CICS® Transaction Server for OS/390®

# CICS Transaction Affinities Utility Guide

*Release 3*

SC33-1777-30

CICS® Transaction Server for OS/390®

# CICS Transaction Affinities Utility Guide

*Release 3*

SC33-1777-30

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

**Fourth edition (November 2000)**

This edition applies to Release 3 of CICS Transaction Server for OS/390, program number 5655-147, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

This edition replaces and makes obsolete the previous edition, SC33-1777-01. Changes since that edition are indicated by a # sign to the left of a change. Any vertical lines in the left margin indicate a change made between Version 1 Release 2 and Version 1 Release 3 of CICS Transaction Server for OS/390.

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

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

IBM United Kingdom Laboratories, Information Development,
Mail Point 95, 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

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

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

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM United Kingdom Laboratories, MP151, Hursley Park, Winchester, Hampshire, England, SO21 2JN. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

## Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

| | |
|---|---|
| BookManager | DB2 |
| CICS | IBM |
| CICS/ESA | Language Environment |
| CICSPlex | VTAM |

Other company, product, and service names may be trademarks or service marks of others.

# Preface

## What this book is about

This book describes the **affinity utility program**. 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 CICS **dynamic routing** 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®SM for workload balancing. For more information about CICSPlex SM, see the *CICSPlex SM Concepts and Planning* manual.

It is also of use if you are planning to implement asynchronous processing using CICS function shipping or are planning to use the CICS transaction isolation facility.

## What you need to know to understand this book

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

"Chapter 1. Introducing 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 Application Programming Guide*.

## How to use this book

This book is intended to be read sequentially, so that you understand how to:

1. Install the affinity utility
2. Run the separate components

Later, when you are familiar with the utility, you need only refer to the chapter dealing with the particular component that you want to run.

## Notes on terminology

**CICS** In general, this book refers to the Customer Information Control System as "CICS", the element in the CICS Transaction Server for OS/390.

**MVS** "MVS" is used for the operating system, which is an element of the CICS Transaction Server for OS/390.

**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.

# Bibliography

## CICS Transaction Server for OS/390

| | |
|---|---|
| *CICS Transaction Server for OS/390: Planning for Installation* | GC33-1789 |
| *CICS Transaction Server for OS/390 Release Guide* | GC34-5352 |
| *CICS Transaction Server for OS/390 Migration Guide* | GC34-5353 |
| *CICS Transaction Server for OS/390 Installation Guide* | GC33-1681 |
| *CICS Transaction Server for OS/390 Program Directory* | GI10-2506 |
| *CICS Transaction Server for OS/390 Licensed Program Specification* | GC33-1707 |

## CICS books for CICS Transaction Server for OS/390

| | |
|---|---|
| **General** | |
| *CICS Master Index* | SC33-1704 |
| *CICS User's Handbook* | SX33-6104 |
| *CICS Transaction Server for OS/390 Glossary* (softcopy only) | GC33-1705 |
| **Administration** | |
| *CICS System Definition Guide* | SC33-1682 |
| *CICS Customization Guide* | SC33-1683 |
| *CICS Resource Definition Guide* | SC33-1684 |
| *CICS Operations and Utilities Guide* | SC33-1685 |
| *CICS Supplied Transactions* | SC33-1686 |
| **Programming** | |
| *CICS Application Programming Guide* | SC33-1687 |
| *CICS Application Programming Reference* | SC33-1688 |
| *CICS System Programming Reference* | SC33-1689 |
| *CICS Front End Programming Interface User's Guide* | SC33-1692 |
| *CICS C++ OO Class Libraries* | SC34-5455 |
| *CICS Distributed Transaction Programming Guide* | SC33-1691 |
| *CICS Business Transaction Services* | SC34-5268 |
| **Diagnosis** | |
| *CICS Problem Determination Guide* | GC33-1693 |
| *CICS Messages and Codes* | GC33-1694 |
| *CICS Diagnosis Reference* | LY33-6088 |
| *CICS Data Areas* | LY33-6089 |
| *CICS Trace Entries* | SC34-5446 |
| *CICS Supplementary Data Areas* | LY33-6090 |
| **Communication** | |
| *CICS Intercommunication Guide* | SC33-1695 |
| *CICS Family: Interproduct Communication* | SC33-0824 |
| *CICS Family: Communicating from CICS on System/390* | SC33-1697 |
| *CICS External Interfaces Guide* | SC33-1944 |
| *CICS Internet Guide* | SC34-5445 |
| **Special topics** | |
| *CICS Recovery and Restart Guide* | SC33-1698 |
| *CICS Performance Guide* | SC33-1699 |
| *CICS IMS Database Control Guide* | SC33-1700 |
| *CICS RACF Security Guide* | SC33-1701 |
| *CICS Shared Data Tables Guide* | SC33-1702 |
| *CICS Transaction Affinities Utility Guide* | SC33-1777 |
| *CICS DB2 Guide* | SC33-1939 |

## CICSPlex SM books for CICS Transaction Server for OS/390

| | |
|---|---|
| **General** | |
| *CICSPlex SM Master Index* | SC33-1812 |
| *CICSPlex SM Concepts and Planning* | GC33-0786 |
| *CICSPlex SM User Interface Guide* | SC33-0788 |
| *CICSPlex SM Web User Interface Guide* | SC34-5403 |
| *CICSPlex SM View Commands Reference Summary* | SX33-6099 |
| **Administration and Management** | |
| *CICSPlex SM Administration* | SC34-5401 |
| *CICSPlex SM Operations Views Reference* | SC33-0789 |
| *CICSPlex SM Monitor Views Reference* | SC34-5402 |
| *CICSPlex SM Managing Workloads* | SC33-1807 |
| *CICSPlex SM Managing Resource Usage* | SC33-1808 |
| *CICSPlex SM Managing Business Applications* | SC33-1809 |
| **Programming** | |
| *CICSPlex SM Application Programming Guide* | SC34-5457 |
| *CICSPlex SM Application Programming Reference* | SC34-5458 |
| **Diagnosis** | |
| *CICSPlex SM Resource Tables Reference* | SC33-1220 |
| *CICSPlex SM Messages and Codes* | GC33-0790 |
| *CICSPlex SM Problem Determination* | GC33-0791 |

## Other CICS books

| | |
|---|---|
| *CICS Application Programming Primer (VS COBOL II)* | SC33-0674 |
| *CICS Application Migration Aid Guide* | SC33-0768 |
| *CICS Family: API Structure* | SC33-1007 |
| *CICS Family: Client/Server Programming* | SC33-1435 |
| *CICS Family: General Information* | GC33-0155 |
| *CICS 4.1 Sample Applications Guide* | SC33-1173 |
| *CICS/ESA 3.3 XRF Guide* | SC33-0661 |

If you have any questions about the CICS Transaction Server for OS/390 library, see *CICS Transaction Server for OS/390: Planning for Installation* which discusses both hardcopy and softcopy books and the ways that the books can be ordered.

# Summary of changes

This book is based on the CICS Transaction Affinities Utility Guide for SC33–1777–21. Changes from that edition are marked by vertical bars in the left margin.

This part lists briefly the changes that have been made for the following recent releases:

## Changes for CICS Transaction Server for OS/390 Release 3

Minor changes.

## Changes for the CICS Transaction Server for OS/390 Version 1 Release 2 edition

Minor changes.

## Changes for the CICS Transaction Server for OS/390 Version 1 Release 1 edition

The chapter entitled “Installing the affinity utility” was renamed to “Preparing to use the Transaction Affinities Utility”, and was largely rewritten.

The messages and codes previously published in Appendix C were moved to the *CICS Messages and Codes* manual. The affinity utility program messages became standard CICS messages. They are prefixed with the letters “DFH”, and have a component identifier of “AU”.

# Chapter 1. Introducing 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 routing (known in previous releases of CICS as dynamic *transaction* routing) environment. For more information about transaction affinities, see the *CICS Application Programming Guide*.

This chapter introduces the following topics:



CICS has been handling customers' online transaction processing requirements for over thirty years. In that time, it 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 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. The CICSPlex SM element of CICS Transaction Server for OS/390 Release 3 includes a workload management component that optimizes processor capacity by dynamically routing transactions to whichever CICS region is the most appropriate at the time, taking into account any transaction affinities that exist. For an introduction to CICSPlex SM, see *CICSPlex SM Concepts and Planning*; for information about CICSPlex SM workload management, see *CICSPlex SM Managing Workloads*.

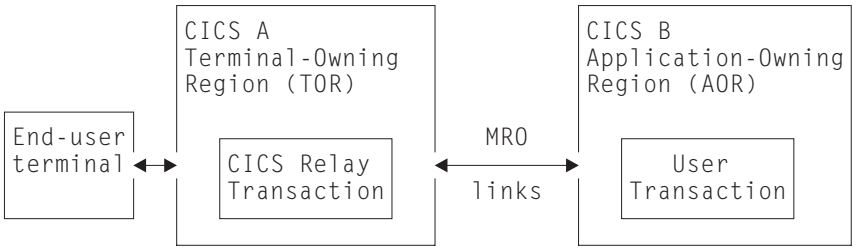


*Figure 1. The CICS transaction routing facility*

Before CICS Transaction Server for OS/390 Release 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.

CICS Transaction Server for OS/390 Release 3 introduces extended dynamic routing facilities, that allow the dynamic routing of:

- Transactions initiated at a terminal
- EXEC CICS START requests that are associated with a terminal
- EXEC CICS START requests that are not associated with a terminal
- Dynamic program link (DPL) requests that are received using:
  - The CICS Web support
  - The CICS Transaction Gateway
  - External CICS interface (EXCI) client programs
  - Any CICS client workstation products using the External Call Interface (ECI)
  - Distributed Computing Environment (DCE) remote procedure calls (RPCs)
  - Open Network Computing (ONC) RPCs
  - Internet Inter-Object Request Block Protocol (IIOP)
  - Any function that issues an EXEC CICS LINK PROGRAM request
- Transactions associated with CICS business transaction services (CICS BTS) activities.

New terms have been introduced that describe the roles played by CICS regions in dynamic routing:

**Requesting region**
: The CICS region in which the dynamic routing request originates. For transactions initiated at a terminal, and inbound client DPL requests, this is typically a TOR. For terminal-related EXEC CICS START commands, for non-terminal-related EXEC CICS START commands, for peer-to-peer DPLs, and for CICS BTS activities, the requesting region is typically an AOR.

**Routing region**
: The CICS region in which the decision is taken on where the transaction or program should be run. For transactions initiated at a terminal, for EXEC CICS START commands associated with a terminal, and for inbound client DPL requests, this is typically a TOR. For non-terminla-related EXEC CICS START commands, for peer-to-peer DPL requests, and for CICS BTS activities, the routing region is typically an AOR.

**Target region**
: The CICS region in which the transaction or program runs. For all dynamically-routed requests, this is typically an AOR.

Full details about the new dynamic routing facilities are described in *CICS Intercommunication Guide*.

The dynamic routing facility removes the need to specify the remote system name of a target region in the transaction definition. Instead, you let the routing determine dynamically to which target region it should route incoming transactions. Unlike static routing, where there can only ever be one target region to which the routing region can route a transaction, dynamic routing gives you the means to create several target regions with the capability to process any given workload, and to let the routing regions choose the best one from a candidate list.

## The benefits of dynamic routing

Being able to route transactions to target regions 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, this depends on your CICS environment and routing policies. It needs a facility for you to specify your routing policies in a form that CICS can use. This can be a user-written dynamic routing program used to supply the name of a suitable target region, or you can use the dynamic routing program EYU9XLOP provided with CICSPlex SM.. You can define the name of a dynamic routing program on either the DTRPGM system initialization (SIT) parameter, for terminal-related START and dynamic program link (DPL) requests, or the DSRTPGM SIT parameter for non-terminal-related START requests and CICS BTS processes.

At the basic level, a dynamic routing program simply contains tables of user transaction identifiers, with the matching system identifiers (SYSIDs) of the target regions 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 target region 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 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.

There are two categories of affinity:
- Inter-transaction affinity; see “Inter-transaction affinity”
- Transaction-system affinity; see “Transaction-system affinity” on page 4

The restrictions on dynamic routing caused by transaction affinities depend on the duration and scope of the affinities. Clearly, the ideal situation for a dynamic routing program is for there to be no transaction affinity at all, which means there is no restriction in the choice of available target regions for dynamic routing. However, even when transaction affinities do exist, there are limits to the scope of these affinities determined by the:
- Affinity relations; see “Affinity relations” on page 4
- Affinity lifetime; see “Affinity lifetimes” on page 4

Note that, if you are dynamically routing non-terminal-related START and DPL requests, you should review your application to determine whether or not the application is suitable for dynamic routing. The Transaction Affinities Utility cannot detect affinities in these circumstances.

## 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. 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 to another transaction, are not eligible for dynamic transaction routing. In general, they are transactions that use INQUIRE and SET commands or, depend on global user exit programs.

## Affinity relations

The affinity relation determines how the dynamic routing program selects a target region for a transaction instance associated with the affinity. 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 target region for the lifetime of the affinity. The affinity lifetime for global relations can be system or permanent.

**BAPPL**
: All instances of all transactions in the group are associated with the same CICS BTS (Business Transaction Services) process. There may be many different userids and terminals associated with the transactions included in this affinity group.

**LUname**
: A group of transactions where **all** instances of all transactions in the group that are initiated from the **same terminal** must execute in the same target region for the lifetime of the affinity. The affinity lifetime for LUname relations can be pseudoconversation, logon, system, or permanent.

**Userid**
: A group of transactions 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 target region for the lifetime of the affinity. The affinity lifetime for userid relations can be pseudoconversation, signon, system, or permanent.

## Affinity lifetimes

The affinity lifetime determines when the affinity is ended. An affinity lifetime can be classified as one of:

**System**
The affinity lasts for as long as the target region exists, and ends whenever the target region 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.

**Process**
The affinity exists until the process completes.

**Activity**
The affinity exists until the activity completes.

**Pseudoconversation**
The (LUname or userid) affinity lasts for the whole pseudoconversation, and ends when the pseudoconversation 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 pseudoconversation and signon lifetimes are possible only in those situations where 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).
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), 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

## Safe programming techniques

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

- The communication area (COMMAREA) on CICS RETURN commands
- A terminal control table user area (TCTUA) optionally available for each terminal defined to CICS
- ENQMODEL definitions to give sysplex-wide scope to ENQs and DEQs

## 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 commands
  - ENQ and DEQ commands that do not specify a length parameter and therefore ENQ by address
  - ENQ and DEQ commands that do specify a length and therefore ENQ by name, unless you have used ENQMODEL definitions to give sysplex-wide scope to the ENQs (and DEQs)

## Suspect programming techniques

Some programming techniques may 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 routing environment.

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

- Temporary storage queues with restrictive naming conventions
- Transient data queues and trigger levels
- Synchronization or serialization of tasks using CICS commands:
  - RETRIEVE WAIT / START
  - START / CANCEL REQID
  - DELAY / CANCEL REQID
  - POST / CANCEL REQID
- INQUIRE and SET commands and global user exits

# Avoiding the effects of transaction affinity

In a dynamic routing environment, your dynamic routing program must take account of transaction affinity in order to route transactions effectively. Where possible, you should avoid creating application programs that cause affinity. 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 routing environment. The Transaction Affinities Utility is designed to help you with this task.

# Protecting applications from one another

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

Transaction isolation ensures that user-key programs[1] 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 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 the 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:

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

- Issue MVS macros directly
- Modify CICS control blocks
- 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, which may prevent them running isolated from each other. To allow such transactions to continue running 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 run. They are then isolated from the other transactions in the system that are running in their own subspaces, but are able to share each other's data within the common subspace.

You may 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, by another task, either an MVS POST or 'hand posting'. 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 hence fail 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 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. Introducing the Transaction Affinities Utility" on page 9 gives an overview of the Transaction Affinities Utility, and details of all the commands and command sequences that the 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. Introducing the Transaction Affinities Utility

This chapter gives an overview of the Transaction Affinities Utility, and describes the basic components:



The 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 routing facility. It can be used to detect programs using 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, the CICSPlex SM element of CICS Transaction Server for OS/390 Release 3.

The commands that can be detected are listed in "Commands detected by the Transaction Affinities Utility" on page 11. The 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.

The Transaction Affinities Utility determines the affinities that apply to a single CICS region: that is, a single pure target region or single combined routing region/target region. 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.

**Important note**

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

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

Figure 2 shows the affinity utility program. Each of the four components is described in more detail in the rest of this chapter.

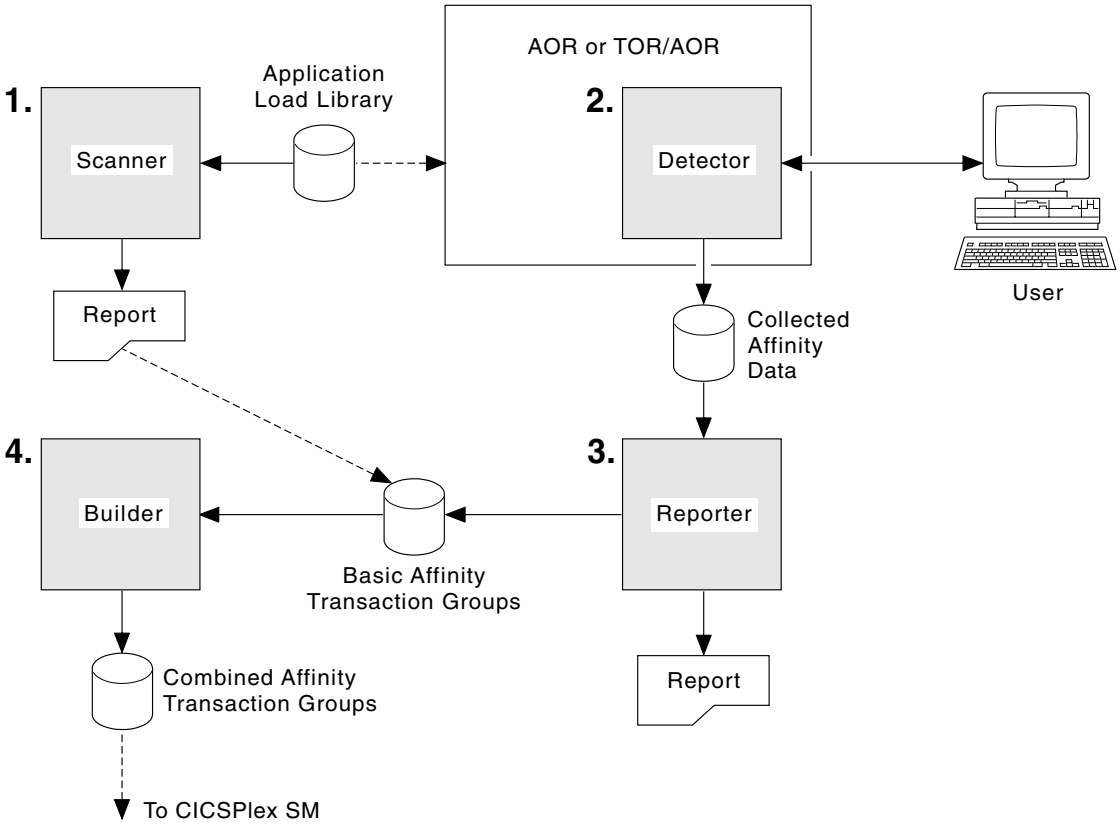


*Figure 2. Affinity utility program components*

## Commands detected by the Transaction Affinities Utility

You can use the Transaction Affinities Utility to detect instances of the EXEC CICS commands listed in Table 1.

*Table 1. Commands detected by the Transaction Affinities Utility*

| Inter-transaction affinity commands | Transaction-system affinity commands |
|---|---|
| ENQ<br>DEQ<br>READQ TS<br>WRITEQ TS<br>DELETEQ TS<br>ADDRESS CWA<br>LOAD<br>RELEASE<br>GETMAIN SHARED<br>FREEMAIN<br>RETRIEVE WAIT<br>DELAY<br>POST<br>START<br>CANCEL<br>COLLECT STATISTICS | ENABLE PROGRAM<br>DISABLE PROGRAM<br>EXTRACT EXIT<br>INQUIRE<br>SET<br>PERFORM<br>RESYNC<br>DISCARD<br>CREATE<br>WAIT EXTERNAL<br>WAIT EVENT<br>WAITCICS<br>CBTS STARTBROWSE<br>CBTS GETNEXT<br>CBTS ENDBROWSE |

**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 only those used to free GETMAIN SHARED storage 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 Transaction Affinities Utility does not search for transient data and file control EXEC CICS commands. 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 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 in which transactions can cause affinity with each other, but they are not readily detectable by the affinity utility program 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 the hand posting of ECBs.)
8. The Detector and the Report ignore ENQ and DEQ commands that specify an ENQSCOPE name.

## The Scanner component

The Scanner is a batch utility that scans a load module library to detect those programs in the library that issue EXEC CICS commands that may cause transaction affinity. It examines the individual object programs looking for patterns matching 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 11, and MVS POST requests.

The report produced by the Scanner indicates only that **potential** affinity problems may exist because it only identifies the programs that issue the commands. It cannot obtain dynamic information about the transactions using the programs, or the names of the resources acted upon. Use the report in conjunction with the main report produced by the Reporter (see "The Reporter component" on page 18).

**Notes:**

1. The Scanner operation is independent of the language the scanned program was written in and the release of CICS 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 zero format for an affinity command.
3. The Scanner does not detect CICS macro-level commands.
4. The Scanner distinguishes between ENQ by name and ENQ by address based on the presence of a length parameter on the EXEC CICS ENQ command. It does the same for DEQs. The reports show which ENQs and DEQs are by name and which are by address.

## The Detector component

You can use the Detector in real-time to detect transaction affinities in a running CICS region, and to save details of the affinities in an MVS 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 13.

The data is collected by the global user exit programs at exit points XEIOUT, XBADEACT, XMEOUT, 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 (pseudoconversation end, terminal log off, user sign off) 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 ix.

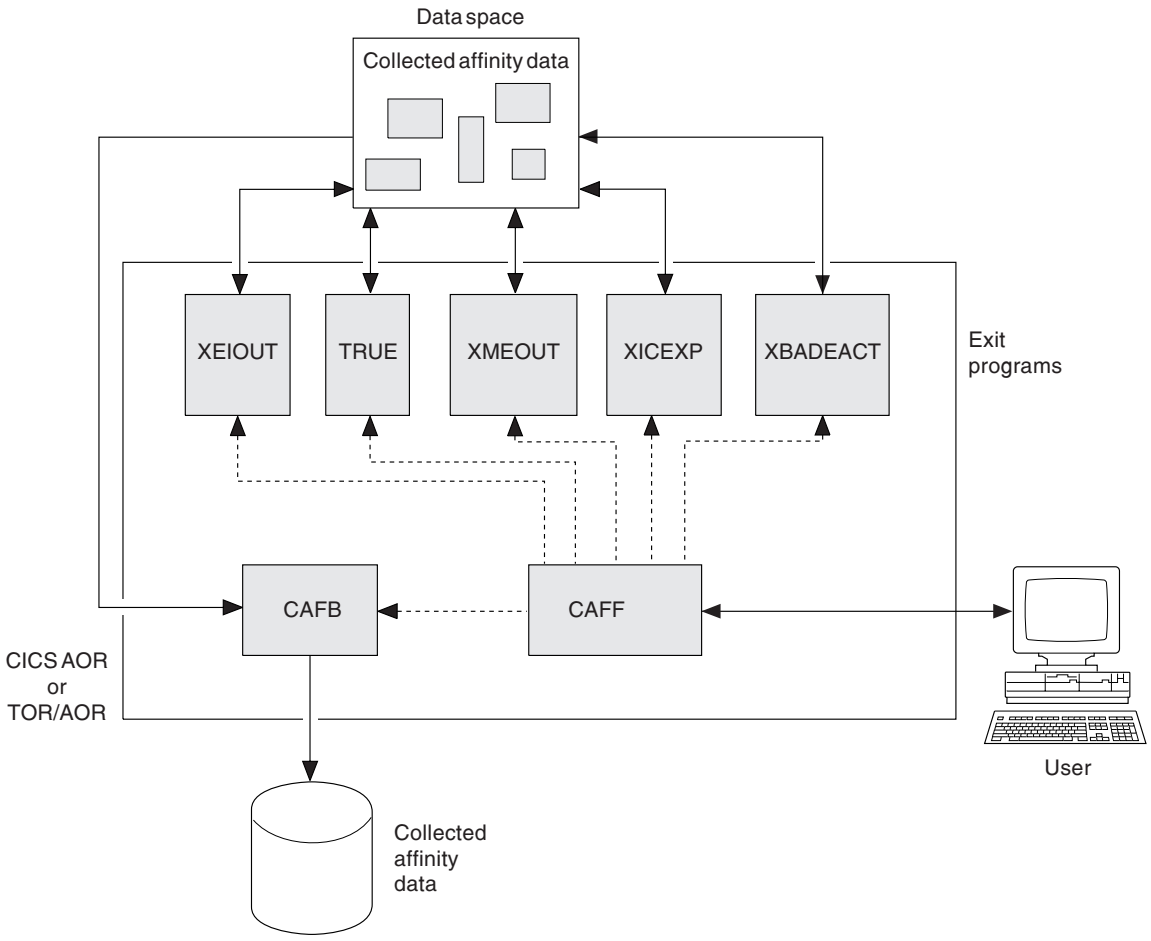


*Figure 3. Detector components*

You are recommended to run the Detector on stable CICS regions only. Do not apply maintenance to application programs while the Detector is running. Such maintenance 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 11 that can cause transaction affinity. For ENQ and DEQ commands, the Detector distinguishes between ENQ by name and ENQ by address based on the presence of a length parameter on the EXEC CICS ENQ command. It does the same for DEQs. The reports show which ENQs and DEQs are by name and which are by address.

It also detects:

- The end of pseudoconversations, by detecting when one of the transactions in the pseudoconversation terminates without issuing an EXEC CICS RETURN TRANSID command with a non-zero transaction identifier. If a pseudoconversation ends, and the resource shared by transactions that take part in the affinity still exists, the lifetime of the affinity must be greater than PCONV.
- Log offs and sign offs by intercepting messages DFHSN1200, DFHZC3462, and DFHZC5966.

- Completion of CICS BTS activities and processes.

For more information, see “Appendix A. Details of what is detected” on page 59.

### Worsening of transaction affinities relations

In some cases, the Detector may not detect enough occurrences (at least 10) of an affinity command to be sure that the affinity is definitely with a terminal (LUNAME), userid (USERID), or CICS BTS process (BAPPL). 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 pseudoconversation 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 log off or sign off occurs, and the resource still exists, the Detector deduces that the lifetime is longer than LOGON or SIGNON: that is, either SYSTEM or PERMANENT.

If a CICS BTS activity completes and the resource still exits, the lifetime is worsened to process. If a CICS BTS process completes and the resource still exits, the liftime is worsened to system.

In some cases, the Detector may not detect a log off or sign off, 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 the Detector is running on is a target region, because it is impossible in some cases to detect a log off or sign off that occurs in a connected routing region.

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 when the:

- Detector is not running
- Issuing program was translated with the SYSEIB option
- Command is an EXEC CICS FEPI command
- Command is not one that can cause transaction affinity
- Program name starts with “CAU” or “DFH”
- Program is a DB2 or DBCTL task-related user exit
- Program is a CICS user-replaceable module
- Transaction identifier does not match the prefix, if specified, for transactions to be detected
- Command is not one of the affinity types specified to be detected
- Command is function shipped to a remote CICS region
- Inter-transaction affinity commands are used within the same task
- Command is a non-terminal-related START or a DPL
- ENQ or DEQ commands that specify a resource name for which an appropriate ENQMODEL definition is enabled, and that ENQMODEL has a non—blank ENQSCOPE

The Detector does not detect CICS macro-level commands, MVS POST calls, or the hand posting of ECBs.

If you continue a pseudoconversation 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 pseudoconversation end.

Ideally the Transaction Affinities Utility should ignore commands issued by task-related user exits and global user exits because they are not part of applications. However, it cannot distinguish such commands from others, 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.

If an exit program at XICEREQ or XTSEREQ modifies the EXEC CICS command, 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, the Detector sees the modified value.

## 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 17.

## 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, CREATE, and RESYNC commands
   - CICS BTS BROWSE commands are treated as inquire 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 Transaction Affinities Utility files when you stop the Detector and, optionally, at predetermined intervals.

## Saving affinity data

The affinity data collected by the Detector is saved to the 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. Furthermore, 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 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. Time stamps 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 the next table is considered.

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. Ensure the files are 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 the 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 of varying length. Each record has a prefix that contains a one-byte table identifier identifying 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 the Reporter to validate that the files they have been presented with are indeed data files suitable for the Transaction Affinities Utility. 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 files to be cross-validated.

## The control record VSAM file

The 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. This will happen when the Detector options change, the Detector statistics change, or the Detector state changes to STOPPED.

**Note:** The supplied definition for the control record VSAM file makes it recoverable to CICS. You can change the definition if you do not require recovery. See “Defining the VSAM files to CICS” on page 20 for more information.

## Detector performance

The Detector is intended to be run against production CICS regions. However, over the period when 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 most likely reason for poor performance is the detection of TS commands. You may prefer to run the Detector twice, once for TS commands and once for all other affinity commands.

## The Reporter component

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

- **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 issuing the EXEC CICS commands that cause inter-transaction and transaction-system affinities. This information should help you understand the transactions making up the workload, and should highlight the changes you need to make 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 a single 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, because 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.

## The Builder component

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 produces 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 stating that a given tranid may appear only 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. Preparing to use the affinity utility program

This chapter describes what needs to be done before you can use the affinity utility program.



## Creating the VSAM files

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

*Table 2. Transaction Affinities Utility VSAM files and associated jobs*

| File | Description | Job |
|---|---|---|
| CICSAFF.CAUCNTL | A recoverable file used to hold control information | CAUJCLCC |
| CICSAFF.CAUAFF1<br>CICSAFF.CAUAFF2<br>CICSAFF.CAUAFF3 | Non-recoverable files used to hold affinity data with different key sizes (17, 33, 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 CICSTS13.CICS.SDFHINST library. Edit and run a copy of the CAUJCLCC and CAUJCLCA jobs for each CICS region against which you are going to use the Transaction Affinities Utility.

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

- The JOB accounting parameters.
- The prefix of the files (&AFFQ), this should contain a CICS region qualifier.
- The volume id (&DSVOL) of the DASD device where the files are to reside.

## Estimating the size of the MVS data space and VSAM 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):

```
Data space: (#transids * #termids * 250) + 5 000,000
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.

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:

```
Data space : 130 Megabytes
CAUAFF1   :   21  Megabytes
CAUAFF2   :   76  Megabytes
CAUAFF3   :    1   Megabyte
```

The space required for the data space is different than that required for the files because:

- each record has a storage overhead in the data space
- 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 5MB.
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.

## Defining the VSAM files to CICS

The CICS-supplied sample group, DFH$AFFY, contains definitions for:

- three affinity data files (CAUAFF1, CAUAFF2 and CAUAFF3)
- the affinity control file, CAUNCNTL

Change some of the attributes of these resource definitions to suit your own environment. To do this, use the CEDA transaction (or the DFHCSDUP utility) to:

1. COPY the sample group to a group of your own choosing. For example,
   ```
   CEDA COPY GROUP(DFH$AFFY) TO(mygroup)
   ```
2. EXPAND group *mygroup* and change the following attributes appropriately:
   - For each resource definition, change the prefixes of the VSAM files, as defined by the CAUJCLCC and CAUJCLCA jobs.
   - For each resource definition, ensure that the LSRPOOLID specified for each file is capable of handling the keylength defined for the file. If it isn't, change it so that it is. See Table 2 on page 19 for more information.
   - For file CAUCNTL only, if recovery is not required ensure that RECOVERY(NONE) and FWDRECOVLOG(NO) are specified.
3. INSTALL group *mygroup* to make these definitions known to CICS.

## Tailoring your CICS startup job

To enable the Transaction Affinities Utility to be run against your CICS region, take account of the following when setting up your CICS startup job:

- The Transaction Affinities Utility sends messages to the transient data destination CAFF. For you to see all the messages sent to CAFF, the queue must be predefined to the transient data component. The CAFF queue is defined in the CICS-supplied group, DFHDCTG. DFHDCTG contains definitions for all the CICS-supplied transient data queues and is installed on cold starts as part of DFHLIST processing. Unlike most CICS-supplied groups, DFHDCTG is unique in that it is not locked. Therefore, you can alter the definition for CAFF using either the DFHCSDUP utility or the CEDA transaction.

  You do not have to specify the extrapartition transient data data set in your start-up JCL, because of the introduction of dynamic allocation for such data sets. If you want to use dynamic allocation, specify a DSName in the definition of the queue. Alternatively, you can add a DD statement for the queue in your JCL. For example:

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

  The default definition in group DFHDCTG routes the messages sent to CAFF to the SYSOUT class '*'.

  For more information about dynamic allocation and defining transient data queues, see the *CICS Resource Definition Guide*.
- Set the ICVR system initialization parameter to at least 10 seconds; that is, ICVR=10000 (or a larger value). If you do not do this, the Detector or one of your own transactions may end prematurely with an abend code of AICA.
- Set the DSHIPINT system initialization parameter to 0. If you specify a DSHIPINT value other than 0, the utility may not correctly observe terminal log offs or determine affinity life times.

If the CICS region is a target region, specify 0 on the AILDELAY system initialization parameter for **all routing regions** that route to it. This enables the Detector to immediately see when a terminal logs off.

(If you specify an AILDELAY value other than 0, the Detector may miss log offs and misinterpret affinity life times.)

## Restarting your CICS region

Restart the CICS region using a CICS startup job modified for affinity utility program support, as described in “Tailoring your CICS startup job”.

# Chapter 4. Running the Scanner

This chapter describes how to run the Scanner that 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 and module list to identify suspect modules 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 by:
- "Creating a summary report"
- "Creating a detailed report" on page 25

## 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]'
  ```

  where:

  **$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 containing at least one possible affinity-causing EXEC CICS command or MVS POST command 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 Transaction Affinities Utility load library where you have installed the Scanner program, CAULMS.
- **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 where 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)**

  You do not need this 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.
- The total count 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 4 on page 25 is an example of a summary report produced by the Scanner.

4. For an explanation of argument zero, see “Notes on terminology” on page ix .

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

```
CICS TRANSACTION AFFINITIES UTILITY                                          1995/11/24  Page   1
LOAD MODULE SCANNER - SUMMARY LISTING OF CICSTEST.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                                          1995/11/24  Page   2
LOAD MODULE SCANNER - SUMMARY LISTING OF CICSTEST.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 4. 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 statements 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 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 Transaction Affinities Utility load library in which you have installed the Scanner program, CAULMS.
- **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**

  You do not need this 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.

## Contents of a detailed report

Each detailed report contains a section for each module, with:

- 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 43 and page 65.)
  - 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, 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), or transaction-system (System).
- 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 5 is an example of a detailed report produced by the Scanner.

```
CICS TRANSACTION AFFINITIES UTILITY                                          1995/11/24  Page     1
LOAD MODULE SCANNER - DETAILED LISTING OF CICSTEST.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                                          1995/11/24  Page     2
LOAD MODULE SCANNER - DETAILED LISTING OF CICSTEST.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 5. Example of a detailed report produced by the Scanner*

# Chapter 5. Running the Detector

This chapter describes how to run the Detector that runs in a CICS region looking for instances of API commands that could cause transaction affinity.

This chapter describes how to perform the following functions:
- "Displaying the Detector control screen"
- "Starting the collection of affinity data" on page 31
- "Pausing the collection of affinity data" on page 32
- "Resuming the collection of affinity data" on page 32
- "Stopping the collection of affinity data" on page 33
- "Changing the Detector options" on page 34
- "Detector errors" on page 36

The commands looked for are those listed in "Commands detected by the Transaction Affinities Utility" on page 11.

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 "The Detector component" on page 12.

You can control the Detector by:

- **Changing the state**

  This is described in topics "Starting the collection of affinity data" on page 31 through "Stopping the collection of affinity data" on page 33.
- **Changing the options**

  The options that you can select are shown in Figure 7 on page 34, and described in "Changing the Detector options" on page 34.

You can optimize the performance of the Detector; 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.
- Collecting data for one affinity type at a time. This can be of particular value for the temporary storage affinity type.
- Collecting data for a restricted set of transaction identifiers by specifying the prefix of those transactions in which you are interested.

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 34.

## 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 6 on page 30), 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 7 on page 34).

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

To leave 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 CICSPDN1
  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. . : 11/24/95 09:05:23 (MM/DD/YY HH:MM:SS)  6
  Date/time of last save . . : 11/24/95 11:13:12 (MM/DD/YY HH:MM:SS)
  Date/time of last change . : 11/24/95 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 6. 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 display shows the reason why it was stopped.

**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 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. (This is not displayed while in STOPPED state.)

**9** The message line used to display diagnostic messages.

**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 collecting affinity data**
You can start collecting affinity data only when the Detector is currently stopped.

To start collecting affinity data, use one of the methods shown in Table 3.

*Table 3. 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 Customization Guide*.

Using one of the methods listed in Table 3 causes the Detector to record transaction affinities in the CICS region, until you pause or stop data collection by the Detector. Data is collected for only the affinity types that you have chosen 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 help with calculating the likely data space storage requirement, see “Estimating the size of the MVS data space and VSAM files” on page 19. You specify this size on the Detector options screen, CAFF02. You can also specify that data from affinity data VSAM files (for example, from previous 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 34.

**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 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 4.

*Table 4. 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 1 |
| Application program | EXEC CICS START TRANSID('CAFF') FROM('PAUSE') |

**Note:**

1 cicsjob is the name of your CICS startup job.

Using one of the methods listed in Table 4 causes the Detector to stop collecting data until you are ready to resume. 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 that data be saved when the Detector is paused on the CAFF02 options screen, as described in "Changing the Detector options" on page 34.)

## Resuming the collection of affinity data

**When you can resume collecting affinity data**
You can resume collecting affinity data only when the Detector is currently paused.

To resume collecting affinity data, use one of the methods shown in Table 5.

*Table 5. 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 1 |
| Application program | EXEC CICS START TRANSID('CAFF') FROM('CONTINUE') |

**Note:**

**1** cicsjob is the name of your CICS startup job.

Using one of the methods listed in Table 5 on page 32 causes the Detector to continue recording any transaction affinities in the CICS region, until you pause or stop data collection.

## Stopping the collection of affinity data

**When you can stop collecting affinity data**
You can stop collecting affinity data only when the Detector is currently running or paused.

To stop collecting affinity data, use one of the methods shown in Table 6.

*Table 6. 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, a program specified in the first half of the PLT for CICS shutdown. This is recommended, to prevent the Detector delaying CICS shutdown if the Detector is not in the STOPPED state. For more information about using PLTSD programs, see the *CICS Customization Guide*.

Using one of the methods in Table 6 stops the Detector recording any transaction affinities in the CICS region until you next start collecting 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 how the Detector operates by changing the options that it uses. Option values are preserved in the 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 17.

**Notes:**

1. Generally, any option 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 34.)
2. You can change an option 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 displays its options screen, CAFF02 (shown in Figure 7) 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 CICSPDN1
  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 MB)
    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
    ADDRESS CWA. . . Y     EXTRACT. . . . . Y      CREATE . . . . . Y
    RETRIEVE WAIT. . Y     COLLECT STATS. . . Y
    LOAD . . . . . Y     PERFORM. . . . . Y
    GETMAIN SHARED . Y     RESYNC . . . . . Y
    CANCEL . . . . . Y
Last updated by <userid>  on 11/24/95 09:04:35  4
F1=Help  5
                                                        F12=Cancel
```

*Figure 7. Detector options screen*

The Detector options screen, CAFF02, shows the options available to you. You can change an option only when the Detector has stopped, unless one of the notes that follow says otherwise.

**Notes:**

**1 The control options:**

- **Perform periodic saves**

  Whether or not you want the affinity data collected 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.

- **Restore data on start**

  Whether or not you want the affinity data to be restored from the affinity data VSAM files when the Detector is started. This enables affinity data to be added to the data collected from previous runs of the Detector. If you are gathering data use this:
  – For one set of transaction identifiers at a time
  – For one set of commands at a time
  – 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 chosen to be detected (by specifying Y for the affinity type on the CAFF02 screen).

     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. You can change this option while the Detector is stopped, paused, or running.

- **Multiple signon with the same userid**

  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.

  Also, if you are running the Detector in an AOR, 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 the multiple signon option to Y if your conventions allow the same AOR userids to be signed on to CICS at the same time.

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

- **Size of dataspace**

  The size that you want to use for the data space 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 data space, see “Estimating the size of the MVS data space and VSAM files” on page 19.

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

  If you are saving affinity data, there may be a delay from the time the data space became full until the time the Detector actually terminated.

- **Transid prefix**

The prefix (zero through three characters) of the transactions for which you want to gather affinity data, 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 with identifiers starting with the prefix.

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

**2 Detect affinity types**

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 “The Detector component” on page 12 and “Appendix A. Details of what is detected” on page 59.

**3 Transaction system**

WAIT commands are listed as transaction-system affinities, because only half 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 Last update by <userid>**

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

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

**5 Help**

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

## Detector errors

If the CAFF or CAFB transaction, or an exit program, encounters a serious error, the Detector stops by terminating CAFF and CAFB with one of the following termination codes:

- A code in the AUxx range accompanied by messages on the CAFF transient data queue that indicate the cause of the error
- A code not in the AUxx range, presumably being one of the other CICS transaction abend codes

For a description of the code, see the *CICS Messages and Codes* manual.

If the CAFF transaction stops with code ATCH, ATNI, or AKCT, the Detector continues to collect affinity data. For all other codes from 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 abnormal termination occurs in an exit program, it results in an abnormal termination of the transaction that caused the exit to be invoked. This probably indicates a problem with the Detector. Use the CAFF transaction to stop the Detector, and contact your IBM Support Center if the evidence points to the Detector being at fault.

If the termination 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 this by increasing the value of the ICVR system initialization parameter.

# Chapter 6. Running the Reporter

This chapter describes how to run the Reporter that 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 Transaction Affinities Utility” on page 11. For information about interpreting the report output by the Reporter, see “Using the affinity report” on page 45.

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.

This chapter contains the following information:



## Requesting a report from the Reporter

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 PARM parameter of the EXEC statement**

  For example:

  ```
  REPORT   EXEC PGM=CAUREP,PARM='WORSEN=YES'
  ```

  **[WORSEN={YES|NO}]**
  Specify whether the Reporter is to worsen transaction affinity relations for those affinities where the Detector has not detected at least 10 occurrences. For more information about worsened relations, see “Worsening of transaction affinities relations” on page 14.
- **The STEPLIB DD statement**

  Specify the name of the Transaction Affinities Utility load library where 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 specifying the applid of the CICS region that created the record. The Reporter checks these applids against 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 Transaction Affinities Utility control VSAM file for this CICS region.
- **The CMDGRPS DD statement**

  Specify the affinity (command) types you want to see 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 |
| CREATE | 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 where the basic affinity transaction groups are to be sent.
- **The SYSPRINT DD statement**

  Specify the destination for the report.

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

## Output from the Reporter

For each affinity type specified on the CMDGRPS DD statement, the Reporter outputs each individual affinity both in 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 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.

## Affinity report

Figure 8 on page 41 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                                    1995/11/24  Page  1
AFFINITY TYPE REPORTING OPTIONS                                              Applid=CICSPDN1
Affinity Type       Reporting      Message
-------------       ---------      -------------------------------------
1
  Inter-Transaction Affinities  2
  ----------------------------
  CWA                 Yes
  CANCEL              No
  ENQ                 No
  GETMAIN             No
  LOAD                No
  RETRIEVE            No
  TS                  Yes
  Transaction-System Affinities
  -----------------------------
  COLLECT             No
  DISCARD             No
  ENABLE              No
  EXTRACT             No
  INQUIRE             No
  PERFORM             No
  RESYNC              No
  WAIT                No
  CREATE              No
```

*Figure 8. A sample report output by the Reporter (Part 1 of 4)*

```
CICS TRANSACTION AFFINITIES UTILITY                                    1995/11/24  Page  2  3
INTER-TRANSACTION AFFINITIES REPORT FOR ADDRESS CWA                          Applid=CICSPDN1
Trangroup    :  CW.00000001
Affinity     :  GLOBAL
Lifetime     :  SYSTEM
Tranid    Program      Offset      Usage    Command      Terminal    BTS Task
------    --------    --------    ------    -----------  --------    ---------
 AUXX     AUXXTST     000000CC         1    ADDRESS CWA     Yes          Yes
 CWA1     AUCWA       FFFFFFFF         2                    Yes          No
          Total Transactions      :        2
          Total Programs          :        2
```

*Figure 8. A sample report output by the Reporter (Part 2 of 4)*

```
CICS TRANSACTION AFFINITIES UTILITY                                    1995/11/24  Page  3  3
INTER-TRANSACTION AFFINITIES REPORT FOR TEMPORARY STORAGE COMMANDS            Applid=CICSPDN1
Trangroup    :  TS.00000001
Affinity     :  LUNAME
Lifetime     :  PCONV
Queue        :  LOCA1         (D3D6C3C1F1404040)
Recoverable  :  No            (MAIN)
Terminal Id  :  V102          (E5F1F0F2)
Tranid    Program      Offset     Command      Usage    Terminal    BTS Task
------    --------    --------    --------    ------    --------    ---------
 AFTD     AFFYTSD     0000012E    DELETEQ         43       Yes          Yes
 AFTR     AFFYTSR     000002BE    READQ           43       Yes          No
 AFTW     AFFYTSW     00000260    WRITEQ          43       Yes          Yes
          Total Transactions      :       3
          Total Programs          :       3
```

*Figure 8. A sample report output by the Reporter (Part 3 of 4)*

```
Trangroup    :  TS.00000002
Affinity     :  LUNAME
Lifetime     :  PCONV
Queue        :  LOCA2         (D3D6C3C1F2404040)
Recoverable  :  No            (MAIN)
Terminal Id  :  V102          (E5F1F0F2)
Tranid    Program      Offset     Command      Usage    BTS Task   Terminal
------    --------    --------    --------    ------    --------    ---------
 AFTD     AFFYTSD     0000012E    DELETEQ         39       Yes         Yes
 AFTR     AFFYTSR     000002BE    READQ           39       No          Yes
 AFTW     AFFYTSW     00000260    WRITEQ          39       Yes         Yes
          Total Transactions      :       3
          Total Programs          :       3
```

*Figure 8. A sample report output by the Reporter (Part 4 of 4)*

**Notes for Figure 8:**

**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 and transaction-system.

**3 Affinities reports**

For each affinity transaction group, a report 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 used only to cross-reference the group 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 an affinity transaction group may vary from one run to another of the Detector or Reporter.

**Affinity**
The affinity relation. If appropriate, this also indicates whether the relation was worsened from a less restrictive relation. For more information about worsened relations, see “Worsening of transaction affinities relations” on page 14.

**Lifetime**
The affinity lifetime. If appropriate, this also indicates whether the lifetime was worsened from a less restrictive lifetime. For more information about worsened lifetimes, see “Worsening of transaction affinities lifetimes” on page 14.

**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:** An unprintable character appears 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 identifier of the terminal where the transactions taking part in the affinity were initiated. This information is available only for TS queue affinity, and is meaningful only if the affinity is LUNAME or worsened from LUNAME to GLOBAL. Therefore, the terminal identifier is included in the report only in these cases.

**Tranid** The identifier of each transaction taking 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 pseudoconversation accesses a TS queue, and each part runs under the same tranid.

**Program**
The name of each program taking part in the affinity.

**Offset** The offset from the load point of the BALR instruction at the EXEC CICS command causing the affinity. The Reporter outputs a negative offset (X'FFFFxxxx') if it could not determine an offset; that is, if 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, a VS COBOL II dynamic call).

**Notes:**

1. This offset is **not** the same as the offset given by the Scanner, which is the offset of the command argument 0 declaration from the start of the load module. (See page 65.)
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 every instance of the affinity command, as there may be more than one.

**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) taking part in the affinity, up to a limit of 5000.

**Note:** The usage count is an indication of the relative importance of the affinity, and is not 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,        neither increment nor save flag set
```

If the usage count is ’1+’, it means that at least one example of the affinity was seen but that the total number of occurrences of that affinity 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.)

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

**BTS Task**
Whether it is a CICS BTS task or not.

**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.

## Producing affinity transaction group definitions

The Reporter produces affinity transaction group definitions suitable for input to the Builder (but **not** to 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 9 on page 45 shows a sample set of definitions to match the report in Figure 8 on page 41.

**Notes:**

1. The transaction group name is **not** a valid CICSPlex SM transaction group name, because 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 49.
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 52.

```
* HEADER APPLID(CICSPDN1)  SAVEDATE(95/11/24)  SAVETIME(10:11:45);
*
* Generated by the CICS Transaction Affinities Utility (Reporter) on 1995/11/25
* 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 9. Sample 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, to 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 two main purposes:

1. To help you 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 investigate whether any application changes could reduce the amount of affinity.

**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 makes dynamic routing less effective).

## 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 them 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, because those affinities most restrict dynamic 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 programming commands. It may not be appropriate to dynamically route such a transaction, because its action may be tied to a particular CICS region (as opposed to a particular set of other transactions).

The affinity report also lists affinities occurring between transactions that were not initiated from a terminal or are not CICS BTS transactions. These background transactions are known as **background relations**. This information is really for completeness, because such transactions cannot be dynamically routed.

To get complete information on affinities, use as many code paths as possible while running the Detector, because it can find an affinity only if the commands that cause it 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 outputs is very useful when establishing the full picture.

**Important note**

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

For more information about interpreting the affinity report, see “Appendix C. Useful tips when analyzing Transaction Affinities Utility reports” on page 69.

## Modifying affinity transaction groups

Consider making the following modifications to the affinity transaction groups before inputting them 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 of removing 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, BAPPL, or USERID may be worsened to GLOBAL because the Detector has not seen enough examples of the affinity to be convinced that it is related to a terminal or userid. Change this to LUNAME or USERID (and correct the lifetime) if you know that the affinity really is terminal- or userid-related. You may want to prevent worsening by specifying WORSEN=NO.

- **Remove affinity lifetime worsening**

  An LUNAME affinity with a lifetime of LOGON, or a USERID affinity with a lifetime of SIGNON, may be worsened to SYSTEM or PERMANENT because the Detector cannot always observe log offs or sign offs. Change this to LOGON or SIGNON if you know that to be the correct lifetime.

- **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, 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 affinities, because the Detector cannot detect the corresponding posting of the ECBs being waited upon. Identify the posting transactions and create affinity transaction groups to describe the affinities. The output from the Scanner may be particularly useful here, because it finds programs that issue MVS POST commands.

- **Add other affinities**

  Scanner output or your knowledge of your applications may identify additional affinities. Create affinity transaction groups to describe them.

- **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. Add such transactions to the affinity transaction group.

## Compressing affinity data

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). They may result in 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 ID. The Builder output for the previous example would be one affinity transaction group with relation LUNAME, lifetime PCONV, and transactions ABCD and ABCE.

# Chapter 7. Running the Builder

This chapter describes how to run the Builder that runs as a batch job to build affinity transaction groups suitable for input to the CICS system management product, the CICSPlex SM element of CICS Transaction Server for OS/390 Release 3.

This chapter contains the following information:


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 produces a file containing combined affinity transaction groups. CICSPlex SM requires a transaction identifier be in one transaction group only, and the Builder satisfies this by combining groups that contain the same transaction identifier.

**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 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 data space created internally by the Builder to store the group tables.

  **[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 CICS-managed address space (CMAS).

  For more information about defining transaction groups to CICSPlex SM, see *CICSPlex SM Managing Business Applications*.
- **The STEPLIB DD statement**

Specify the name of the Transaction Affinities Utility load library where 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 the lines 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 where the combined affinity transaction groups are to be sent. This data set is suitable for input to CICSPlex SM.

- **The 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 *CICS Transaction Server for OS/390 Installation Guide*.) The differences are given in the following list:

1. The only statements you can supply are:
   - CREATE statements for TRANGRPs and DTRINGRPs.
   - REMOVE statements for TRANGRPs.
   - TEXT statements and line comments. (A line comment is a line that starts with an asterisk (*) in column 1.)
   - HEADER statements for the Builder, and not for 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 allows only 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 reported as errors.
7. CONTEXT statements in the input data set are optional and are ignored by the Builder. They are overridden by the CONTEXT operand of the PARM statement (if specified) or the default.
8. A HEADER statement requires no keyword. 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 52.)

9. If a line comment contains the characters HEADER anywhere in it, it is not treated as a comment and is parsed like any ordinary 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, and BAPPL. 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 can 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 four 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 10. 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. The HEADER statement is vital because without it the Builder cannot detect the change from one input data set to another.

If you omit a HEADER statement, the Builder may generate error messages or add transactions to the wrong group, and give 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, 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 alphanumerically first 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 11 on page 53 shows the set of combined definitions that correspond to the basic definitions in Figure 9 on page 45, 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(95/11/27)  SAVETIME(12:00:51);          1
*
* Generated by the CICS Transaction Affinities Utility (Builder) on 1995/06/28
* 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 11. Sample 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 any of the input data sets. (See Figure 11 ( **1** ) and Figure 12 on page 55.)
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 “Compressing affinity data” on page 47.

## 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 7 through “Affinity relations” on page 4 show the relations and lifetimes that result from combining basic affinity transaction groups.

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

*Table 7. Resultant affinity relations*

| Relation A | Relation B | Resultant relation C |
|---|---|---|
| GLOBAL | Any relation | GLOBAL |
| BAPPL | BAPPL | BAPPL |
| BAPPL | LUNAME | GLOBAL |
| BAPPL | USERID | GLOBAL |
| LUNAME | LUNAME | LUNAME |
| LUNAME | USERID | GLOBAL |
| USERID | USERID | USERID |

*Table 8. 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 9. Resultant affinity lifetimes (BAPPL relation)*

| Lifetime X | Lifetime Y | Resultant lifetime Z |
|---|---|---|
| PERMANENT | Any lifetime | PERMANENT |
| SYSTEM | Any other combination | SYSTEM |
| PROCESS | PROCESS | PROCESS |
| PROCESS | ACTIVITY | SYSTEM |
| ACTIVITY | PROCESS | SYSTEM |
| ACTIVITY | ACTIVITY | ACTIVITY |

*Table 10. 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 11. 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 data sets that contain a HEADER statement appear in the report.

```
CICS TRANSACTION AFFINITIES UTILITY                                          1995/11/28    Page    1
BUILDER REPGRPS DATASETS PROCESSED REPORT
                Dataset Name                      CICS    Detector Last   Detector Last
                                                 APPLID     Save Date       Save Time
-------------------------------------------   --------  --------------  --------------
CICSPDN1.TRANGRPS.MERGE1                      CICSPDN1    95/11/25        09:05:09
CICSPDN1.TRANGRPS.MERGE2                      CICSPDN1    95/11/26        15:22:34
CICSPDN1.TRANGRPS.ONE                         CICSPDN1    95/11/27        12:00:51
```

*Figure 12. Sample data sets processed report*

## Empty transaction groups report

This report gives all basic transaction groups (trangroups) that were defined, but contained no transactions. It is produced only if the input data sets have no errors. 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                                          1995/11/28    Page    2
BUILDER EMPTY TRANGROUP DEFINITIONS
      CICSPDN1.TRANGRPS.EMPTY1
        G1          (GLOBAL SYSTEM   )     G2          (GLOBAL PERMANENT)     G3      (GLOBAL SYSTEM   )
      CICSPDN1.TRANGRPS.EMPTY2
        L2          (LUNAME PERMANENT)     L3          (LUNAME LOGON    )     L4      (LUNAME PCONV    )
```

*Figure 13. Example empty trangroups report*

## Group merge report

For each combined group, this report 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 an affinity lifetime. For example, in Figure 14 on page 56, four groups were merged: three were LUNAME and PCONV, and one was LUNAME and SYSTEM. The latter caused the lifetime worsening.

```
CICS TRANSACTION AFFINITIES UTILITY                                    1995/11/28    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
      CICSPDN1.TRANGRPS.MERGE1
        TS.00000001  (LUNAME PCONV    )    TS.00000002  (LUNAME PCONV    )
      CICSPDN1.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
      CICSPDN1.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
      CICSPDN1.TRANGRPS.ONE
        CW.00000001  (GLOBAL SYSTEM   )
```

*Figure 14. Sample group merge report*

## Error report

This report gives the syntax or logic of any errors that were detected in the processing of the input files. Each error is accompanied by a message. For a description of the message, see the *CICS Messages and Codes* manual.

```
CICS TRANSACTION AFFINITIES UTILITY                                          1996/02/08    Page     1
BUILDER REPGRPS ERROR REPORT
Dataset = CICSPDN1.TRANGRPS.ERR1
          Line Number  Statement in error
          -----------  ------------------------------------------------------------------------------
                   5   CREATE TRANGRP NAME(G3         ) AFFINITY(GLOBAL) AFFLIFE(LOGON    );
                       DFHAU5038  INVALID AFFLIFE for AFFINITY.
                   6   CREATE TRANGRP NAME(G4         ) AFFINITY(GLOBAL) AFFLIFE(SIGNON   );
                       DFHAU5038 INVALID AFFLIFE for AFFINITY.
                   7   CREATE TRANGRP NAME(G5         ) AFFINITY(GLOBAL) AFFLIFE(PCONV    );
                       DFHAU5038 INVALID AFFLIFE for AFFINITY.
Dataset = CICSPDN1.TRANGRPS.ERR2
          Line Number  Statement in error
          -----------  ------------------------------------------------------------------------------
                  11   CREATE TRANGRP NAME(L4         ) AFFINITY(LUNAME) AFFLIFE(SIGNON   );
                       DFHAU5038 INVALID AFFLIFE for AFFINITY.
                  15   CREATE TRANGRP NAME(U3         )
                  16          AFFINITY(USERID) AFFLIFE(LOGON    );
                       DFHAU5038 INVALID AFFLIFE for AFFINITY.
```

*Figure 15. Sample error report*

# 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 cannot see beyond the command argument zero to eliminate commands that do not actually cause affinity.)

This information adds details to the general description in “What is detected” on page 13.

This chapter contains the following information:



## 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).
- The affinity relation can be GLOBAL, BAPPL, or USERID.
- Lifetime is always SYSTEM.
- Commands that result in a LENGERR condition are grouped together and treated as a resource of ’LENGERR’. Any other condition results in a valid resource and does not affect the treatment of the 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 than those at either end). This means that such resources may be flagged as being the same when they are not, if the only variation is in the removed bytes. 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.

## 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.
- The affinity relation can be GLOBAL, BAPPL, LUNAME, or USERID.
- Lifetime can be PCONV, LOGON, SIGNON, ACTIVITY, PROCESS,SYSTEM, and PERMANENT. A MAIN queue cannot be recovered, regardless of definition, so cannot cause PERMANENT.

- No data is collected if a TS queue is defined as remote or if a remote SYSID is specified on the TS command. In such cases, the request is satisfied by a remote CICS region or by a temporary storage pool in the coupling facility.
- Commands in error are treated in the same way as commands that give a NORMAL response, so data is collected.
- If a TS queue is created and deleted within the same task, 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 applies only to programs that are defined with RELOAD(NO). If the Detector can not establish the RELOAD attribute for some reason, 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.
- The affinity relation is GLOBAL or BAPPL.
- Lifetime is always SYSTEM.
- Commands in error are treated in the same way as commands that give a NORMAL response, so data is collected.
- LOAD with no HOLD for programs defined as RESIDENT is 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.
- The affinity relation can be GLOBAL or USERID.
- Lifetime can be 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, no data is collected.
- Commands in error are treated in the same way as commands that give a NORMAL response, so data is collected.

*Scanner differences*: Scanner detects 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 affinity relation is GLOBAL or BAPPL.
- 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, BAPPL,LUNAME, or USERID.
- Lifetime can be PCONV, LOGON, SIGNON, ACTIVITY, PROCESS, or 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 for matching GETMAIN with FREEMAIN, 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 so initiated 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 applies only to programs defined as RELOAD(YES). Note that HOLD is irrelevant, as CICS Program Control never sees the FREEMAIN, or knows the storage location of the individual task's copy, and so cannot 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, BAPPL, LUNAME, or USERID.
- Lifetime can be PCONV, LOGON, SIGNON, ACTIVITY, PROCESS,or 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, because 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 so initiated 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.

*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 cannot be canceled, and therefore cannot be detected.

  In order for another task to CANCEL a START or POST, it is not necessary to specify REQID on the command because 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 canceled (because an element control interval (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 cannot cancel another task. CANCEL commands that return a NOTFND response are also ignored because the ICE must have expired and the CANCEL must have failed. No data is collected for these.
- REQIDs are assumed to be unique; that is, there are no simultaneous pairs of START/CANCEL using the same REQID. Having such a pair violates CICS programming guidelines, and the results from CICS are unpredictable.
- The affinity relation for START may be GLOBAL, BAPPL, LUNAME, or USERID.
- Lifetime can be PCONV, LOGON, SIGNON, ACTIVITY, PROCESS, SYSTEM, or 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, BAPPL, LUNAME, or USERID.

- Lifetime can be only SYSTEM, PROCESS, ACTIVITY, or PCONV. If the affinity relation is LUNAME or USERID, the lifetime 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, 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 so initiated 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, CREATE, DISCARD, ENABLE, DISABLE, EXTRACT EXIT, COLLECT STATISTICS, PERFORM, and RESYNC.
- CBTS BROWSE COMMANDS are treated as inquire COMMANDS.
- The affinity here is not an affinity between transactions, but rather an affinity with the system on which the command was issued; 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.

*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 the affinity can be detected.
- This means affinity transaction groups cannot be created, because the affinity degenerates to an affinity with the system on which the WAIT command was issued; 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, 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 CICS command argument zero.[6] This is a constant and is therefore located in the literal pool for the program. As with the Reporter, 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, match the argument zero to the command, which involves finding the instruction that 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.

6. For an explanation of **argument zero**, see “Notes on terminology” on page ix.

## 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 16 on page 67.

For the Scanner output:

```
CICS TRANSACTION AFFINITIES UTILITY                                          1995/11/19  Page     1
LOAD MODULE SCANNER - DETAILED LISTING OF CICS.PRODN1.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 16. Example for finding an EXEC CICS command from the argument zero*

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.

# Appendix C. Useful tips when analyzing Transaction Affinities Utility reports

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 for resolving 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 causes affinity only if the program is not reentrant; that is, if it modifies itself. Otherwise, there is no affinity.

CICS COBOL run-time code writes data to a TS queue if an abnormal termination occurs. The TS queue name used is “CEBR” plus the termid of the terminal, or blanks if there is 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 pseudoconversation, 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 pseudoconversation, the 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 pseudoconversation and causing a VTAM® line error. This causes an error transaction to be attached internally at the terminal. The affinity utility program 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 pseudoconversation, the data in it will never be used again. Normally the first action of a new pseudoconversation is to delete the contents of all such TS queues for that terminal to ensure that everything is tidy.

## Unrecognized Transids

Transids that consist of garbage data are reported in the 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. Diagnostics

This appendix contains these sections:



## 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 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 | 21 |
| AUTM_UT | 22 |
| 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 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 error occurs on a call to the Transaction Affinities Utility date formatter, CAUCAFDT.

## Reason code values

| | |
|---|---|
| CAUD_NO_DATE | 1 |
| CAUD_FORMAT | 2 |

# Index

## G



## H



## I



## L



## P



## R



## S

# T



# U



# V

# Sending your comments to IBM

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 ask questions, make comments about the functions of IBM products or systems, or to request additional publications, contact your IBM representative or 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, to this address:

  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER
  Hampshire
  SO21 2JN
  United Kingdom
- By fax:
  - From outside the U.K., after your international access code use 44–1962–842327
  - From within the U.K., use 01962–842327
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink™: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Program Number: 5655-147

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-1777-30

Spine information:

CICS TS for OS/390 CICS Transaction Affinities Utility Guide Release 3