUCAFF3, CAUCAFF6

AUZJ

Explanation: An unexpected error occurred when calling Detector program CAUTABM to add an element to an affinity table in the dataspace, from transaction CAFF.

System Action: The Detector is stopped.

User Response: Refer to message CAU2212S.

Modules: CAUCAFF3

AUZN

Explanation: An unexpected error occurred when calling Detector program CAUTABM to destroy the dataspace, from transaction CAFF.

System Action: The Detector is stopped.

User Response: Refer to message CAU2216S.

Modules: CAUCAFF4

AUZO

Explanation: An unexpected error occurred when calling Detector program CAUTABM to destroy a table in the dataspace, from transaction CAFF.

System Action: The Detector is stopped.

User Response: Refer to message CAU2217S.

Modules: CAUCAFF6

AUZQ

Explanation: An unexpected error occurred when calling Detector program CAUCAFP to create its MVS CPOOL storage, from transaction CAFF.

System Action: The Detector is stopped.

User Response: Refer to message CAU2220S.

Modules: CAUCAFF3

AUZR

Explanation: An unexpected error occurred when calling Detector program CAUCAFP to access its MVS CPOOL storage, from transaction CAFF or CAFB.

System Action: The Detector is stopped.

User Response: Refer to message CAU2221S.

Modules: CAUCAFF4, CAUCAFF5, CAUCAFB1

AUZS

Explanation: An unexpected error occurred when calling Detector program CAUCAFP to destroy its MVS CPOOL storage, from transaction CAFF.

System Action: The Detector is stopped.

User Response: Refer to message CAU2222S.

Modules: CAUCAFF4

AUZU

Explanation: An unexpected error occurred when calculating what percentage of the dataspace is occupied by affinity data, from transaction CAFF.

System Action: The Detector is stopped.

User Response: Refer to message CAU2224S.

Modules: CAUCAFF1

AUZV

Explanation: The method of initiating transaction CAFF is incorrect.

System Action: The Detector is stopped.

User Response: Refer to message CAU2225E.

Modules: CAUCAFF1

AUZY

Explanation: An unexpected error occurred when calling Detector program CAUTABM to replace a table element in the dataspace, from transaction CAFF or CAFB.

System Action: The Detector is stopped.

User Response: Refer to message CAU2228S.

Modules: CAUCAFF3, CAUCAFB2

AUZZ

Explanation: An unexpected error occurred when calling a Detector subroutine to update the termid table (TT) or userid table (UT), from transaction CAFF.

System Action: The Detector is stopped.

User Response: Refer to message CAU2229S.

Modules: CAUCAFF3, CAUCAFF6

AUZ1

Explanation: When the Detector was being started by transaction CAFF, the header record could not be found on one of the VSAM affinity data files.

System Action: The Detector is stopped.

User Response: Refer to message CAU2230S.

Modules: CAUCAFF3

AUZ3

Explanation: Transaction CAFF or CAFB is running on a release of CICS which does not support the Detector.

System Action: The Detector is stopped.

User Response: Refer to message CAU2232E.

Modules: CAUCAFF1, CAUCAFB1

Detector table manager diagnostics

This section

Lists the meaning for each possible value of the call parameters that are included in the error messages issued if an an error occurs on a call to the Detector and Builder table manager, CAUTABM.

Function code values

AUTM_CREATE_POOL	1
AUTM_DESTROY_POOL	2
AUTM_CREATE_TABLE	3
AUTM_DESTROY_TABLE	4
AUTM_ADD_ELEMENT	5
AUTM_DELETE_ELEMENT	6
AUTM_REPLACE_ELEMENT	7
AUTM_GET_KEY_ELEMENT	8
AUTM_GET_FIRST_ELEMENT	9
AUTM_GET_NEXT_ELEMENT	10
AUTM_GET_ELEMENT	11
AUTM_GET_KEY_GE_ELEMENT	12

Table identifier values

AUTM_EDSR	1
AUTM_EDST	2
AUTM_EDR	3
AUTM_EDT	4
AUTM_TSQ	5
AUTM_TST	6
AUTM_LRP	7
AUTM_LRT	8
AUTM_SRS	9
AUTM_SRT	10
AUTM_CWA	11
AUTM_CWT	12
AUTM_GFA	13
AUTM_GFM	14
AUTM_LFA	15
AUTM_LFM	16
AUTM_ICR	17
AUTM_ICM	18
AUTM_SPI	19
AUTM_WAIT	20
AUTM_TT	24
AUTM_UT	25
AUTM_BLD_DNT	28
AUTM_BLD_GNT	29
AUTM_BLD_TT	30
AUTM_BLD_MERGED	31

Reason code values

AUTM_INVALID_FUNCTION	0
AUTM_NO_STORAGE	1
AUTM_ELEMENT_NOT_FOUND	2
AUTM_ELEMENT_EXISTS	3
AUTM_INVALID_TABLE	4
AUTM_IEFUSI_HIT	5
AUTM_TABLE_EXISTS	6
AUTM_TABLE_DOES_NOT_EXIST	7
AUTM_POOL_EXISTS	8
AUTM_POOL_DOES_NOT_EXIST	9
AUTM_INVALID_CURSOR	10
AUTM_DEFAULT_SIFD_ERROR	192
AUTM_DEFAULT_SIFA_ERROR	193
AUTM_DEFAULT_DSP_ERROR	194
AUTM_DEFAULT_AVL_ERROR	195
AUTM_SIFD_CREATE_POOL_ERROR	196
AUTM_SIFA_CREATE_POOL_ERROR	197
AUTM_DSP_CREATE_POOL_ERROR	198
AUTM_SIFD_DESTROY_POOL_ERROR	199
AUTM_SIFA_DESTROY_POOL_ERROR	200
AUTM_DSP_DESTROY_POOL_ERROR	201
AUTM_AVL_CREATE_TABLE_ERROR	202
AUTM_SIFA_CREATE_TABLE_ERROR	203
AUTM_AVL_DESTROY_TABLE_ERROR	204
AUTM_SIFA_DESTROY_TABLE_ERROR	205
AUTM_AVL_ADD_ERROR	206
AUTM_SIFA_ADD_ERROR	207
AUTM_AVL_GET_KEY_ERROR	208
AUTM_AVL_GET_FIRST_ERROR	209
AUTM_AVL_GET_NEXT_ERROR	210
AUTM_AVL_DELETE_ERROR	211
AUTM_SIFA_DELETE_ERROR	212
AUTM_AVL_REPLACE_ERROR	213
AUTM_DSP_RESERVE_ERROR	214
AUTM_DSP_RELEASE_ERROR	215
AUTM_DSPSERV_CREATE_ERROR	216
AUTM_DSPSERV_DELETE_ERROR	217
AUTM_ALESERV_ADD_ERROR	218
AUTM_ALESERV_DELETE_ERROR	219
AUTM_AVL_GET_ERROR	220

Detector CAFB request queue manager diagnostics

This section

Lists the meaning for each possible value of the call parameters that are included in the error messages issued if an an error occurs on a call to the Detector CAFB request queue manager, CAUCAFP.

Function code values

AUCP_ADD_CELL_FIRST	1
AUCP_ADD_CELL_LAST	2
AUCP_CREATE_CPOOL	3
AUCP_DESTROY_CPOOL	4
AUCP_GET_CELL	5

Reason code values

AUCP_NO_STORAGE	1
AUCP_CPOOL_FAILED	2
AUCP_INVALID_FUNCTION	3
AUCP_NOT_FOUND	4

Date formatter diagnostics

This section

Lists the meaning for each possible value of the call parameters that are included in the error messages issued if an an error occurs on a call to the CICS Transaction Affinities Utility date formatter, CAUCAFDT.

Reason code values

CAUD_NO_DATE	1
CAUD_FORMAT	2

Index

G

I

L

P

R

S

T

U

V

Sending your comments to IBM

IBM CICS Transaction Affinities Utility MVS/ESA

User's Guide

SC33-1159-01

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book. Please limit your comments to the information in this book and the way in which the information is presented.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail, use the Readers' Comment Form.
- By fax:
  - From outside the U.K., use your international access code + 44 962 870229
  - From within the U.K., use 0962 870229
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: WINVMJ(IDRCF)
  - Internet: idrcf@winvmj.vnet.ibm.com

Whichever you use, ensure that you include:

- The publication number and title
- The page number or topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Readers' Comments

IBM CICS Transaction Affinities Utility MVS/ESA

User's Guide

SC33-1159-01

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

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

Be sure to print your name and address below if you would like a reply.

Name	Address
Company or Organization	
Telephone	Email

You can send your comments POST FREE on this form from any one of these countries:

Australia	Finland	Iceland	Netherlands	Singapore	United States
Belgium	France	Israel	New Zealand	Spain	of America
Bermuda	Germany	Italy	Norway	Sweden	
Cyprus	Greece	Luxembourg	Portugal	Switzerland	
Denmark	Hong Kong	Monaco	Republic of Ireland	United Arab Emirates	

If your country is not listed here, your local IBM representative will be pleased to forward your comments to us. Or you can pay the postage and send the form direct to IBM (this includes mailing in the U.K.).

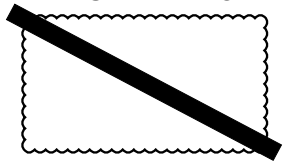
1 Cut along this line

2 Fold along this line

By air mail
Par avion

IBRS/CCRI NUMBER: PHQ - D/1348/SO

NE PAS AFFRANCHIR



NO STAMP REQUIRED

REPONSE PAYEE
GRANDE-BRETAGNE

IBM United Kingdom Laboratories Limited
Information Development Department (MP 095)
Hursley Park
WINCHESTER, Hants
SO21 2ZZ United Kingdom

3 Fold along this line

From: Name _____

Company or Organization _____

Address _____

_____

EMAIL _____

Telephone _____

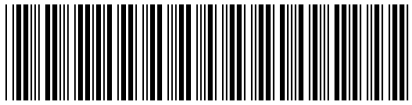
1 Cut along this line

4 Fasten here with adhesive tape

Program Number: 5696-582

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-1159-01



Spine information:

IBM CICS Transaction Affinities Utility MVS/ESA User's Guide