CICS® Transaction Server for
OS/390®

IBM

CICS Resource Definition Guide

Release 3

SC33-1684-33

CICS® Transaction Server for OS/390®

IBM

CICS Resource Definition Guide

Release 3

SC33-1684-33

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, SC33-1684-02. 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

Trademarks

Preface

What this book is about

This book tells you how to define the characteristics of your data processing resources to your CICS® system. It describes four methods of resource definition:
- Online definition (CEDA)
- Batch definition (DFHCSDUP)
- Automatic installation (autoinstall)
- Macro definition

A fifth method—EXEC CICS CREATE commands—is described in the *CICS System Programming Reference* manual.

Who should read this book

This book is for those responsible for defining resources to CICS.

What you need to know to understand this book

This book assumes that you have a basic understanding of CICS concepts and facilities. You must also be familiar with your own system and the resources to be defined and maintained.

How to use this book

"Part 1. Introduction" on page 1 is an introduction to resource definition for CICS; read it if you are new to CICS. It also introduces the CICS system definition (CSD) file.

"Part 2. Resource definition online (RDO)" on page 21 tells you how to use resource definition online, and provides guidance and reference for using the CEDA online transaction.

"Part 3. The CSDUP offline utility" on page 73 tells you how to use the DFHCSDUP offline utility.

"Part 4. Autoinstall" on page 117 explains automatic installation (autoinstall); read this if you want to make use of the benefits offered by autoinstall.

"Part 5. RDO resource types and their attributes" on page 151 describes each of the resources definable by CEDA, DFHCSDUP, and autoinstall. It is intended as reference material.

"Part 6. Macro resource definition" on page 365 explains how to define resource by macro tables. Use it for reference if you use macro definitions.

Notes on terminology

When the term "CICS" is used without any qualification in this book, it refers to the CICS element of IBM CICS Transaction Server for OS/390®.

"CICS/ESA®" is used for IBM® Customer Information Control System/Enterprise System Architecture.

Other abbreviations that may be used for CICS releases are as follows:

CICS/MVS® Version 2 Release 1 and subsequent modification levels—CICS/MVS 2.1

CICS/ESA Version 3 Release 3—CICS/ESA 3.3

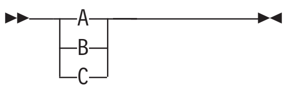
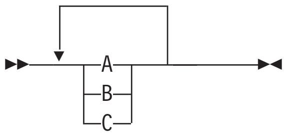
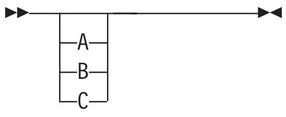
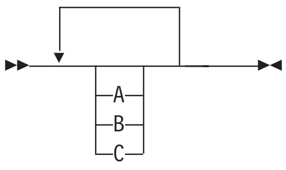
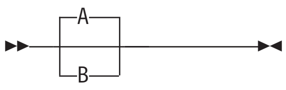
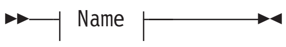
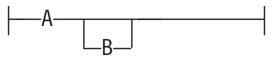
CICS/ESA Version 4 Release 1—CICS/ESA 4.1.

$ (the dollar symbol)

In the character sets given in this book, the dollar symbol ($) is used as a national currency symbol and is assumed to be assigned the EBCDIC code point X'5B'. In some countries a different currency symbol, for example the pound symbol (£), or the yen symbol (¥), is assigned the same EBCDIC code point. In these countries, the appropriate currency symbol should be used instead of the dollar symbol.

Railroad syntax notation

The syntax diagrams are in railroad notation. You interpret the syntax by following the arrows from left to right. The conventions are:

Symbol	Action
	A set of alternatives—one of which you ***must*** code.
	A set of alternatives—one of which you ***must*** code. You ***may*** code more than one of them, in any sequence.
	A set of alternatives—one of which you ***may*** code.
	A set of alternatives — any number (including none) of which you may code once, in any sequence.
	Alternatives where **A** is the default.
 **Name:** 	Use with the named section in place of its name.
Punctuation and uppercase characters	Code exactly as shown.
Lowercase characters	Code your own text, as appropriate (for example, `name`).

For example, with `DEFINE FILE(filename)` you must specify `DEFINE FILE` and `()` unchanged, but are free to specify any valid text string to mean the name of the file.

Macro syntax notation

The operand field is used to specify the services and options to be generated. Operands are always in a keyword format and any parameters are specified according to the following general rules:

- If the parameter associated with the operand is shown entirely in capital letters (for example, TYPE=INITIAL), the operand and parameter must be specified exactly as shown.
- If the parameter associated with the operand is shown in lowercase letters (for example, FILE=name), the operand must be specified exactly as shown and a value, address, or name must be substituted for the lowercase letters.
- Commas and parentheses must be specified exactly as shown, except that a comma following the last operand specified must be omitted.
- The parentheses may be omitted when only one parameter of a particular operand is used.
- Because a blank character indicates the end of the operand field, the operand field must not contain blanks except within quotes, after a comma on a continued line, or after the last operand of the macro. The first operand on a continuation line must begin in column 16.
- When a CICS macro is written on more than one line, each line containing part of the macro (except the last line) must contain a character (for example, an asterisk) in column 72, indicating that the macro is continued on the next line.

The symbols [ ], { }, | and ,... are used in this book to show the operands for the macros as clearly as possible. **Do not use these symbols in your specifications**. They act only to indicate how a command or macro can be written; their definitions are given below:

[ ]	indicates optional operands. You may or may not need to specify the operand enclosed in the brackets (for example, [FB]), depending on whether the associated option is desired. If more than one item is enclosed within brackets (for example, [BLOCKED|UNBLOCKED]), you can specify either one or none. Any default value available is indicated like this: **default**. The default is taken if you do not specify an option from the group.
{ }	indicates that a choice must be made. One of the operands from the list within braces separated by a | symbol (for example, {YES|NO}) must be specified, depending on which of the associated services is desired. Any default value is indicated like this: **default**.
|	indicates that a choice must be made between the operands that are separated by this symbol.
,...	indicates that more than one set of operands can be designated.

To simplify the syntax notation in the case where one or more operands may be specified, the notation:

```
PARM=([A][,B][,C][,D])
```

indicates that any number or none of A, B, C, or D may be specified. Do not code a leading comma. If you specify only one operand, you need not code the enclosing parentheses.

For example:

```
PARM=A
PARM=(A,B)
PARM=(B,D)
PARM=(C)
```

are all valid interpretations of the above notation.

The default value of an operand is shown like this: **default**. For example:

```
PARM={A|B|C}
```

means that you can code A, B, or C as the value for this operand. If you do not code the operand at all, the value A is assumed.

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.

Books from related libraries

This section lists the non-CICS books that are referred to in this manual.

ACF/TCAM books

- *ACF/TCAM Version 3 Application Programming*, SC30-3233

MVS books

See the relevant books in the current MVS library.

Miscellaneous books

- *DATABASE 2 Version 2 Administration Guide*, SC26-4374
- *IBM 3290 Information Panel Description and Reference*, GA23-0021
- *OS/390 Language Environment Programming Guide*, SC28-1939
- *Resource Access Control Facility (RACF) Security Administrator's Guide*, SC28-1340.
- *DPPX/Distributed Presentation Services Version 2: System Programming Guide*, SC33-0117

Determining if a publication is current

IBM regularly updates its publications with new and changed information. When first published, both hardcopy and BookManager softcopy versions of a publication are usually in step. However, due to the time required to print and distribute hardcopy books, the BookManager version is more likely to have had last-minute changes made to it before publication.

Subsequent updates will probably be available in softcopy before they are available in hardcopy. This means that at any time from the availability of a release, softcopy versions should be regarded as the most up-to-date.

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

Updates to the softcopy are clearly marked by revision codes (usually a "#" character) to the left of the changes.

Summary of changes

Significant changes for this edition are indicated by vertical bars to the left of the change.

Changes for the CICS Transaction Server for OS/390 Release 3 edition

- The following new resource type are introduced:
  - DOCTEMPLATE
  - ENQMODEL
  - PROCESSTYPE
  - REQUESTMODEL
  - TCPIPSERVICE
  - TSMODEL

  CEDA and DFHCSDUP commands are modified to support them.
- A remove option is added to the DELETE and MOVE commands.
- The MIGRATE command is enhanced so that you can migrate existing TST macro definitions to TSMODEL resource definitions.
- Two new fields, JVM and JVMClass, have been added to the CEDA DEFINE PROGRAM panel, to support running Java™ applications under the control of a JVM.
- The USERDEFINE command is made available from DFHCSDUP

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

- The BREXIT option is added to the TRANSACTION definition to support the 3270 bridge.
- The FACILITYLIKE option is added to the PROFILE definition to support the 3270 bridge.
- The CICS DB2® attachment facility supplied with CICS Transaction Server is enhanced to provide RDO support for DB2 resources, as an alternative to the DSNCRCT macro. Thus the DB2CONN, DB2ENTRY, and DB2TRAN resource definition types are added to the resources defined in the CICS CSD. In this release they offer alternatives to specifying DSNCRCT macros to create a Resource Control Table (RCT), but RDO definition of these resources is expected to be the only method offered in future releases.
- A new command, SCAN, is added to DFHCSDUP to search all the IBM-supplied and user-defined groups in the CSD for a resource definition of a specified name and type.

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

A number of editorial changes have been made to improve the layout of this book.

- The MIGRATE command is enhanced so that you can migrate existing DCT macro definitions to RDO format.
- The UPGRADE command is enhanced to include definitions for all transient data queues that CICS uses for error messages and audit trails.
- The COPY command in DFHCSDUP is enhanced to allow the specification of generic names in both the Resource name and the Group name.

- Group DFHDCTG, containing definitions for all the CICS-supplied transient data queues, is added. The DFHCSDUP INITIALIZE and UPGRADE commands are extended to include DFHDCTG.
- The CEDA transaction has been enhanced to link to the CMAC transaction (Messages and Codes) from the Expanded messages panel.

Changes for the CICS/ESA 4.1 edition

The main changes for CICS/ESA 4.1 were:

Changes to resource definition online (RDO)

- There are seven new attributes on the TRANSACTION definition:
  - CONFDATA
  - ISOLATE
  - RUNAWAY
  - SHUTDOWN
  - STORAGECLEAR
  - TASKDATAKEY
  - TRANCLASS
- The TCLASS attribute on the TRANSACTION definition is now obsolete, and is replaced by the TRANCLASS attribute.
- A new resource type, TRANCLASS, has been added.
- The meaning of EXECKEY on the PROGRAM definition has been changed.
- There are five new attributes on the CONNECTION definition:
  - CONNTYPE
  - MAXQTIME
  - QUEUELIMIT
  - PSRECOVERY
  - REMOTESYSNET
- A new value, EXCI, on the PROTOCOL attribute of the CONNECTION definition.
- A new value, EXCI, on the PROTOCOL attribute of the SESSIONS definition.
- Removal of the RECOVNOTIFY attribute from the SESSIONS definition.
- Changes to the RECOVOPTION attribute of the SESSIONS definition.
- Changes to the RECOVNOTIFY and RECOVOPTION attributes of the TYPETERM definition.
- CEDA gives the current resource name for the ALTER, DEFINE, and VIEW commands.
- Before-images of a CEDA ALTER command are logged to the CSDL queue.
- New and changed CICS-supplied resource definitions.
- The introduction of autoinstall for programs, mapsets, and partitionsets.
- The introduction of autoinstall for APPC connections.
- The system's SYSID is displayed on all RDO panels.
- The CEDA CHECK LIST command is extended to four lists.

Changes to resource definition at macro level

- DFHDCT now operates over a whole CICSPlex®.
- CPU and CONV are removed from DFHMCT TYPE=INITIAL.
- Increase in size of the MOVE option of the PERFORM operand of DFHMCT TYPE=EMP.

- Removal of the DFHPCT macro table.
- Removal of the DFHPPT macro table.
- Removal of VSAM file definition from DFHFCT.
- Removal of data tables definition from DFHFCT.
- Removal of local shared resource pool definition from DFHFCT.

Part 1. Introduction

This part introduces resource definition as it applies to the CICS element of CICS Transaction Server for OS/390. Read it if you are new to CICS or to resource definition. It consists of the following chapters:

Chapter 1. Resource definition—an introduction

Before you can use CICS, you must supply it with information about the resources it should use, and how it should use them. Some examples of resources are:

- Connections
- Databases
- Files
- Journals
- Journalmodels
- Programs
- Terminals
- Transactions
- Transient data queues (destinations)

Your CICS system has to know which resources to use, what their properties are, and how they are to interact with each other.

You supply this information to CICS by using one or more of the following methods of **resource definition**:

1. **Resource definition online (RDO)**: This method uses the CICS-supplied online transactions CEDA, CEDB, and CEDC. Definitions are stored on the CICS system definition (CSD) file, and are installed into an active CICS system from the CSD file.
2. **DFHCSDUP offline utility**: This method also stores definitions in the CSD file. DFHCSDUP allows you to make changes to definitions in the CSD file by means of a batch job submitted offline.
3. **Automatic installation (autoinstall)**: Autoinstall minimizes the need for a large number of definitions, by dynamically creating new definitions based on a "model" definition provided by you.
4. **System programming, using the EXEC CICS CREATE commands**: You can use the EXEC CICS CREATE commands to create resources independently of the CSD file. For further information, see the *CICS System Programming Reference* manual.
5. **Macro definition**: You can use assembler macro source to define resources. Definitions are stored in assembled tables in a program library, from which they are installed during CICS initialization.

Which methods you use depends on the resources you want to define. Table 1 shows you the methods you can use for each resource. Table 2 on page 4 suggests some of the things you should consider when deciding which definition method to use when there is a choice.

Table 1. Resources and how you can define them to the running CICS system

Resource	RDO/EXEC CICS CREATE commands	DFHCSDUP	Autoinstall	Macro
Connections	Yes	Yes	Yes	No
DB2 Connections	Yes	Yes	No	No
DB2 Entries	Yes	Yes	No	No
DB2 Transactions	Yes	Yes	No	No

Table 1. Resources and how you can define them to the running CICS system (continued)

Resource	RDO/EXEC CICS CREATE commands	DFHCSDUP	Autoinstall	Macro
Doctemplate	Yes	Yes	No	No
ENQmodel	Yes	Yes	No	No
Files (BDAM)	No	No	No	Yes
Files (VSAM)	Yes	Yes	No	No
Journals	No	No	Yes	No
Journalmodels	Yes	Yes	No	No
Local shared resource (LSR) pools	Yes	Yes	No	No
Mapsets	Yes	Yes	Yes	No
Partitionsets	Yes	Yes	Yes	No
Partners	Yes	Yes	No	No
Processtype	Yes	Yes	No	No
Profiles	Yes	Yes	No	No
Programs	Yes	Yes	Yes	No
Recoverable service elements	No	No	No	Yes
Requestmodel	Yes	Yes	No	No
Sessions	Yes	Yes	No.	No
Shared data tables	Yes	Yes	No	No
TCPIPservice	Yes	Yes	No	No
Temporary storage (defined by macro)	No	No	No	Yes
TSmodel (resource definition)	Yes	Yes	No	No
Terminals (non-VTAM)	No	No	No	Yes
Terminals (VTAM)	Yes	Yes	Yes	No
Transactions	Yes	Yes	No	No
Transaction classes	Yes	Yes	No	No
Transient data queues (destinations)	Yes	Yes	No	Yes
Typeterms	Yes	Yes	No	No

Table 2. Methods of resource definition

Method	Description	Advantages	Disadvantages
RDO	This method uses the CEDA transaction, which allows you to define, alter, and install resources in a running CICS system.	RDO is used while CICS is running, so allows fast access to resource definitions.	Because CEDA operates on an active CICS system, care should be taken if it is used in a production system. Use some form of auditing as a control mechanism.

Table 2. Methods of resource definition (continued)

Method	Description	Advantages	Disadvantages
EXEC CICS CREATE system commands	This method allows you to add CICS resources to a CICS region without reference to the CSD file.	It enables configuration and installation of CICS resources for large numbers of CICS regions from a single management focal point. It also allows you to write applications for administering the running CICS system.	CREATE commands neither refer to nor record in the CSD file. The resulting definitions are lost on a cold start, and you cannot refer to them in a CEDA transaction.
DFHCSDUP	DFHCSDUP is an offline utility that allows you to define, list, and modify resources using a batch job. DFHCSDUP can be invoked as a batch program or from a user-written program running either in batch mode or under TSO. Using the second method, you can specify up to five user exit routines within DFHCSDUP.	• You can modify or define a large number of resources in one job. • You can run DFHCSDUP against a non-recoverable CSD file while it is being shared between CICS regions using RLS access mode.	• You cannot install resources into an active CICS system. • You cannot make updates via DFHCSDUP against a recoverable CSD file that is being accessed in RLS mode.
Autoinstall	This applies to VTAM® terminals, LU6.2 sessions, journals, programs, mapsets, and partitionsets. You set up “model” definitions using either RDO or DFHCSDUP. CICS can then create and install new definitions for these resources dynamically, based on the models.	If you have large numbers of resources, much time is needed to define them, and if they are not all subsequently used, storage is also wasted for their definitions. Using autoinstall reduces this wasted time and storage.	You must spend some time initially setting up autoinstall in order to benefit from it.
Macro	Using this method, you code and assemble macroinstructions to define resources in the form of tables.	Where possible, use the other methods.	• You can change the definitions contained in the tables while CICS is running, but you must stop and restart CICS if you want it to use the changed tables. • You must do time-consuming assemblies to generate macro tables.

Chapter 2. An overview of resource definition

This chapter contains the following information:

The following list shows you what you must do to begin defining resources. If you are new to CICS, all the items in the list may apply to you; if you are an existing CICS user but are new to RDO, some items may not apply to you.

1. **Create and initialize a CSD file**. If you are a new CICS user, you must create and initialize a CICS system definition (CSD) file. See the *CICS System Definition Guide* for information on how to do this.
2. **Migrate existing definitions**. If you are moving to CICS Transaction Server for OS/390 from an earlier release and you have any of the following definitions, you **must** migrate them to the CSD file within the earlier release, **before** migrating to CICS Transaction Server for OS/390:
   - All of your PCT
   - All of your PPT
   - All FCT entries except those for BDAM files
   - VTAM TCT entries

   CICS Transaction Server for OS/390 does not support macro definition of these resources.

   | RCT macro definitions are not supported in ths release, except for migration. They must be migrated to the CSD either in CICS Transaction Server for OS/390 Release 2 or in CICS Transaction Server for OS/390 Release 3. The MIGRATE command (see “MIGRATE” on page 102) will help you to do this.

   | Furthermore, it is recommended that you migrate your DCT and TST tables to the CSD file, because macro DCTs and TSTs will be withdrawn in a future release of CICS. The MIGRATE command (see “MIGRATE” on page 102) will help you to do this.
3. | **Run a DFHCSDUP UPGRADE job**. Do this to bring the CICS supplied definitions in your CSD file up to the level of CICS Transaction Server for OS/390 function. See “UPGRADE” on page 111 for information on how to do this.
4. **Work with your resource definitions**. Use Table 1 on page 3 and Table 2 on page 4 to help you decide which methods to use to define and manage your resources. If you are an existing CICS user, some of these decisions will have been made for you already.

   When you know which methods you want to use, read the following parts:
   - If you want to use CEDA, read “Part 2. Resource definition online (RDO)” on page 21 for guidance information, and “Part 5. RDO resource types and their attributes” on page 151 for reference information.

- If you want to use DFHCSDUP, read “Part 3. The CSDUP offline utility” on page 73 .
- If you want to use autoinstall, read “Part 4. Autoinstall” on page 117.
- If you want to use macro definitions, read “Part 6. Macro resource definition” on page 365.

The CICS system definition (CSD) file

The CICS system definition (CSD) file is a VSAM data set containing a resource definition record for every resource defined to CICS by means of CEDA or DFHCSDUP. It can be defined as recoverable, so that changes made by CEDA or CEDB that were incomplete when an abend occurred, are backed out.

Creating a CSD file

If you do not already have a CSD file, you must create one. For detailed information about creating a CSD file, see the *CICS System Definition Guide*.

You can create more than one CSD file, depending on your requirements. For example, you can have different CSD files for different systems, so that your test systems and production systems are separate from each other.

You can also share one CSD file between CICS releases; see “Compatibility mode (CSD file sharing)” on page 12.

When the CSD file has been initialized, it contains a number of groups (all beginning with the letters ‘DFH’) containing related resource definitions, and one list, called DFHLIST. These definitions are supplied by CICS and are necessary for some system functions and for running CICS-supplied transactions.

How the CSD file is organized—groups and lists

Information on the CSD file is organized into **groups** and **lists**. The main purpose of the **group** is to provide a convenient method of collecting related resources together on the CSD file. Every resource that you define must belong to a group; you cannot define a resource without also naming its group.

A **list** contains the names of groups that CICS installs at an initial or cold start. You can add groups to lists if you want them to be installed at an initial or cold start, or if it helps you to manage your groups better. Groups do not have to belong to lists, and can be defined independently.

Creating groups and lists

A group is created when you specify it as the GROUP name in a DEFINE command or as the TO group in a COPY command. For example, the command:

```
CEDA DEFINE PROGRAM(PROG1) GROUP(MYGROUP)
```

defines a program called PROG1, and creates a group called MYGROUP if it does not already exist. See page 45 for information on the COPY command, and page 48 for information on the DEFINE command.

These are the only ways to create a group; a nonexistent group can be named in a list, but naming it in a list does not create it.

A group must not have the same name as an existing group or list.

You can create a list in either of the following ways:

- Use the ADD command to add a group to a list. If the specified list does not exist, it is created. See page 38 for information about the ADD command.
- Use the APPEND command to append the contents of one list to another list. If the appended-to list does not exist, it is created, containing the contents of the first list. See page 41 for information about the APPEND command.

A list must not have the same name as an already existing group or list.

Managing resource definitions

You manage your resource definitions using commands supplied as part of CEDA or DFHCSDUP. These commands allow you to work with your resources, for example, by defining, deleting, copying, and renaming.

The commands are listed in Table 3. For the syntax of these commands and information on how to use them, see “Chapter 4. CEDA commands—syntax and examples” on page 37 and “Chapter 6. DFHCSDUP commands” on page 85. To help you use CEDA, see “Chapter 3. CEDA tutorial” on page 23.

CEDA and DFHCSDUP commands

Table 3. CEDA and DFHCSDUP commands

Command	**Function**	**CEDA—see page**	**DFHCSDUP—see page**
ADD	Adds a group name to a list.	38	115
ALTER	Modifies the attributes of an existing resource definition.	39	87
APPEND	Copies a list to the end of another list.	41	89
CHECK	Cross checks the resource definitions within a group, or within the groups in a list or lists, up to a maximum of four lists. The CHECK command is CEDA only.	42	—
COPY	Copies one or more resource definitions from one group to another, or one resource definition within a group.	45	91
DEFINE	Creates a new resource definition.	48	93
DELETE	Deletes one or more resource definitions.	50	95
DISPLAY	Shows the names of one or more groups, lists, or resource definitions within a group. The DISPLAY command is CEDA only.	52	—
EXPAND	Shows the names of the resource definitions in one or more groups or lists. The EXPAND command is CEDA only.	54	—
EXTRACT	Extracts and processes resource definition data from groups or lists on the CSD file. The EXTRACT command is DFHCSDUP only.	—	97
INITIALIZE	Prepare a newly-defined data set for use as a CSD file. The INITIALIZE command is DFHCSDUP only.	—	108
INSTALL	Dynamically adds a resource definition or a group of resource definitions to the active CICS system. The INSTALL command is CEDA only.	56	—

Table 3. CEDA and DFHCSDUP commands (continued)

Command	Function	CEDA—see page	DFHCSDUP—see page
LIST	Produce listings of the current status of the CSD file. The LIST command is DFHCSDUP only.	—	100
LOCK	Prevents other operators updating or deleting a group or the groups in a list. The LOCK command is CEDA only.	58	—
MIGRATE	Transfers the contents of a terminal control table (TCT), temporary storage table (TST), file control table (FCT), or destination control table (DCT) to the CSD. The MIGRATE command is DFHCSDUP only.	—	102
MOVE	Moves one or more resource definitions from one group to another. The MOVE command is CEDA only.	60	—
PROCESS	Applies maintenance to the CSD file for a specific APAR. The PROCESS command is DFHCSDUP only.	—	106
REMOVE	Removes a group name from a list.	62	107
RENAME	Renames a resource definition, either within a group, or while simultaneously moving it to another group. The RENAME command is CEDA only.	63	—
SCAN	Scans all of the IBM-supplied groups and user-defined groups for a resource. The definition of the matched resource in an IBM-supplied group is compared to the definition(s) of the corresponding matched resource in the user groups.	—	108
SERVICE	Applies corrective maintenance to the CSD file. The SERVICE command is DFHCSDUP only.	—	110
UNLOCK	Releases a lock on a group or list. The UNLOCK command is CEDA only.	65	—
UPGRADE	Upgrades the CICS-supplied resource definitions on the CSD file (for example, when you migrate to a higher release of CICS). The UPGRADE command is DFHCSDUP only.	—	111
USERDEFINE	Creates a new resource definition with your own defaults.	67	113
VERIFY	Removes internal locks on groups and lists. The VERIFY command is DFHCSDUP only.	—	115
VIEW	Shows the attributes of an existing resource definition. The VIEW command is CEDA only.	71	—

How many resource definitions should a group contain?

Try to keep your groups to a manageable size; ideally, there should be no more than about 100 resource definitions in a group. Allocate your resource definitions between groups to obtain optimum performance, in both system and administration terms. The following considerations may help:

- A large group can involve a lot of unnecessary processing time to install. This is particularly true of those containing TERMINAL and SESSIONS definitions, because they take a large amount of dynamic storage.
- A large number of very small groups can also use unnecessary processing time, because of the extra I/O involved in reading many group names from the CSD file. In theory, you could have one resource definition per group, but this is not recommended; the processing of a large number of single-resource groups can affect DASD space, initial or cold start performance, and the performance of both CEDA and DFHCSDUP.
- Administration is easier if you have smaller groups. For example, the DISPLAY GROUP ALL command involves a lot of scrolling if the resource definitions in the group extend over many screens. You cannot see at a glance the contents of a large group.
- You may find that you have storage problems when you EXPAND, COPY, or INSTALL a large group. In particular, if a very large number of CSD file records are defined in a region with a small dynamic storage area, issuing a CEDA EXPAND GROUP(*) command can result in the system going short on storage (SOS).

What should be in a group?

Usually, the definitions within a group have something in common. For example:

For application resources:

- It is more convenient to keep all the resource definitions belonging to one application in one group.
- If you use PARTITIONSET or PROFILE definitions for many applications, keeping them separate in their own groups avoids the possibility of unnecessary duplication.

For communication resources:

- SESSIONS definitions **must** be in the same group as the CONNECTION definition to which they refer. You may have more than one group of definitions for each system and its sessions with other systems, in a single CSD file that is shared by all the systems. Be careful that you install each group of definitions in the correct system.
- Restrict a group to contain only one CONNECTION definition with its associated SESSIONS definitions.
- Keep all your TYPETERM definitions in one group. This avoids the possibility of unnecessary duplication. You **must** put the group of TYPETERMs before the groups of TERMINAL definitions in your lists.
- It is convenient to group TERMINAL definitions according to departmental function or geographical location.
- You **must** keep all the TERMINAL definitions for one pool of pipeline terminals in the same group.
- Keep AUTINSTMODEL TERMINAL definitions separately in a group of their own.

For transient data resources, sample definitions for the CICS-supplied transient data queues (those beginning with the letter “C”) are provided in group DFHDCTG. For these definitions to become available for use at the earliest possible point during CICS initialization, include group DFHDCTG as the first group installed during an initial or cold start.

Compatibility mode (CSD file sharing)

CICS allows a CSD file and its resource definitions to be shared between different CICS systems. The systems might be running the same or different releases of CICS. **Compatibility mode** is intended for use when you want to create or change resource definitions on a CSD file that is shared between different releases.

All maintenance should be done under the latest release of CICS. This avoids the risk of earlier releases modifying entries created under more recent releases with new attributes that the older version does not recognize. Ensure this by restricting write access to the CSD file to the latest release. See the *CICS System Definition Guide* for further details on defining CSD files.

Compatibility mode is entered by using PF2 on the CEDA panels where it is available. It gives you access to those attributes that were current at your earlier release, but are obsolete at your later release. However, you can use compatibility mode only with commands affecting individual resources: you cannot perform generic commands (ALTER, DEFINE, and VIEW) in compatibility mode.

There is more information about issues relating to compatibility mode in the following places:

- For the usage and meaning of attributes and their compatibility with previous releases of CICS, see "Appendix A. Obsolete attributes retained for compatibility" on page 497.
- For information about what compatibility groups you need in your startup group list for CSD file sharing to work, see Table 94 on page 508, "Sharing the CSD between different releases of CICS" on page 509 and the *CICS Transaction Server for OS/390 Migration Guide* which has a table showing the DFHCOMPx groups you need to include for the earlier releases.
- If you need more detailed information about migration and compatibility, this is also available in the *CICS Transaction Server for OS/390 Migration Guide*.

Installing resource definitions

When a resource definition is installed, the information on it is moved into CICS storage, to become an entry in a "table". This does not necessarily mean that all the information is stored in a large, contiguous area; the tables are virtual, but it is convenient to refer to them in this way.

The following sections explain what happens to resource definitions:
- When CICS is initialized
- When you use the INSTALL command

What happens when CICS is initialized

When you initialize CICS, what happens to your resource definitions depends on the type of start. This is defined in the START system initialization parameter; START=INITIAL or an initial start, START=COLD for a cold start, and START=AUTO for a warm or emergency restart.

Initial or cold start

During an initial or cold start, CICS creates system tables by installing groups named in the list or lists named by the GRPLIST system initialization parameter. If you installed a group with the INSTALL command during the previous CICS execution, you must add its name to a list if you want it to be installed during a cold start.

If you usually use START=COLD at CICS initialization, installing by means of a list will probably be your standard way of making resource definitions available to CICS. Use of the INSTALL command is a supplementary method, which you could find very useful when testing a system, or if an unexpected need for a resource arises when CICS is running.

You may not want to use the RDO transactions in a production system, for security or performance reasons. In this case, the CSD file is shared by both systems, but is read-only in the production system. You define all your production resources using your development system, and install them in the production CICS system when you cold start it.

Warm or emergency start

During a warm or emergency start, CICS recreates the tables from the resource definitions stored in the system log and global catalog.

No reference is made to the CSD file, nor is the GRPLIST name used. So all groups that had been installed by the end of the previous CICS execution are reinstalled automatically at a warm or emergency restart. Thus any CICS system modifications you have introduced using RDO will persist. For autoinstalled resources, see the following:

- "What happens at CICS restart" on page 129
- "Connection autoinstall and recovery and restart" on page 144
- "Program autoinstall and recovery and restart" on page 148

If you have named a different list in the GRPLIST operand, or if you have added new groups to it after the last system initialization, CICS does not install the groups in the new list during a warm or emergency restart, because CICS does not refer to the list.

If you usually use START=AUTO at CICS initialization, using the INSTALL command is your standard way of making resource definitions available to CICS. You use a list to define your system only when you need to do an initial or cold start. You can ensure that your list is up to date by adding to it each group installed using the INSTALL command.

What happens when you use the INSTALL command

CICS has two ways of installing and committing resource definitions:

- Some VTAM terminal control resource definitions must be installed in groups and are committed in **installable sets**.
- Other resource definitions can be installed in groups or individually, and are committed at the individual resource level.

Some terminal control resource definitions within a CSD group are committed at the installable set level. An installable set comprises those resources, such as a CONNECTION and its associated SESSIONS, which are dependent in some way. The backout of an installable set does not cause the whole group to be backed out.

The following types of resource definition are installed in installable sets:

- CONNECTION and associated SESSIONS definitions
- Pipeline terminals—all the terminal definitions sharing the same POOL name

If a member of an installable group fails to install, CICS issues message DFHZC6216 identifying the member that caused the installation of the set to fail.

Installing resource definitions

The following resource types are committed by **individual resource** and not by installable set:
- DB2CONN
- DB2ENTRY
- DB2TRAN
- DOCTEMPLATE
- ENQMODEL
- FILE
- JOURNALMODEL
- LSRPOOL
- MAPSET
- PARTITIONSET
- PARTNER
- PROCESSTYPE
- PROFILE
- PROGRAM
- REQUESTMODEL
- TCPIPSERVICE
- TDQUEUE
- TERMINAL (non-pipeline terminals)
- TRANCLASS
- TRANSACTION
- TSMODEL
- TYPETERM

For these resources, the effect of a partially successful group INSTALL is to leave the resources that were added in a committed state.

For the installation of installable sets and for individual definition, an INSTALL may not be successful for one of two reasons:
1. A resource definition could not be installed because it is currently in use.
2. A system failure occurred during installation.

You can use the CEDA INSTALL command to reinstall the same group used at a cold or initial start, and the new definitions are installed successfully, even if some original definitions are in use and fail to install.

If you have several CICS systems that share the same CSD file, do not install a group of resources in the wrong system.

Installing a limited number of data definitions

If you wish to install only a few new or changed definitions, install single resources as described on page 56. (Note that the single-resource INSTALL of some CONNECTIONs and SESSIONS is not possible.) Use of the single-resource INSTALL eliminates the problems of a partial INSTALL caused by a failure.

However, if you wish to change or add a larger number of definitions, you might prefer to install a new group. In that case, the following considerations apply:
- When you install a group containing an updated definition of an existing resource, the installation fails if the resource is being used at the time. Make sure that none of the resources in a group is in use before trying to install the group.
- Installation is a two-stage process: any existing definition for the resource must be “deleted” from the system tables before a definition can be installed. This can result in more than one message if the “deletion” fails and causes the installation to fail.

- If you have several CICS systems that share the same CSD file, you must be careful not to install a group of resources in the wrong system.

Dual-purpose resource definition

Dual-purpose definition of a resource is a full definition of a resource that is installed in both the local and the remote CICS systems. Instead of duplicating the definitions in the CSD files for the local and remote systems, you create a dual-purpose definition of that resource in a CSD file that is shared by the local and remote systems.

This has the advantage of reducing disk storage and maintenance, because you require only one CSD file record for each shared resource.

If you decide to use dual-purpose resource definition, you may want to consider reorganizing your resources within your resource definition groups. For example, you might currently have two groups: one containing all the resources for a CICS transaction-owning region (TOR), and one containing all the resources for a CICS application-owning region (AOR).

When you use shared resource definitions, you can have three groups, with the first group containing resources specific to the TOR, the second group containing resources specific to the AOR, and the third group containing resources to be installed in both the TOR and the AOR.

These resources should be defined as both local and remote. When the definition is installed on the TOR, CICS compares the SYSIDNT name with the REMOTESYSTEM name. If they are different, a remote transaction definition is created. When the definition is installed on the AOR, CICS compares the REMOTESYSTEM name with the SYSIDNT name. If they are the same, a local transaction definition is installed.

Dual-purpose resource definition can be used with the following resources:
- Files
- Programs
- # Temporary storage models (TSMODELs)
- Terminals
- Transient data queues (TDQUEUEs)
- Transactions.

Duplicate resource definition names

An RDO-defined definition overrides a macro-defined definition of the same name. For example, if you try to install a definition for a VTAM terminal that has the same name as a non-VTAM terminal, the VTAM terminal entry overwrites the non-VTAM terminal entry.

If you INSTALL a group while CICS is active, the resource definitions in the group override any of the same type and name already installed.

When an existing resource definition is replaced in this way, the statistics associated with the old resource definition are transferred to the new definition. If a PROGRAM definition is replaced, the program is relocated on the library and loaded when the new definition is referenced for the first time. In effect, the new definition implies a NEWCOPY operation. The same rules apply to map sets and partition sets.

Duplicate resource definition names

An exception to these rules can occur with duplicate file definitions. If the file is defined as ENABLED, the later installation of a duplicate fails. However, if the file is defined as DISABLED, the later installation of a duplicate succeeds.

It is probably unwise to have more than one resource definition of the same name on the CSD file, even for different resource types. You must keep PROGRAM, MAPSET, and PARTITIONSET names unique. If you have, for example a PROGRAM and a MAPSET with the same name, only one of them is available to CICS. As far as names are concerned, after installation these definitions are treated as if they were the same resource type.

If two groups in a list contain resource definitions of the same name, of the same
# resource type, CICS uses the definition in the group that is later in the list, for all
# resource types except TDQUEUES. For TDQUEUE definitions, the first definition is
# used.

The only reason why you might have more than one resource definition of the same name is if you have alternative definitions of the same real resource, with different attributes. These resource definitions must be in different groups.

Using lists for initialization

The lists that you name in the GRPLIST system initialization parameter must include all the resource definitions required by CICS. These are supplied by CICS and are added to the CSD file when you initialize it before starting to use RDO. (For further information about this, see the *CICS Operations and Utilities Guide*.)

Setting up lists for initialization

You can specify up to four lists, using specific or generic naming, on the GRPLIST system initialization parameter. The default list is the CICS-supplied list DFHLIST.

To create a list containing both CICS-supplied and your own resource definitions:

1. Start to create the list that you use to initialize CICS, by appending DFHLIST to a new list. For example:
   ```
   CEDA APPEND LIST(DFHLIST) TO(INITLIST)
   ```
   This ensures that all CICS-supplied definitions are installed, whether or not you need to change them.
2. Remove the groups containing definitions for function that you do not require. For example:
   ```
   CEDA REMOVE GROUP(DFHMISC) LIST(INITLIST)
   ```
3. Copy all the resource definitions that you need to change into your own groups. For example:
   ```
   CEDA COPY TRANSACTION(CEDF) GROUP(DFHOPER)  TO(SECTRANS)
   CEDA COPY PROFILE(DFHCICST) GROUP(DFHSTAND) TO(REQMOD)
   ```
   Do **not** rename the copies. You can now use ALTER to change the attributes as necessary. For example:
   ```
   CEDA ALTER TRANSACTION(CEDF) GROUP(SECTRANS)
   ```
4. Add these groups to your list for initialization. For example:
   ```
   CEDA ADD GROUP(SECTRANS) LIST(INITLIST)
   CEDA ADD GROUP(REQMOD)   LIST(INITLIST)
   ```

Make sure that you add this group **after** the DFH groups. Although you now have two definitions for the resources that you have altered, the second definition in the list is the one that will be installed, if you name this list as a GRPLIST parameter when you initialize CICS.

5. Add any other groups containing resource definitions of your own that you want to use, or append other lists. Your list might look like this:

```
DFHBMS
DFHCONS
:
DFHVTAMP
SECTRANS
REQMOD
ZEMAPPL
ZEMCOMM
ZEMTYPES
ZEMTERMS
```

   Note that the group containing the TYPETERMs should come before the groups containing the TERMINAL definitions.

6. Cold start your CICS system, naming the list or lists that you have created in the GRPLIST system initialization parameter. For example:

```
START=COLD,GRPLIST=INITLIST
```

Using several lists

You can create lists that contain different sets of groups so that you can initialize different “flavors” of CICS using the GRPLIST system initialization parameter.

Using different lists at different times

It is recommended that you initialize your CICS system with the START=AUTO system initialization parameter, so that the CICS catalog is used to define the system whenever possible, instead of the list or lists named in the GRPLIST operand. However, if you use CICS differently each time you initialize it, specify the START=COLD system initialization parameter, and specify a different list to define your system every time you initialize CICS. For example, you might have:

- A different list for each day of the week, if the pattern of work is different on each day.
- A list for the CICS used for the day shift, and a list for the CICS used for the night shift.
- A test only list used only when CICS is started up by the system programmers on a day of rest (for example).
- For security reasons, a special list containing groups of restricted resource definitions. You could append this list to your usual one, when these resources are needed.

Consider how you might use the list and group mechanisms with transactions related to a company’s salary operations.

Assume that some transactions used by the salary administrators are used every day. For example, a transaction for handling an employee’s tax details may have to be performed at any time. Other transactions, such as minor weekly or monthly payroll adjustments, are run at predefined intervals, or on specific days or dates. You would therefore not want to include the same mixture of transactions and programs every time the system was started up.

Using lists for initialization

By creating a resource definition group for taxation transactions, and another for payroll transactions, you could add them to different lists to produce the required system tables for different days. In the above example, one list would identify only the taxation group; the other would identify both taxation and payroll groups. You would specify the appropriate list in a system initialization parameter.

Clearly, a real system would have many more groups and lists than this.

Using different lists for different CICS systems

If you are running more than one CICS system in the same MVS™ image, you may use the same CSD file to define your resources to both systems. This helps you to ensure that each system has the same definition of resources where necessary. You probably do not want to use all the same resources in each system, so you could create a list for each system. You name the appropriate list in the system initialization parameter for each system.

For example, you might have two production CICS systems sharing a CSD file. Assume that one production system runs three applications: customer inquiry, billing, and adjustments. Each application has its own resources (programs, map sets, and transactions), so you put the resource definitions in three groups: CUSTINQ, CUSTBILL, and CUSTADJ. Then you add these groups to a list called CICS1A.

Another production system runs two more applications in addition to customer inquiry: customer update and customer credit authorization. For these, you create two more groups (CUSTCRED and CUSTUPDT) and another list called CICS1B.

CICS1B contains the same CUSTINQ group as CICS1A, and it also contains CUSTCRED and CUSTUPDT. If you decide, for performance reasons, to move one of your applications to a different CICS system, all you need to do is add the appropriate group name to the appropriate list. The next time you initialize CICS with this list specified in the GRPLIST system initialization parameter, you install the new group.

Using different lists when you introduce changes

The list with which you initialize CICS is a definition of your system (for RDO resources). When you introduce changes to your resources, it is useful to create a new list, keeping the old list to return to if something goes wrong. Then you can reinitialize CICS with the old list, knowing that everything is as it was previously.

Checking groups and lists of resource definitions for consistency

The CHECK command (see page 42) checks the consistency of definitions within a group or within all of the groups within a list or lists. It does not, however, cross-check every attribute of a resource. You may still get error messages when installing a group, although there were no problems when you used the CHECK command.

If you use the CHECK GROUP command, CEDA cross-checks all of the resources in a specified group to ensure that the group is ready to be used. For example, CHECK might warn you that a transaction definition within the group does not name a program within the same group. (Note, however, that this might not be an error. The group might intentionally be paired with a group that does contain the program, or you may want the program to be autoinstalled, in which case it would not have a definition.)

If you use the CHECK LIST command, CEDA cross-checks every group named in the list. It does not simply check each group in turn, but merges the definitions in all of the listed groups, and checks them all. In this way it warns you if there are duplicate resource definitions, or references to definitions that do not exist.

Security and RDO

This section discusses security for RDO and the CSD file.

Resource security checking

Resource security checking ensures that terminal operators can access only those resources for which they have been authorized. You can use resource security checking (RESSEC) for the TRANSACTION definition.

Multiple CSD files

You can have different CSD files for different CICS systems. The users of one CICS do not have access to the CSD file for another CICS.

You could have a test CSD file in a system where the RDO transactions can be used, and a production CSD file in a system where the RDO transactions are not available. There would then be no chance of unauthorized users altering resource definitions needed for production work.

Read-only and update definitions for the same CSD file

Having two CSD files means duplicating resource definitions for resources that are shared by more than one system. An advantage of RDO is that you need only one definition for each resource. You can define one CSD file to be shared among several CICS systems with only one having write access. To do this, you define one CSD file differently to different systems by using the CSDACC system initialization parameter. For the system where the CSD file can be used but not updated, you specify:

```
CSDACC=READONLY
```

and, for the system where you are planning to update the CSD, you specify:

```
CSDACC=READWRITE
```

You need READONLY access to install definitions. This also allows you to use the DISPLAY and VIEW commands. You need READWRITE access to use the ADD, APPEND, ALTER, COPY, MOVE, and RENAME commands. For information on defining the CSD file, see the *CICS Operations and Utilities Guide*.

Controlling access to a group or list—LOCK and UNLOCK

RDO also provides a means of controlling access to any group or list, so that users in the same system can have different types of access. This is done with the LOCK command (see page 58).

The LOCK and UNLOCK commands enable you to control update access to a group or list so that only operators with the same operator identifier can make changes.

The lock is held on the CSD file and remains in effect across restarts of CICS. The lock is owned by the user, who is identified by a combination of the CICS generic applid (specified by the APPLID system initialization parameter), and the user's operator identifier (OPIDENT).

The OPIDENT is the one associated with the user when he or she signs on to the terminal used for RDO. For further information on OPIDENT, see the *CICS RACF Security Guide*.

Any user who is not signed on or who has a different OPIDENT is not allowed to perform any operation that would change the locked group. However, any user is allowed to do the following things to a locked group:
- COPY
- CHECK
- DISPLAY
- INSTALL
- VIEW

The lock can be removed, using the UNLOCK command, only by a user on the same system and with the same operator identifier.

It would be wise to put a lock on your group of TYPETERMs and on your group of AUTINSTMODEL TERMINALs.

Controlling access to the RDO transactions

Recommended access for the CEDA, CEDB, and CEDC transactions is as follows:
- CEDC can be given fairly wide access, because it allows only read-only commands.
- CEDB should be restricted, because it allows modification of the CSD file as well as read-only commands.
- CEDA should be further restricted to the few people allowed to modify both the active CICS system and the CSD file.

Dependent default values

Some attribute values have default values that differ, depending on the values for the other attributes you have already specified on the command line.

These default values are known as **dependent defaults**. They are similar to forced values, but you can change them if you want to.

Dependent and forced values mean that you do not have to remember which attributes are relevant to different devices. (See Table 17 on page 335 for a list of TYPETERM dependent defaults and forced values.)

RDO does not record that they are defaults, as it can with default names and forced values, so if you change the value of the attribute on which they depend, **the dependent defaults stay as they were**. For example, if you copy a 3270P TYPETERM and then change its DEVICE attribute to 3270, you do not get the appropriate default values for the other attributes, and your TYPETERM definition is not applicable to a display device or to a printer, unless you change all the values yourself.

Note: Do not change the value of DEVICE, SESSIONTYPE, or TERMMODEL, either using ALTER or by overtyping. If you make a mistake with these attributes, delete the definition and define a new one. For the same reason, do **not** use COPY and ALTER as a method of creating many TYPETERMs with different DEVICE values.

Part 2. Resource definition online (RDO)

This part tells you how to use the CEDA transaction to add, remove, or change resource definitions online. It contains the following chapters:

See "Chapter 1. Resource definition—an introduction" on page 3 for information on the resources that you can define using RDO.

Chapter 3. CEDA tutorial

This chapter is a tutorial showing you how to use the RDO transactions CEDA, CEDB, and CEDC. It assumes that you have read the information on resources, groups, and lists in “How the CSD file is organized—groups and lists” on page 8 and “Using lists for initialization” on page 16.

At the end of this tutorial, you should be familiar with the CEDA panels, and be able to manage your resources efficiently.

The examples in the tutorial all use CEDA because, if you have CEDA authorization, you can issue all RDO commands. If you have access to only CEDB or CEDC, you can issue the following commands:

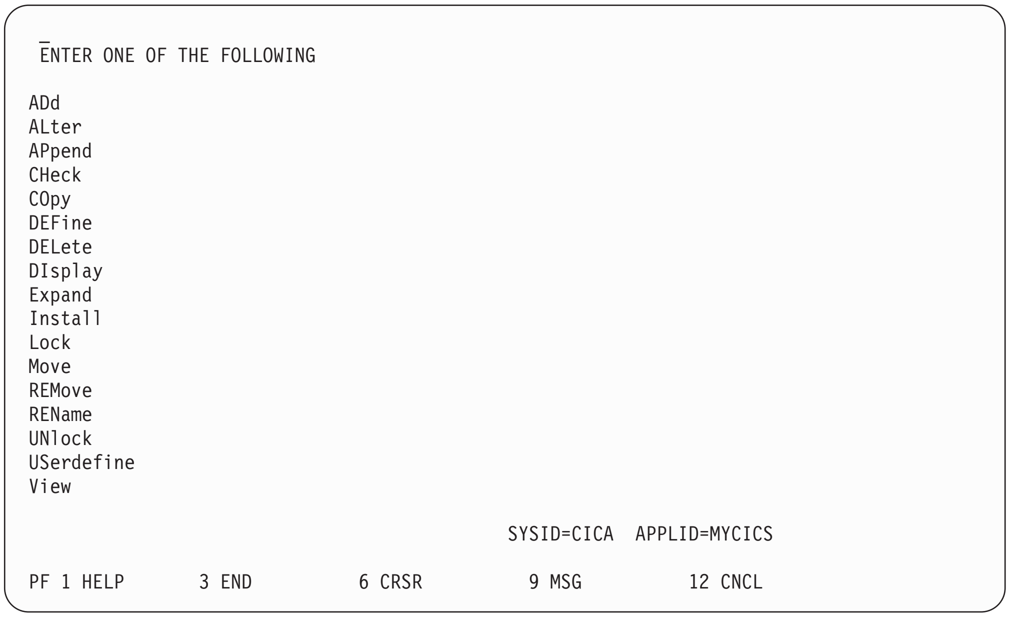
CEDB All RDO commands except INSTALL.

CEDC DISPLAY, EXPAND, and VIEW commands only.

Using the CEDA panels

Enter `CEDA` at a CICS terminal. The following panel appears:

(The cursor is indicated by the symbol ‘_.’.)



```
 _
 ENTER ONE OF THE FOLLOWING

ADd
ALter
APpend
CHeck
COpy
DEFine
DELete
DIsplay
Expand
Install
Lock
Move
REMove
REName
UNlock
USerdefine
View
                                             SYSID=CICA  APPLID=MYCICS

PF 1 HELP       3 END          6 CRSR          9 MSG          12 CNCL
```

Figure 1. CEDA transaction: initial panel

Figure 1 shows a list of all the CEDA commands. CEDB has all of these except INSTALL, and CEDC offers you only DISPLAY, EXPAND, and VIEW.

Abbreviating commands

Each command can be abbreviated. The minimum abbreviations are shown in uppercase—for example, you can enter REM for REMOVE, but not just R or RE, because to CEDA that would be ambiguous with RENAME.

Note: Minimum abbreviations may change between CICS releases because of the introduction of new commands.

So that you have some resources of your own to work with, use the DEFINE command to create a new resource in a new group. At the top of the screen, enter `DEFINE`. You see the following screen:

```
 DEFINE
 ENTER ONE OF THE FOLLOWING
Connection
DB2Conn
DB2Entry
DB2Tran
DOctemplate
Enqmodel
File
Journalmodel
Lsrpool
Mapset
PARTItionset
PARTNer
PROCesstype
PROFile
PROGram
Requestmodel
Sessions
TCpipservice
TDqueue
TErminal
TRANClass
TRANSaction
TSmodel
TYpeterm


                                              SYSID=CICA APPLID=MYCICS

PF 1 HELP     3 END          6 CRSR          9 MSG        12 CNCL
```

Figure 2. CEDA DEFINE panel

This is a list of all the resource types you can define for CICS by means of CEDA or DFHCSDUP. They are explained in detail in “Part 5. RDO resource types and their attributes” on page 151.

SYSID
This is your system identifier specified in the SYSIDNT system initialization parameter.

APPLID
This is the VTAM application identifier for the CICS system, as specified in the APPLID system initialization parameter.

PF keys
The PF keys on this screen are:

PF1 HELP
Provides help in using CEDA.

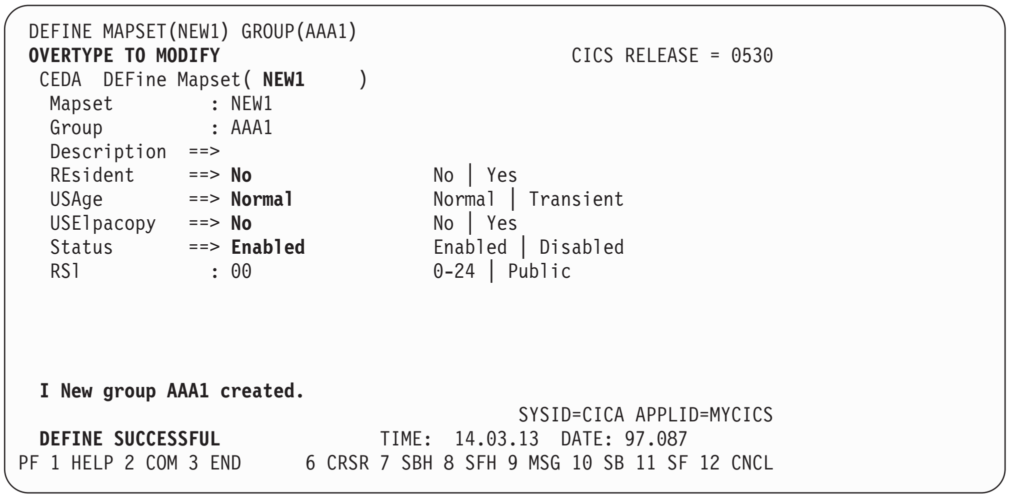
PF3 END
Takes you out of CEDA. After using PF3, you can clear the screen and continue with another transaction.

PF6 CRSR
Puts the cursor in the top left corner of the screen.

PF9 MSG
Shows you any error messages produced by CEDA. If the CMAC transaction is available, after you press PF9, placing the cursor under a message and pressing the Enter key will link to the Messages and Codes Online transaction (CMAC) for that message.

PF12 CNCL
Takes you back to the previous panel.

After the word DEFINE that you entered, type in `MAPSET(NEW1) GROUP(AAA1)`.



```
 DEFINE MAPSET(NEW1) GROUP(AAA1)
 OVERTYPE TO MODIFY                                 CICS RELEASE = 0530
  CEDA  DEFine Mapset( NEW1     )
   Mapset         : NEW1
   Group          : AAA1
   Description  ==>
   REsident     ==> No                 No | Yes
   USAge        ==> Normal             Normal | Transient
   USElpacopy   ==> No                 No | Yes
   Status       ==> Enabled            Enabled | Disabled
   RSl            : 00                 0-24 | Public




  I New group AAA1 created.
                                               SYSID=CICA APPLID=MYCICS
  DEFINE SUCCESSFUL              TIME:  14.03.13  DATE: 97.087
PF 1 HELP 2 COM 3 END      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 3. CEDA DEFINE MAPSET panel

The whole command (DEFINE MAPSET(NEW1) GROUP(AAA1)) remains at the top of the screen. You must specify a group name for every resource you want to define or work with; if you do not, CEDA displays a severe error message informing you that you have not supplied a group.

The main part of the screen shows the attributes of the MAPSET that you have just defined. All the attributes and values that you see are described in “MAPSET” on page 223.

Attributes and values

The attributes of the map set are listed in the first column, and their associated values are listed in the second column. For example:

```
Status       ==> Enabled            Enabled | Disabled
```

Status is the attribute.
Enabled is the value it has been defined with.
Enabled | Disabled shows the values it is possible to define it with.

Use of ==>

When you see a '==>' symbol between an attribute and its value, it means that you can change the value. The values that can be changed are also highlighted.

Use of colon (:)

When you see a colon (:) between an attribute and its value, it means that you cannot change the value. This can be for several reasons:

- You do not have authority to change the values (for example, if you are a CEDC user).
- You are using the VIEW command (explained later in this tutorial).
- The length of the attribute is such that the whole of the attribute does not fit in the current screen view. In order to update the attribute value you must first scroll forwards (PF8 or PF11) or backwards (PF7 or PF10) until the whole attribute is seen.
- The attribute is the resource name or the group name (in this example, MAPSET name NEW1 and GROUP name AAA1).
- The attribute is obsolete for the current release of CICS (for example, the RSL attribute). To change the value of obsolete attributes, you must use the **compatibility mode** option, as described in the PF keys below.

PF keys

The PF keys on this screen which were not on the previous screen are:

PF2 COM
: Allows you to use **compatibility mode** to change the value of obsolete attributes. In Figure 3 on page 25, if you press PF2, you see that the symbol between RSL and its value changes from '==>' to ':', the value is highlighted, and the display at the top right of the screen changes from 'CICS RELEASE = 0530' to 'COMPATIBILITY MODE'. You can now overtype the value to change it, then press PF2 again to get out of compatibility mode. Compatibility mode is described in "Compatibility mode (CSD file sharing)" on page 12.

PF7 SBH
: Scrolls back half a screen. Because MAPSET has a small number of attributes, this does not apply to this panel.

PF8 SFH
: Scrolls forward half a screen. Because MAPSET has a small number of attributes, this does not apply to this panel.

PF9 MSG
: Shows you any error messages produced by CEDA. If you now press PF9, you see the message "NEW GROUP AAA1 CREATED". This is the same message as is displayed on the screen. If you issue a command that generates more than one message, you may have to use PF9 to see them all. If the CMAC transaction is available, after you press PF9, placing the cursor under a message and pressing the Enter key will link to the Messages and Codes Online transaction (CMAC) for that message.

PF10 SB
: Scrolls up one full screen. Because MAPSET has a small number of attributes, this does not apply to this panel.

PF11 SF
: Scrolls down one full screen. Because MAPSET has a small number of attributes, this does not apply to this panel.

Messages

CEDA has produced the informational message ‘I New group AAA1 created’. There are four levels of error messages in CEDA, as follows:

S Severe. CEDA cannot continue until you correct what is wrong.

E Error. Commands resulting in these messages will be executed when you press enter, but the results may not be what you intended; for example, if you abbreviate a command to the point where it becomes ambiguous, CEDA warns you, and tells you which command it is going to assume when you press enter. It may not be the command you intended.

W Attention.

I Information.

You can use PF9 to view CEDA messages. If you press PF9 now, the result will be as shown in Figure 3 on page 25 because there is only one message. When you have read the messages, press the Enter key to get back to where you were.

Invoking CMAC from CEDA

If the CMAC transaction is available, you can also place the cursor under any of the messages and press the Enter key to link to the Messages and Codes online information for that specific message.

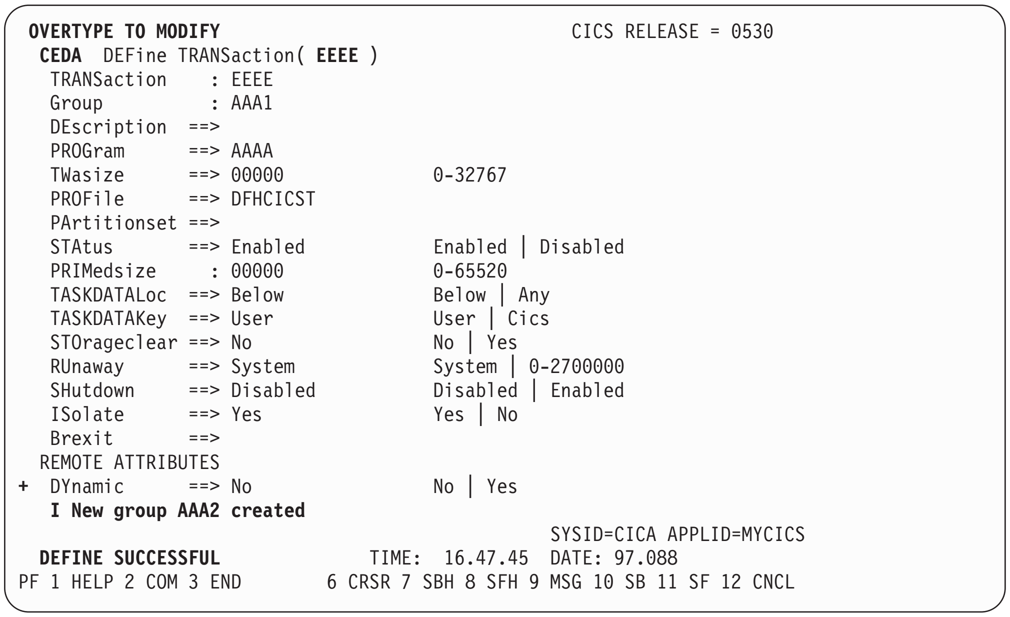
Press PF3 to get out of CEDA. The following is displayed:

```
CEDA DEFINE MAPSET(NEW7) GROUP(AAA1)
 STATUS:  SESSION ENDED
```

Figure 4. SESSION ENDED panel

Clear the screen, and you will be able to enter another transaction.

So far, you have gone through a number of panels to define a resource, but you have the alternative of entering a complete CEDA command on the command line. For example, you could enter: `CEDA DEFINE TRANSACTION(XYZ1) PROGRAM(XYZ2) GROUP(AAA2)`. The screen you would see is very similar to Figure 3 on page 25:



```
 OVERTYPE TO MODIFY                                 CICS RELEASE = 0530
  CEDA  DEFine TRANSaction( EEEE )
   TRANSaction    : EEEE
   Group          : AAA1
   DEscription  ==>
   PROGram      ==> AAAA
   TWasize      ==> 00000             0-32767
   PROFile      ==> DFHCICST
   PArtitionset ==>
   STAtus       ==> Enabled           Enabled | Disabled
   PRIMedsize     : 00000             0-65520
   TASKDATALoc  ==> Below             Below | Any
   TASKDATAKey  ==> User              User | Cics
   STOrageclear ==> No                No | Yes
   RUnaway      ==> System            System | 0-2700000
   SHutdown     ==> Disabled          Disabled | Enabled
   ISolate      ==> Yes               Yes | No
   Brexit       ==>
  REMOTE ATTRIBUTES
+  DYnamic      ==> No                No | Yes
   I New group AAA2 created
                                                 SYSID=CICA APPLID=MYCICS
  DEFINE SUCCESSFUL            TIME:  16.47.45  DATE: 97.088
PF 1 HELP 2 COM 3 END       6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

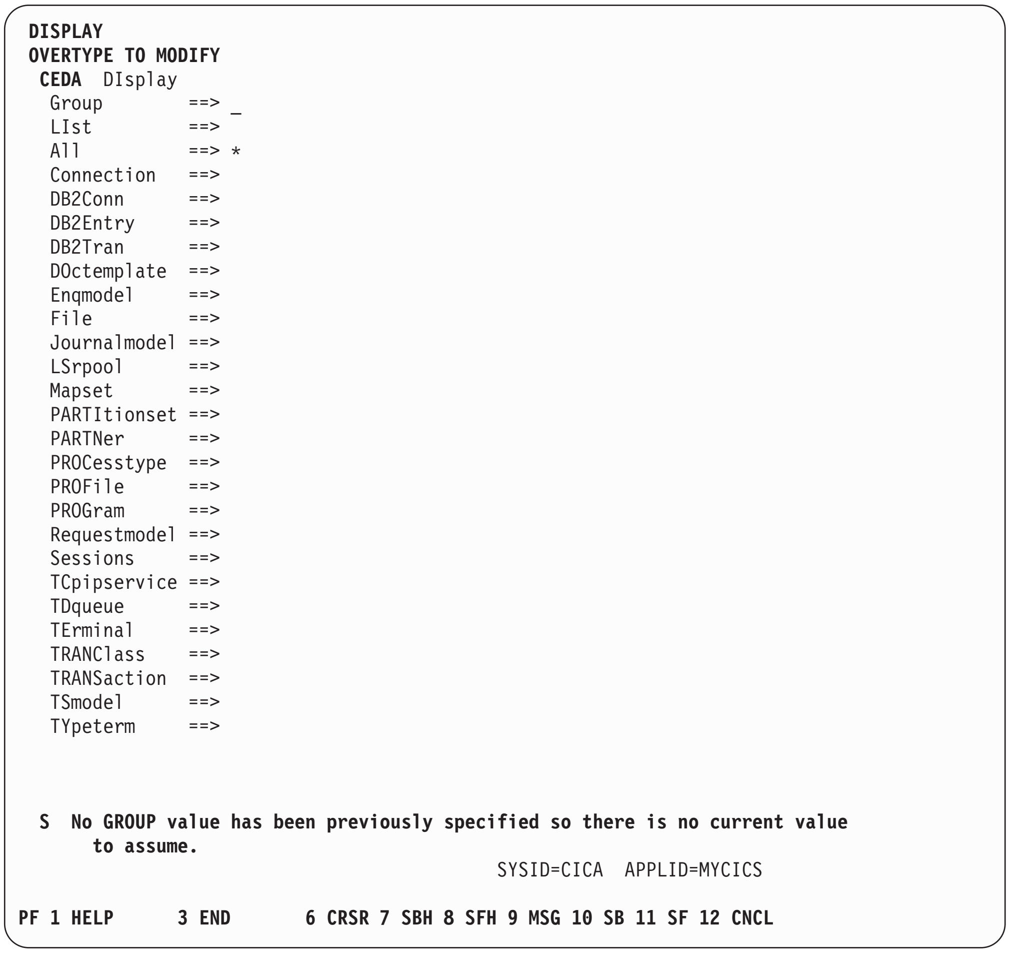
Figure 5. CEDA DEFINE TRANSACTION panel

You must specify a PROGRAM whenever you define a new TRANSACTION; if you do not, CEDA displays a severe error message saying that you must specify either a PROGRAM or REMOTESYSTEM. The attributes of the TRANSACTION definition are described in “TRANSACTION” on page 314.

You can see that the TRANSACTION definition has many more attributes than the MAPSET definition. The **plus sign (+)** to the left of the last attribute in the list means that there are more attributes, so you can use either PF8 or PF11 to scroll down through them, then PF7 or PF10 to scroll back up.

You now have two new resources to work with: a MAPSET in group AAA1, and a TRANSACTION in group AAA2. These resources are used throughout the remainder of the tutorial to demonstrate the other CEDA commands.

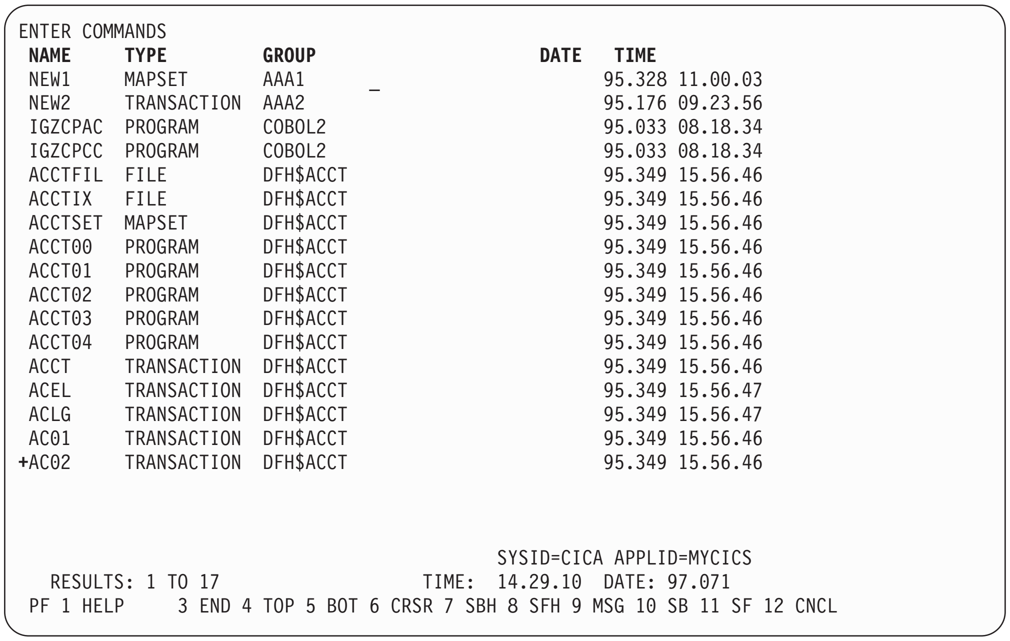
Press PF3 to get out of CEDA. Instead of clearing the screen as you did before, you have the alternative of typing a new command over the existing one at the top of the screen. Overtype `DEFINE` with `DISPLAY` and delete the rest of the line. You see a screen like this:



```
 DISPLAY
 OVERTYPE TO MODIFY
  CEDA  DIsplay
   Group       ==> _
   LIst        ==>
   All         ==> *
   Connection  ==>
   DB2Conn     ==>
   DB2Entry    ==>
   DB2Tran     ==>
   DOctemplate ==>
   Enqmodel    ==>
   File        ==>
   Journalmodel ==>
   LSrpool     ==>
   Mapset      ==>
   PARTItionset ==>
   PARTNer     ==>
   PROCesstype ==>
   PROFile     ==>
   PROGram     ==>
   Requestmodel ==>
   Sessions    ==>
   TCpipservice ==>
   TDqueue     ==>
   TErminal    ==>
   TRANClass   ==>
   TRANSaction ==>
   TSmodel     ==>
   TYpeterm    ==>



  S  No GROUP value has been previously specified so there is no current value
       to assume.
                                             SYSID=CICA  APPLID=MYCICS

PF 1 HELP     3 END      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 6. CEDA DISPLAY panel

You may not know any group names or list names yet, so type in an asterisk (*) where the cursor is, beside GROUP, then press Enter. The asterisk means that you want to display all groups. If your system has a large number of groups in it, this command may take some time to execute. You get a screen like this:



```
ENTER COMMANDS
 NAME     TYPE        GROUP                    DATE   TIME
 NEW1     MAPSET      AAA1     _                     95.328 11.00.03
 NEW2     TRANSACTION AAA2                           95.176 09.23.56
 IGZCPAC  PROGRAM     COBOL2                         95.033 08.18.34
 IGZCPCC  PROGRAM     COBOL2                         95.033 08.18.34
 ACCTFIL  FILE        DFH$ACCT                       95.349 15.56.46
 ACCTIX   FILE        DFH$ACCT                       95.349 15.56.46
 ACCTSET  MAPSET      DFH$ACCT                       95.349 15.56.46
 ACCT00   PROGRAM     DFH$ACCT                       95.349 15.56.46
 ACCT01   PROGRAM     DFH$ACCT                       95.349 15.56.46
 ACCT02   PROGRAM     DFH$ACCT                       95.349 15.56.46
 ACCT03   PROGRAM     DFH$ACCT                       95.349 15.56.46
 ACCT04   PROGRAM     DFH$ACCT                       95.349 15.56.46
 ACCT     TRANSACTION DFH$ACCT                       95.349 15.56.46
 ACEL     TRANSACTION DFH$ACCT                       95.349 15.56.47
 ACLG     TRANSACTION DFH$ACCT                       95.349 15.56.47
 AC01     TRANSACTION DFH$ACCT                       95.349 15.56.46
+AC02     TRANSACTION DFH$ACCT                       95.349 15.56.46



                                       SYSID=CICA APPLID=MYCICS
   RESULTS: 1 TO 17                 TIME:  14.29.10  DATE: 97.071
 PF 1 HELP     3 END 4 TOP 5 BOT 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 7. CEDA DISPLAY GROUP(*) panel

This shows the groups in your system and the resource definitions that they contain.

Plus sign
The plus sign (+) beside the last value means that there are more resources, so you can use PF8 or PF10 to scroll down to see them, then PF7 or PF11 to scroll back up again. PF7 and PF8 enable you to scroll down and up half a screen at a time. PF10 and PF11 enable you to scroll down and up a full screen at a time.

NAME
The list under the NAME heading tells you the name of each individual resource definitions installed on your system. You can have duplicate resource names only if the resource definitions are in different groups. The resource names are listed alphabetically according to their resource types (that is, within any one group, all the transactions are listed alphabetically, then all the programs, and so on).

TYPE
This tells you what resource type each definition is for. The resource types are introduced briefly on page 153, and are described fully in “Part 5. RDO resource types and their attributes” on page 151. Within each group, the similar resource types are shown together (that is, all the transactions, then all the programs, and so on).

GROUP
This tells you which group each resource definition belongs to. This whole display of resource definitions is listed in alphabetic sequence of group names.

DATE and TIME
This shows the date and time when the resource definition was last updated. The time and date immediately above the PF key descriptions at the bottom of the screen are the current time and date.

RESULTS: 1 TO 17

Below the list of groups, there is a line that says “RESULTS: 1 TO 17”. This tells you how many group names are displayed on the screen. If there are more than 17, this message changes as you scroll up and down the list of group names. At the beginning of a CEDA DISPLAY GROUP(*) command, it says “RESULTS: 1 TO 17”, but as you scroll down, CEDA builds up a count of the total number of groups, and eventually the message changes to be of the form “RESULTS: 52 TO 68 OF 97”. If the list is very long, you can use PF5 to go straight to the bottom of it, and PF4 to get back to the top of it.

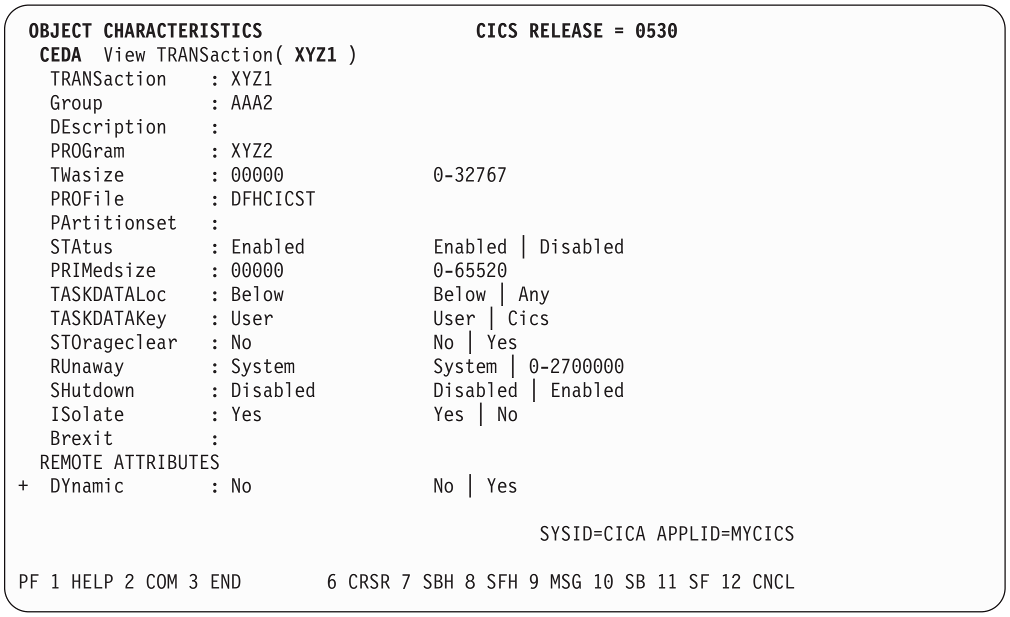
ENTER COMMANDS

You can enter commands in this area of the screen. If you are using CEDA, valid commands from this area are:

ALTER
COPY
DELETE
INSTALL
MOVE
RENAME
VIEW
?
=

If you are using CEDB, all of these are valid except INSTALL. If you are using CEDC, only VIEW, ?, and = are valid.

Move the cursor down to the TRANSACTION XYZ1 that you defined earlier and type in `VIEW` beside it. A screen like this is displayed:



```
 OBJECT CHARACTERISTICS                    CICS RELEASE = 0530
  CEDA  View TRANSaction( XYZ1 )
   TRANSaction    : XYZ1
   Group          : AAA2
   DEscription    :
   PROGram        : XYZ2
   TWasize        : 00000             0-32767
   PROFile        : DFHCICST
   PArtitionset   :
   STAtus         : Enabled           Enabled | Disabled
   PRIMedsize     : 00000             0-65520
   TASKDATALoc    : Below             Below | Any
   TASKDATAKey    : User              User | Cics
   STOrageclear   : No                No | Yes
   RUnaway        : System            System | 0-2700000
   SHutdown       : Disabled          Disabled | Enabled
   ISolate        : Yes               Yes | No
   Brexit         :
  REMOTE ATTRIBUTES
+  DYnamic        : No                No | Yes

                                                SYSID=CICA APPLID=MYCICS

PF 1 HELP 2 COM 3 END       6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 8. CEDA VIEW panel

This is almost identical to the panel you saw when you defined this transaction (shown in Figure 5 on page 28), with the difference that on the VIEW panel there is a colon (:) between each attribute and its value. You cannot change any values from a VIEW panel.

If you notice at this point that you have defined TRANSACTION(XYZ1) incorrectly, you may want to alter one or more of its values. Press PF12 to return to the DISPLAY screen. There is now an asterisk (*) beside the transaction to indicate that you have worked with it. Type `ALTER` over the asterisk, and you will see a panel that begins like this:

```
OVERTYPE TO MODIFY                          CICS RELEASE = 0530
 CEDA  ALter TRANSaction( XYZ1 )
  TRANSaction    : XYZ1
  Group          : AAA2
  DEscription  ==>
  PROGram      ==> XYZ2
  TWasize      ==> 00000             0-32767
  PROFile      ==> DFHCICST
   ⋮
```

Figure 9. CEDA ALTER panel

The highlighted text at the top left of the screen, ‘OVERTYPE TO MODIFY’, means that you can overtype any of the highlighted values. Overtype the TWASIZE value, changing it from 00000 to 32767. When you press Enter, CEDA displays an ‘ALTER SUCCESSFUL’ message at the bottom of the screen.

Press PF12 to return to the DISPLAY screen, where the ‘ALTER SUCCESSFUL’ message is displayed on the same line as the TRANSACTION. Type `VIEW` against the TRANSACTION, and you can see that the value of TWASIZE has been changed.

So far, you have defined two resources. You can create more using the DEFINE command as before, or you can copy existing resources, both within their own groups and to other groups.

Press PF12 to return to the DISPLAY screen. Beside TRANSACTION(XYZ1) type in:

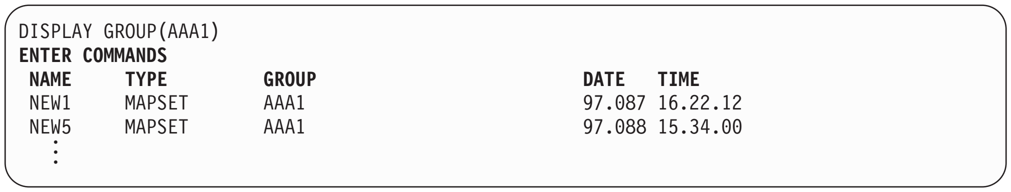
```
COPY TO(AAA1) AS(XYZ2)
```

A ‘COPY SUCCESSFUL’ message is displayed in the right hand column of the screen.

The COPY command has copied the definition for TRANSACTION(XYZ1) from group AAA2 to group AAA1, renaming it as TRANSACTION(XYZ2). The original transaction, TRANSACTION(XYZ1), still exists in group AAA2. Press PF3 to display the SESSION ENDED panel, and overtype the existing ‘CEDA DISPLAY’ with:

```
CEDA DISPLAY GROUP(AAA1)
```

Group AAA1 now contains two resource definitions: a MAPSET called NEW1 and a TRANSACTION called XYZ2. Because this is a DISPLAY screen similar in format to Figure 7 on page 30, you can enter the same commands against the definitions.



```
DISPLAY GROUP(AAA1)
ENTER COMMANDS
 NAME     TYPE        GROUP                            DATE   TIME
 NEW1     MAPSET      AAA1                             97.087 16.22.12
 NEW5     MAPSET      AAA1                             97.088 15.34.00
   :
```

Figure 10. CEDA DISPLAY GROUP(AAA1) panel

The next part of the tutorial is about entering commands from the command line. Press PF3 to get out of CEDA and clear the screen.

Entering commands from the command line

So far, this tutorial has taken you through panels to execute CEDA commands, but when you become familiar with your system's resources and with CEDA, you can enter most CEDA commands on the command line.

Here is a sequence of commands; if you follow them, you can see that entering commands on the command line is quicker than going through all the CEDA panels.

You do not need to use PF3 or PF12 at this point.

1. Use PF6 to move the cursor to the top left corner of the screen. Insert a transaction name and remove the group; the command now looks like this:
   ```
   CEDA DEFINE TRANSACTION(BBB1) PROGRAM(CCC1)
   ```
2. When you press Enter, the new transaction BBB1 is defined. Note that it has been defined without an associated group name; this is because, as long as you are within the same CEDA session, the current group (that is, the one you have most recently been working with) is used by default. If you had used PF3 before this, your CEDA session would have ended, so there would be no current group name for CEDA to assume.
3. Rename the transaction by overtyping the command to read:
   ```
   RENAME TRANSACTION(BBB1) AS(DDD1)
   ```
4. Move the transaction to group AAA2 by overtyping the command to read:
   ```
   MOVE TRANSACTION(DDD1) TO(AAA2)
   ```
5. Before installing groups AAA1 and AAA2, you can check them by using the CHECK command. This ensures that all transactions have access to their related programs, that terminal definitions have access to their related typeterm definitions, and so on. Overtype the command to read:
   ```
   CHECK GROUP(AAA1)
   ```

   This should result in the message:
   ```
   W PROGRAM CCC1 referenced by TRANSACTION DDD1 in group AAA2
     cannot be found
   ```

   This is because when you moved transaction DDD1, you did not also move its
   # related program, CCC1.

   # **Note:** You will not see this message if you have autoinstall progams active. If
   # autoinstall programs is active, CEDA assumes that the program will be
   # insntalled at execution time and does not check the program definitions.
6. # You can remedy this by using the COPY command:

```
COPY PROGRAM(CCC1) G(AAA1) TO G(AAA2)
```

If you repeat the CHECK command on both groups, there should be no more error messages.

This is clearly faster and more efficient than using panels. You can now use PF3 to get out of CEDA, then clear the screen.

Removing resource definitions from the CSD file

The resource definitions that you have created for the tutorial can be removed; this allows other people to follow the tutorial, and ensures that your CSD file space is not being used unnecessarily. Using the DELETE command, you can delete the whole of groups AAA1 and AAA2 from your CSD file:

```
CEDA DELETE ALL GROUP(AAA1)
CEDA DELETE ALL GROUP(AAA2)
```

Note that you cannot delete the group itself; it ceases to exist only when it contains no resource definitions. See “DELETE” on page 95 for more information on the DELETE command.

Generic naming in CEDA

Generic names allow you to use one command to perform the same operation on many objects. You used this feature of CEDA on page 29 when you used an asterisk to mean ‘all’ groups.

The asterisk can be used to mean ‘all’ resource names, groups, or lists. For example, CEDA VIEW TRANSACTION(DISC) GROUP(*) shows you transactions called DISC in any group where they occur. CEDA DISPLAY GROUP(*) TRANSACTION(*) shows you all the transactions in every group.

Another way to see groups without having to specify their exact names is to use the plus sign (+) as part of the group name to represent one character. For example, whereas CEDA DISPLAY GROUP(DFH*) displays all groups beginning with DFH, CEDA DISPLAY GROUP(DFH+++) displays only those groups that begin with DFH and are six characters long, CEDA DISPLAY GROUP(DFH++++) displays those that begin with DFH and are seven characters long, and so on.

You can also use the ALL attribute in some commands, instead of specifying a resource type. The command then operates on all resource definitions that fit the type specified with ALL.

Which commands accept generic names?

Table 4 on page 35 tells you which CEDA commands you can use generic naming with.

yes	means that you may use a generic name or an actual name.
no	means that you must use an actual name.
—	means that generic naming is not applicable for this command.

Table 4. CEDA commands accepting generic names

Command	Generic resource name	Generic group name	Generic list name	ALL resource types
ADD	—	no	no	—
ALTER	yes	yes	—	no
APPEND	—	—	no	—
CHECK GROUP	—	no	—	—
CHECK LIST	—	—	no	—
COPY	yes	no	—	yes
DEFINE	no	no	—	no
DELETE	yes	no	—	yes
DISPLAY GROUP	—	yes	—	—
DISPLAY GROUP ALL	yes	yes	—	yes
DISPLAY LIST	—	—	yes	—
DISPLAY LIST GROUP	—	yes	yes	—
EXPAND GROUP	yes	yes	—	yes
EXPAND LIST	—	yes	yes	—
INSTALL	—	no	no	—
INSTALL GROUP	—	no	—	—
LOCK GROUP	—	no	—	—
LOCK LIST	—	—	no	—
MOVE	yes	no	—	yes
REMOVE	—	yes	no	—
RENAME	no	no	—	yes
UNLOCK GROUP	—	no	—	—
UNLOCK LIST	—	—	no	—
USERDEFINE	no	no	—	no
VIEW	yes	yes	—	no

Chapter 4. CEDA commands—syntax and examples

This chapter explains the syntax for each of the CEDA commands and shows examples of use. The commands are:

ADD

Add a group to a list.

ADD syntax

```
►►──ADd──Group(groupname1)──LIst(listname)──┬─────────────────────┬──────────────────────►◄
                                            ├─Before(groupname2)──┤
                                            └─After(groupname3)───┘
```

Description

You can use the ADD command from a DISPLAY screen.

Options

After(*groupname3***)**
You can use this to control the placing of the new group name. If you do not specify BEFORE or AFTER, the group name is added at the end of the list.

Before(*groupname2***)**
You can use this to control the placing of the new group name. If you do not specify BEFORE or AFTER, the group name is added at the end of the list.

Group(*groupname1***)**
specifies the name of the group to be added. The name must not already exist in the list. A generic group name is not accepted. If you do not specify a group, the current group name is added.

LIst(*listname***)**
specifies the name of the list to which the group is to be added. If the list does not already exist, a new one is created. If LIST is not specified, the group name is added to the current list if there is one. A generic list name is not accepted.

Examples

To create a list LA01 by adding a group to it:

```
ADD GROUP(GA001) LIST(LA01)
```

To add another group to list LA01, without specifying where:

```
ADD GROUP(GA002) LIST(LA01)
```

LA01 now looks like this:
GA001
GA002

To add another group at the top of the list:

```
ADD GROUP(GA003) LIST(LA01) BEFORE(GA001)
```

and another group between GA001 and GA002:

```
ADD GROUP(GA004) LIST(LA01) AFTER(GA001)
```

LA01 now looks like this:
GA003
GA001
GA004
GA002

ALTER

Change some or all of the attributes of an **existing** resource definition.

ALTER syntax

```
►►──ALter───┬─Connection(name)───┬──Group(groupname)──attribute list(new value)──────────────►◄
            ├─DB2Conn(name)──────┤
            ├─DB2Entry(name)─────┤
            ├─DB2Tran(name)──────┤
            ├─DOctemplate(name)──┤
            ├─Enqmodel(name)─────┤
            ├─File(name)─────────┤
            ├─Journalmodel(name)─┤
            ├─Lsrpool(name)──────┤
            ├─Mapset(name)───────┤
            ├─PARTItionset(name)─┤
            ├─PARTNer(name)──────┤
            ├─PROCesstype(name)──┤
            ├─PROFile(name)──────┤
            ├─PROGram(name)──────┤
            ├─Requestmodel(name)─┤
            ├─Sessions(name)─────┤
            ├─TCpipservice(name)─┤
            ├─TErminal(name)─────┤
            ├─TDqueue(name)──────┤
            ├─TRANClass(name)────┤
            ├─TRANSaction(name)──┤
            ├─TSmodel(name)──────┤
            └─TYpeterm(name)─────┘
```

Description

For information about the attributes that you can specify on the ALTER command for the various resource types, see "Part 5. RDO resource types and their attributes" on page 151.

Do **not** use ALTER to change the value of the attributes of a TYPETERM definition on which other attributes depend. If you make a mistake with DEVICE, SESSIONTYPE, or TERMMODEL, delete the definition and define a new one with the correct values.

You can specify null attribute values, for example:

```
ALTER FILE(TEST) GROUP(ACT1) DESCRIPTION()
```

If an attribute for which you have specified a null value has a default, the value used depends upon the type of field. For example:

- The command:
  ```
  ALTER FILE(TEST) GROUP(ACT1) RLSACCESS()
  ```
  behaves as if RLSACCESS was not specified. The RLSACCESS attribute has a CVDA default value which is ignored.
- The command:
  ```
  ALTER FILE(TEST) GROUP(ACT1) DESCRIPTION()
  ```

has the effect of blanking out the description, as there is no default value for the
DESCRIPTION field, and it is option.

• The command:

#
```
ALTER FILE(TEST) GROUP(ACT1) PROFILE()
```

puts the default value of DFHCICSA into the PROFILE field. In this case, the
default value is a character string, not a CVDA value.

Changes to resource definitions in the CSD file do not affect the running CICS system until you install the definition, or the group in which the resource definition resides.

You can use CEDA ALTER from a DISPLAY panel. If you use PF12 after making your alterations, CEDA gives you the DISPLAY panel again, with an 'ALTER SUCCESSFUL' message in the Date and Time fields. If you do this but do not make any alterations, an asterisk replaces your 'ALTER' command.

With a generic name, you can use one ALTER command to change the same attributes in the same way on more than one resource definition.

Options

Attribute list
specifies the attributes to be altered. For information about the attributes that you can specify on the ALTER command for each resource type, see "Part 5. RDO resource types and their attributes" on page 151.

Group(*groupname*)
specifies the name of the group containing the resource to be altered.

Resource(*name*)
specifies the resource whose attributes you want to alter.

Examples

To make a program resident:

```
ALTER PROGRAM(ERR01) GROUP(GENMODS) RESIDENT(YES)
      DATALOCATION()
```

To change the status of a whole group of programs:

```
ALTER PROGRAM(*) GROUP(GENMODS) STATUS(ENABLED)
```

If you do not specify an attribute list and type in:

```
ALTER PROGRAM(ERR01) GROUP(GENMODS)
```

CEDA gives an "ALTER SUCCESSFUL" message followed by the 'overtype to modify' panel.

APPEND

Add the groups in one list to the end of another list.

APPEND syntax

```
►►──APpend──LIst(listname1)──To(listname2)──────────────────────────────────►◄
```

Options

LIst(*listname1***)**
specifies the source list to be appended. A generic list name is not accepted.

To(*listname2***)**
specifies the target list to be appended to. A generic list name is not accepted. If **listname2** already exists, the source list is appended to it. If **listname2** does not exist, it is created.

Examples

A list called LISTA contains the following groups:
GB001
GB002
GB003

A list called LISTB contains the following groups:
G001
G002
G003

Append LISTB to LISTA, like this:

```
APPEND LIST(LISTB) TO(LISTA)
```

After this, LISTA contains the following groups, in this order:
GB001
GB002
GB003
G001
G002
G003

and LISTB still contains:
G001
G002
G003

CHECK

Check the consistency of definitions.

CHECK syntax

```
►►──CHeck──┬─Group(groupname)──────────────────────────────┬──┬──────────────────────┬───────────────►◄
           └─List(listname1, listname2, listname3, listname4)─┘  └─Remotesystem(sysid)─┘
```

Description

The CHECK command performs a cross-check of a group, list, or lists of resource definitions, and should be used before the resource definitions are installed.

It checks that the resource definitions within the group or lists are consistent with one another.

For example: for each TRANSACTION in the list being checked, a check is made that the named PROGRAM definition exists in one of the groups. The success of the check does not necessarily mean that the PROGRAM is available to the running system.

Lists should be checked before being used to initialize CICS during an initial or cold start (when the list or lists are specified in the GRPLIST system initialization parameter).

A group should be checked before you use the INSTALL command to install it on the running CICS system.

A group can be checked before you use the ADD command to add the group to a list. (The group might not be self-contained, in which case there is no point in checking it alone. Put it into a list with the groups containing related resource definitions.)

You can use the CHECK command from a DISPLAY panel.

Checking groups and lists of resource definitions for consistency

The CHECK command checks the consistency of definitions within a group or within all of the groups within a list. It does not, however, cross-check every attribute of a resource. You may still get error messages when installing a group although you found no problems when using the CHECK command.

If you use the CHECK GROUP command, CEDA cross-checks all of the resources in a specified group to ensure that the group is ready to be used. For example, CHECK might warn you that a transaction definition within the group does not name a program within the same group. (Note, however, that this might not be an error. The group might intentionally be paired with a group that does contain the program, or you may want the program to be autoinstalled, in which case it would not have a definition.)

If you use the CHECK LIST command, CEDA cross-checks every group named in the list or lists. It does not simply check each group in turn, but merges the definitions in all of the listed groups, and checks them all. In this way it warns you if there are duplicate resource definitions, or references to definitions that do not exist.

Checking terminal definitions

Although you can use the CHECK command to check a group of TERMINAL definitions (it resolves references from display devices to printers, for instance), it is not very useful in resolving TYPETERM references if these are in a separate group because it would produce many unwanted messages for missing TYPETERMs.

When you add a new cluster of terminal devices, create a new group for them, and then create a list containing your TYPETERM definitions group and the new group of TERMINAL definitions. You can then use the CHECK command to check the whole list, without it being too time-consuming. This list is needed only for the duration of the checking, and is never named as the GRPLIST.

To avoid duplicating TERMINAL names, you could maintain a list of all groups containing TERMINAL definitions. You can use CHECK LIST to ensure that all new TERMINAL names are unique. If this is too lengthy a process, you can avoid it if TERMINAL names beginning with similar characters are kept in separate groups, for example:

```
TERMINAL(AZ01) GROUP(AZTERMS)
TERMINAL(AZ02) GROUP(AZTERMS)
TERMINAL(AZ03) GROUP(AZTERMS)
          .
          .
          .
TERMINAL(AZnn) GROUP(AZTERMS)

TERMINAL(BJ01) GROUP(BJTERMS)
TERMINAL(BJ02) GROUP(BJTERMS)
TERMINAL(BJ03) GROUP(BJTERMS)
          .
          .
          .
TERMINAL(BJnn) GROUP(BJTERMS)
```

If you are using autoinstall for terminals, you must install the TYPETERM definitions **before** installing the autoinstall model definitions, to ensure that the model is created. The CHECK command does not check the order of such definitions.

Checking definitions of DB2 resources

For a DB2CONN object, the following checks are made:

- To ensure that there is only one DB2CONN defined in the group or list. If more than one is found (even one with a different name), a warning message is issued. Only one DB2CONN can be installed at a time.
- That PLANEXITNAME exists as a program definition in the group or list if a PLANEXITNAME is specified, and program autoinstall is not active.

For a DB2ENTRY object, the following checks are made:

- That two DB2ENTRYs of the same name do not appear in the same list. If they do, a warning message is issued.
- That a DB2CONN exists in the group or list. If one does not, a warning message is issued. This may not be an error because the DB2CONN may exist in another group elsewhere, but a DB2CONN must be installed before a DB2ENTRY can be installed.
- Whether a transaction resource definition exists for TRANSID in the group or list if TRANSID has been specified on the DB2ENTRY. If it does not, a warning message is issued.

- Whether PLANEXITNAME exists as a program definition in the group or list if PLANEXITNAME has been specified, and program autoinstall is not active.

For a DB2TRAN object, the following checks are made:

- That a DB2TRAN of the same name does not appear in the same list. If one does, a warning message is issued.
- Whether a DB2CONN exists in the group or list. If one does not, a warning message is issued. This may not be an error because the DB2CONN may exist in another group, but a DB2CONN must be installed before a DB2TRAN can be installed.
- Whether the DB2ENTRY specified on the DB2TRAN exists in the group or list. If not, a warning message is issued.
- That the TRANSID specified on the DB2TRAN exists in the group or list. If not, a warning message is issued.

Options

Group(*groupname***)**
specifies the group you want to check. A generic group name is not accepted.

List(*listname1, listname2, etc.***)**
specifies the list or lists you want to check. A generic list name is not accepted.

Remotesystem(*sysid***)**
specifies that a check is to be run on a group or list in a CICS region with a different sysid from the region that the CHECK command is being issued from. If this option is not used, the CHECK command will use the sysid of the region from which the CHECK command is issued.

COPY

Copy a resource definition, either within the same group or to a different group.

COPY syntax

```
         ┌─All──────────────────────┐
►►──COpy─┼──────────────────────────┼─Group(groupname)──┬─AS(newname)──────────────────┬──┬─Replace─┬──────►◄
         ├─Connection(name)─────────┤                   ├─TO(newgroupname)─────────────┤  └─MErge───┘
         ├─DB2Conn(name)────────────┤                   └─AS(new-name) TO(newgroupname)┘
         ├─DB2Entry(name)───────────┤
         ├─DB2Tran(name)────────────┤
         ├─DOctemplate(name)────────┤
         ├─Enqmodel(name)───────────┤
         ├─File(name)───────────────┤
         ├─Journalmodel(name)───────┤
         ├─Lsrpool(name)────────────┤
         ├─Mapset(name)─────────────┤
         ├─PARTItionset(name)───────┤
         ├─PARTNer(name)────────────┤
         ├─PROCesstype(name)────────┤
         ├─PROFile(name)────────────┤
         ├─PROGram(name)────────────┤
         ├─Requestmodel(name)───────┤
         ├─Sessions(name)───────────┤
         ├─TCpipservice(name)───────┤
         ├─TDqueue(name)────────────┤
         ├─TErminal(name)───────────┤
         ├─TRANClass(name)──────────┤
         ├─TRANSaction(name)────────┤
         ├─TSmodel(name)────────────┤
         └─TYpeterm(name)───────────┘
```

Description

If you do not specify either MERGE or REPLACE, a message warns you that you are attempting to create duplicate resources, and your COPY will be unsuccessful.

Options

AS(*newname***)**
: If you copy a definition within a group, you must use AS to rename it. You can also use AS if you want to copy a definition to another group and rename it at the same time. You cannot use a generic name when using AS.

Group(*groupname***)**
: specifies the group containing the definitions to be copied.

MErge
: This applies when there are duplicate definition names in the groups named in the COPY command. If you specify MERGE, duplicate definitions in the TO group are not replaced.

Replace
: This applies when there are duplicate definition names in the groups named in the COPY command. If you specify REPLACE, the definitions being copied replace those in the group named in the TO operand.

Resource(*name*)
specifies the resource you want to copy. The default is ALL, which copies all the resource definitions in a group to another group.

TO(*newgroupname*)
You can copy definitions to a different group, using TO to specify the new group.

Examples

You can copy a single resource definition into a new group, using the TO option to specify the new group. For example:

```
COPY SESSIONS(L122) GROUP(CICSC1) TO(CICSC2)
```

You can copy a resource definition within the same group. If you do this, you must rename it using the AS option. For example:

```
COPY TERMINAL(TD12) AS(TD34) GROUP(TERMVDU1)
```

When copying between groups, you can give the copy a new name, using the AS option to specify the new name.

```
COPY PROGRAM(ABC01) GROUP(XYZ) AS(ABC02) TO(NEWXYZ)
```

(If you leave the copy with the same name as the original definition, be careful that you install the one you want when the time comes.)

Using the ALL option, without a name, you can copy all the resource definitions in the group to the new group. For example:

```
COPY ALL GROUP(N21TEST) TO(N21PROD)
```

You can copy more than one resource definition to a new group, using the TO option to specify the new group.

Using a generic resource definition name, you can copy all or some definitions of the same resource type to the new group. For example:

```
COPY CONNECTION(*) GROUP(CICSG1) TO(CICSG2)
COPY PROGRAM(N21++) GROUP(NTEST) TO(NPROD)
```

Using the ALL option with a generic name, you can copy all or some of the resource definitions in the group to the new group. For example:

```
COPY ALL(N21*) GROUP(N21OLD) TO(N21NEW)
```

Using the ALL option with a specific name, you can copy all the resource definitions of that name (which must necessarily be for different resource types) in the group to the new group. For example:

```
COPY ALL(XMPL) GROUP(EXAMPLE) TO(EX2)
```

If you are copying a number of definitions into another group, and the groups contain duplicate resource names, you must specify either MERGE or REPLACE. For example:

```
COPY ALL GROUP(TAX1) TO(TAX2) MERGE
COPY ALL GROUP(TAX1NEW) TO(TAX1) REPLACE
```

The following example copies a group named GA001 to a group named GA002, which already exists, replacing any duplicate resource definitions by those in group GA001.

```
COPY GROUP(GA001) TO(GA002) REPLACE
```

The following example copies group GA003 to group GA004, but if any duplicate definitions occur, preserves the group GA004 definitions.

```
COPY GROUP(GA003) TO(GA004) MERGE
```

DEFINE

Create new resource definitions.

DEFINE syntax

```
►►──DEFine──┬─Connection(name)───┬─Group(groupname)──attribute list(newvalue)──────────────────────────►◄
            ├─DB2Conn(name)──────┤
            ├─DB2Entry(name)─────┤
            ├─DB2Tran(name)──────┤
            ├─DOctemplate(name)──┤
            ├─Enqmodel(name)─────┤
            ├─File(name)─────────┤
            ├─Journalmodel(name)─┤
            ├─LSRpool(name)──────┤
            ├─Mapset(name)───────┤
            ├─PARTItionset(name)─┤
            ├─PARTNer(name)──────┤
            ├─PROCesstype(name)──┤
            ├─PROFile(name)──────┤
            ├─PROGram(name)──────┤
            ├─Requestmodel(name)─┤
            ├─Sessions(name)─────┤
            ├─TCpipservice(name)─┤
            ├─TDqueue(name)──────┤
            ├─TErminal(name)─────┤
            ├─TRANClass(name)────┤
            ├─TRANSaction(name)──┤
            ├─TSmodel(name)──────┤
            └─TYpeterm(name)─────┘
```

Description

You can use the same name for more than one resource definition in a group, if the definitions are for different resource types. For example:

```
DEFINE PROGRAM(N28A) GROUP(N28APPL)
DEFINE TRANSACTION(N28A) GROUP(N28APPL)

DEFINE TERMINAL(USER) GROUP(USERDEF)
DEFINE PROGRAM(USER) GROUP(USERDEF)
```

Be careful, when you have any resource definitions with the same name, that you install the one you really want. See “Duplicate resource definition names” on page 15.

You must specify the resource type and name on the command line. You may also specify the group and other attributes on the command line.

The whole definition is displayed on an overtype-to-modify panel, and you can change any of the attributes there, unless you entered a complete DEFINE command on the command line, in which case you can not change the **name** or **groupname** attributes.

The definition was created when you entered the DEFINE... command. If you do not wish to further modify it, you may enter another command from the command line.

If a resource definition of the same name and type already exists in the group, any attributes specified on the command line are ignored, and the **existing** resource definition is displayed. You can then overtype and modify any of the existing values if you wish. If you do not wish to modify the definition, you may enter another command from the command line.

Beware of overtyping values on which other attribute values are dependent. See "Creating groups and lists" on page 8.

Options

Attribute list
The attribute list depends on the resource type being defined; some resources have attributes that must be included in the definition. See "Part 5. RDO resource types and their attributes" on page 151 for a description of the attributes and default values of each resource type. Attributes that you do not specify are given default values.

Group(*groupname*)
specifies the name of the group containing the resource definition being defined. Do not use a generic group name. If you specify the name of a group which does not already exist, the group is created.

Resource(*name*)
specifies the name of the resource you want to define. Do not use a generic resource name.

DELETE

Delete one or more resource definitions from the CSD file.

DELETE syntax

```
►►──DELete──┬─All───────────────────┬──Group(groupname)──┬────────┬──────────────────────────────────►◄
            ├─Connection(name)──────┤                    └─REMove─┘
            ├─DB2Conn(name)─────────┤
            ├─DB2Entry(name)────────┤
            ├─DB2Tran(name)─────────┤
            ├─DOctemplate(name)─────┤
            ├─Enqmodel(name)────────┤
            ├─File(name)────────────┤
            ├─Journalmodel(name)────┤
            ├─Lsrpool(name)─────────┤
            ├─Mapset(name)──────────┤
            ├─PARTItionset(name)────┤
            ├─PARTNer(name)─────────┤
            ├─PROCesstype(name)─────┤
            ├─PROFile(name)─────────┤
            ├─PROGram(name)─────────┤
            ├─Requestmodel(name)────┤
            ├─Sessions(name)────────┤
            ├─TCpipservice(name)────┤
            ├─TDqueue(name)─────────┤
            ├─TErminal(name)────────┤
            ├─TRANClass(name)───────┤
            ├─TRANSaction(name)─────┤
            ├─TSmodel(name)─────────┤
            └─TYpeterm(name)────────┘
```

Description

Deleting a resource definition is different from removing a group from a list (see "REMOVE" on page 62). A deleted resource definition really does disappear from the CSD file.

When you DELETE the last resource in a group, the group is automatically deleted. An empty group cannot exist.

This command does **not** affect definitions installed on the active CICS system. To remove installed definitions from the active CICS system, you can use either the CEMT DISCARD transaction or the EXEC CICS DISCARD command in an application program. You can discard AUTINSTMODEL, FILE, PARTNER, PROFILE, PROGRAM, TRANCLASS, and TRANSACTION resources. For information on the CEMT transaction, see the *CICS Supplied Transactions* manual. For programming information on the EXEC CICS DISCARD command, see the *CICS System Programming Reference* manual.

Options

Group(*groupname***)**
specifies the group containing the resource. Do not use a generic group name.

REMove
specifies that, when a group is deleted, the group is to be removed from all lists that had contained it.

Resource(*name*)
specifies the resource you want to delete. You can use a generic resource name. To delete all the resource definitions in the group, specify ALL(*).

Examples

To delete all resources in a group:

```
DELETE ALL(*) GROUP(TOPS)
```

To delete all programs in a group:

```
DELETE PROGRAM(*) GROUP(NSOS)
```

To delete all resources with names beginning D0 in a group:

```
DELETE ALL(D0*) GROUP(REINDEER)
```

DISPLAY

Display one or more group names, list names, or resources on a full screen panel.

DISPLAY syntax

```
►►──DISplay──┬─List(listname)──┬─────────────────┬──────────────────────────┬──────────────►◄
             │                 └─Group(groupname)─┘                          │
             └─Group(groupname)─┬─────────────────────────┬──┬────────┬──────┘
                                ├─ALl(name)───────────────┤  └─REname─┘
                                ├─Connection(name)────────┤
                                ├─DB2Conn(name)───────────┤
                                ├─DB2Entry(name)──────────┤
                                ├─DB2Tran(name)───────────┤
                                ├─DOctemplate(name)───────┤
                                ├─Enqmodel(name)──────────┤
                                ├─File(name)──────────────┤
                                ├─Journalmodel(name)──────┤
                                ├─Lsrpool(name)───────────┤
                                ├─Mapset(name)────────────┤
                                ├─PARTItionset(name)──────┤
                                ├─PARTNer(name)───────────┤
                                ├─PROCesstype(name)───────┤
                                ├─PROFile(name)───────────┤
                                ├─PROGram(name)───────────┤
                                ├─Requestmodel(name)──────┤
                                ├─Sessions(name)──────────┤
                                ├─TCpipservice(name)──────┤
                                ├─TDqueue(name)───────────┤
                                ├─TErminal(name)──────────┤
                                ├─TRANClass(name)─────────┤
                                ├─TRANSaction(name)───────┤
                                ├─TSmodel(name)───────────┤
                                └─TYpeterm(name)──────────┘
```

Description

DISPLAY GROUP command

You can enter the following commands next to the names on the panel:

CHECK
DISPLAY ALL
EXPAND[1]
INSTALL [2]
LOCK
UNLOCK

On this panel, all these commands can be abbreviated down to their initial letter. It is unnecessary to type the group name. For the syntax of each command, see that command in this section.

To DISPLAY the group names, you must use a generic name. For more information about using the DISPLAY panel, see Figure 6 on page 29 and Figure 7 on page 30.

1. Only for compatibility with previous releases.

2. You cannot install CONNECTION and SESSIONS resources from a DISPLAY panel. They have to be installed in groups.

DISPLAY LIST command

- Displays one or more list names on a full screen panel.
- You can enter the following commands next to the names on the panel:

  CHECK
  DISPLAY GROUP
  EXPAND[1]
  LOCK
  UNLOCK

  On this panel, all these commands can be abbreviated down to their initial letter. It is not necessary to type the list name. For the syntax of each command, see that command in this section.
- To DISPLAY the list names, you must use a generic list name.
- For more information about using the DISPLAY panel, see Figure 6 on page 29 and Figure 7 on page 30.

Options

Group(*groupname*)
specifies the name of the group to be displayed. You can use a generic group name.

List(*listname*)
specifies the name of the list to be displayed. You can use a generic list name.

REname
This option applies only to GROUP. Specifying RENAME allows you to rename resource definitions within the group.

Resource(*name*)
specifies the name of the resource to be displayed.

Examples

DISPLAY GROUP command

To display all groups on the CSD file:

```
DISPLAY GROUP(*)
```

To display all groups whose names begin with PROD and have seven characters:

```
DISPLAY GROUP(PROD+++)
```

DISPLAY LIST command

To display all lists on the CSD file:

```
DISPLAY LIST(*)
```

To display all lists whose names begin with PROD and have five characters:

```
DISPLAY LIST(PROD+)
```

EXPAND

Show the resource definitions in one or more groups or lists. This command has been retained for compatibility with previous releases.

EXPAND syntax

```
►►──EXPand──┬─List(listname)──┬──────────────────┬─────────────────────────────►◄
            │                 └─Group(groupname)─┘          │
            └─Group(groupname)──┬─────────────────────┬─┬────────┬─┘
                                ├─ALl(name)───────────┤ └─REname─┘
                                ├─Connection(name)────┤
                                ├─DB2Conn(name)───────┤
                                ├─DB2Entry(name)──────┤
                                ├─DB2Tran(name)───────┤
                                ├─DOctemplate(name)───┤
                                ├─Enqmodel(name)──────┤
                                ├─File(name)──────────┤
                                ├─Journalmodel(name)──┤
                                ├─Lsrpool(name)───────┤
                                ├─Mapset(name)────────┤
                                ├─PARTItionset(name)──┤
                                ├─PARTNer(name)───────┤
                                ├─PROCesstype(name)───┤
                                ├─PROFile(name)───────┤
                                ├─PROGram(name)───────┤
                                ├─Requestmodel(name)──┤
                                ├─Sessions(name)──────┤
                                ├─TCpipservice(name)──┤
                                ├─TDqueue(name)───────┤
                                ├─TErminal(name)──────┤
                                ├─TRANClass(name)─────┤
                                ├─TRANSaction(name)───┤
                                ├─TSmodel(name)───────┤
                                └─TYpeterm(name)──────┘
```

Description

EXPAND GROUP command

Shows the resource definitions in one or more groups on a full screen panel.

You can enter the following commands next to the names on the panel:

ALTER
COPY
DELETE
INSTALL
MOVE
RENAME
VIEW

All these commands can be abbreviated down to their initial letter. It is unnecessary to type the group or list name. For the syntax of each command, see that command in this section.

You may EXPAND a generic group name. For example:

Show all the resource definitions in all groups on the CSD file:

```
EXPAND GROUP(*)
```

Show all the resource definitions in groups whose names begin with PROD and are seven characters long:

```
EXPAND GROUP(PROD+++)
```

You may EXPAND a group to show only one resource type. The name you specify as the resource name may be a generic name. For example:

Show all PROFILE definitions in all groups on the CSD file:

```
EXPAND GROUP(*) PROFILE(*)
```

Show all TERMINAL definitions that begin with SZ in a group:

```
EXPAND GROUP(ZEMGROUP) TERMINAL(SZ++)
```

You may EXPAND a group or groups, limiting the resource definitions to those that share a generic name. For example:

Show all resource definitions, of all types, ending in A1:

```
EXPAND GROUP(REINDEER) ALL(*A1)
```

If you use the RENAME option, you get a special panel on which you can overtype the resource definition names. This is a quick way of renaming many resource definitions.

EXPAND LIST command

Shows the group names in one or more lists on a full screen panel.

When expanding a list, you can enter the following commands next to the names on the panel:
ADD
REMOVE

You may EXPAND part of a list, by using a generic group name, for example:

```
EXPAND LIST(INITLIST) GROUP(DFH*)
```

You may EXPAND more than one list, by using a generic list name. For example, to show the groups in all lists on the CSD file:

```
EXPAND LIST(*)
```

Show the groups in lists whose names begin with PROD and are five characters long:

```
EXPAND LIST(PROD+)
```

Options

Group(*groupname*)
specifies the group to be expanded.

List(*listname*)
specifies the list to be expanded.

REname
This option applies only to GROUP. Specifying RENAME allows you to rename resource definitions within the group.

Resource(*name*)
specifies the resource you want to see within the expanded group.

INSTALL

Dynamically make the resource definitions in the named group or group list available to the active CICS system.

INSTALL syntax

```
►►──Install──┬──────────────────────┬──┬─Group(groupname)─┬──────────────────────────────►◄
             ├─All(name)────────────┤  └─List(listname)───┘
             ├─Connection(name)─────┤
             ├─DB2Conn(name)────────┤
             ├─DB2Entry(name)───────┤
             ├─DB2Tran(name)────────┤
             ├─DOctemplate(name)────┤
             ├─Enqmodel(name)───────┤
             ├─File(name)───────────┤
             ├─Journalmodel(name)───┤
             ├─Lsrpool(name)────────┤
             ├─Mapset(name)─────────┤
             ├─PARTItionset(name)───┤
             ├─PARTNer(name)────────┤
             ├─PROCesstype(name)────┤
             ├─PROFile(name)────────┤
             ├─PROGram(name)────────┤
             ├─Requestmodel(name)───┤
             ├─TCpipservice(name)───┤
             ├─TDqueue(name)────────┤
             ├─TErminal(name)───────┤
             ├─TRANClass(name)──────┤
             ├─TRANSaction(name)────┤
             ├─TSmodel(name)────────┤
             └─TYpeterm(name)───────┘
```

Description

You can use single-resource install to install only the resources you require.

When applied to telecommunication and intercommunication resources, there are restrictions on the single-resource INSTALL command. The restrictions apply to resource definitions that are linked—for example, CONNECTION and SESSIONS definitions.

You can install the following resource types **only** as part of a group:

- CONNECTION definitions, except a CONNECTION with ACCESSMETHOD(INDIRECT)
- SESSIONS definitions
- TERMINAL definitions for pipeline terminals that share the same POOL.

If you want to use single-resource INSTALL for TERMINAL and related TYPETERM definitions, you must install the TYPETERM that is referred to by a TERMINAL definition **before** you install the TERMINAL definition.

The same applies when installing groups containing TERMINAL and TYPETERM definitions; install the group containing the TYPETERM definition before you install the group containing the TERMINAL definition. Similarly, if you install a list that contains groups containing TERMINAL and TYPETERM definitions, the group containing the TYPETERM definition must be before the group containing the TERMINAL definition. Also, in a list containing groups of DB2® resource definitions (DB2CONN, DB2ENTRY, and DB2TRAN definitions), the DB2CONN should be defined in the first group in the list.

If the named resource already exists in the active CICS system, the existing definition is replaced by the new one.

Replacement of an existing definition occurs only if it is not actually in use.

If the installation of one or more of the resources in the group(s) being installed fails because the resource is in use or for any other reason, the following takes place:

1. The install process continues with the next resource in the group.
2. When the group install is complete, if the resource that failed was part of an **installable set**, all the resources in the installable set are backed out. Resources that were committed at the individual resource level are not backed out. For a list of the resource types that are committed as part of an installable set, see "What happens when you use the INSTALL command" on page 13.
3. A message is displayed to indicate that the group has been only partially installed. A message is also produced on the message panel for each of the resources that failed to install, stating the reason for each failure. No messages are produced for those resources that have been backed out.

In addition, a message is written to the CEMT log, saying that the install completed with errors.

CEDA INSTALL can be performed from a console, using the MVS MODIFY command.

Options

Group(*groupname*)
specifies the group to be installed, or containing the resource to be installed. A generic group name is not accepted.

List(*listname*)
specifies the list of groups to be installed, or containing the resource to be installed. A generic list name is not accepted.

Resource(*name*)
specifies the resource to be installed.

Note: If you want to use singe-resource install for ENQMODEL, remember that ENQMODELs usually have the default status of *ENABLED*, so the order of installing ENQMODELs must follow the rules for ENABLING ENQMODELs; that is, ENQMODELs forming nested generic ENQnames must be enabled in order from the most to the least specific. For example, ABCD* then ABC* then AB*.

LOCK

Restrict update and delete access to a single operator identifier.

LOCK syntax

```
►►──Lock──┬─Group(groupname)─┬──────────────────────────────────────────────────►◄
          └─List(listname)───┘
```

Description

The group or list can be used, looked at, and copied by other users of RDO, but cannot be changed or deleted by them.

You can LOCK a nonexistent group or list, thereby reserving the named group or list for your own future use.

The only RDO command that releases a lock is the UNLOCK command. No other RDO commands can unlock a group or list. For example, if you DELETE all the resources in a group, or all the groups in a list, the lock remains.

You must specify either GROUP or LIST, even if you are locking the current group or list.

A generic group or list name is not accepted.

Controlling access to a group or list—LOCK and UNLOCK

The LOCK and UNLOCK commands enable you to control update access to a group or list so that only operators with the same operator identifier can make changes.

The lock is held on the CSD file and remains in effect across restarts of CICS. The lock is owned by the user, who is identified by a combination of the CICS generic applid (specified by the APPLID system initialization parameter), and the user's operator identifier (OPIDENT).

The OPIDENT is the one associated with the user when he or she signs on to the terminal used for RDO. For further information on OPIDENT, see the *CICS RACF Security Guide*. Any user who is not signed on or who has a different OPIDENT is not allowed to perform any operation that would change the locked group. However, any user is allowed to do the following things to a locked group:
- CHECK
- COPY
- DISPLAY
- INSTALL
- VIEW

The lock can be removed, using the UNLOCK command, only by a user on the same system and with the same operator identifier.

It would be wise to put a lock on your group of TYPETERMs and on your group of AUTINSTMODEL TERMINALs.

Options

Group(*groupname*)
specifies the group to be locked.

List(*listname*)
specifies the list to be locked.

Examples

To lock a list L1:

```
LOCK LIST(L1)
```

To lock a group G1:

```
LOCK GROUP(G1)
```

MOVE

Move one or more resource definitions from the group named by the GROUP option to the group named by the TO option.

MOVE syntax

```
              ┌─All──────────────────┐
►►──Move──────┼──────────────────────┼──Group(groupname)──┬────────┬──┬─AS(newname)─────────────────┬──────────►
              ├─Connection(name)─────┤                    └─REMove─┘  ├─TO(newgroupname)────────────┤
              ├─DB2Conn(name)────────┤                                └─AS(newname) TO(newgroupname)┘
              ├─DB2Entry(name)───────┤
              ├─DB2Tran(name)────────┤
              ├─DOctemplate(name)────┤
              ├─Enqmodel(name)───────┤
              ├─File(name)───────────┤
              ├─Journalmodel(name)───┤
              ├─Lsrpool(name)────────┤
              ├─Mapset(name)─────────┤
              ├─PARTItionset(name)───┤
              ├─PROCesstype(name)────┤
              ├─PROFile(name)────────┤
              ├─PROGram(name)────────┤
              ├─Requestmodel(name)───┤
              ├─Sessions(name)───────┤
              ├─TCpipservice(name)───┤
              ├─TDqueue(name)────────┤
              ├─TErminal(name)───────┤
              ├─TRANClass(name)──────┤
              ├─TRANSaction(name)────┤
              ├─TSmodel(name)────────┤
              └─TYpeterm(name)───────┘

►──┬─────────┬──────────────────────────────────────────────────────────────────────────────────────────►◄
   ├─REPlace─┤
   └─MErge───┘
```

Description

This command has the effect of copying the resource definitions from the first group into the second, followed by the deletion of the resource definitions from the first group.

When you MOVE the last resource in a group TO a different group, the group is automatically deleted. An empty group cannot exist.

If you do not specify either MERGE or REPLACE, a message warns you that you are attempting to create duplicate resource definitions. The definitions are not moved.

Options

AS(*newname*)
: If you move a definition within a group, you must use AS to rename it. You can also use AS if you want to move a definition to another group and rename it at the same time. You cannot use a generic name when using AS.

Group(*groupname*)
specifies the group containing the definitions to be moved.

MErge
This applies when there are duplicate definition names in the groups named in the MOVE command. If you specify MERGE, duplicate definitions in the TO group are not replaced.

REMove
| specifies that, when a group is deleted because the last resource in the group is
| moved elsewhere, the group is to be removed from all lists that had contained it.

REPlace
This applies when there are duplicate definition names in the groups named in the MOVE command. If you specify REPLACE, the definitions being moved replace those in the group named in the TO operand.

RESource(*name*)
specifies the resource you want to move. The default is ALL, which moves all the resource definitions in a group to another group.

TO(*newgroupname*)
You can move definitions to a different group, using TO to specify the new group.

Examples

When you move a single resource definition, you can simultaneously rename it, using the AS option to specify the new name. For example:

```
MOVE PARTITIONSET(PSETQ1) GROUP(PSET1) AS(PSETQ4)
TO(PSET2)
```

A generic resource definition name can be specified, to move all or some definitions of the same resource type. For example:

```
MOVE TRANSACTION(*) GROUP(DENTRY) TO(TEST1)
MOVE MAPSET(ACCT+++) GROUP(ACCOUNTS1) TO(ACCOUNTS2)
```

To move all the resource definitions in a group to the new group, you can use ALL. For example:

```
MOVE ALL GROUP(N21TEST) TO(N21PROD)
```

You can use ALL with a generic name, to move all qualifying resource definitions in the group to the new group. For example:

```
MOVE ALL(N21*) GROUP(N21OLD) TO(N21NEW)
```

You can use ALL with a specific name, to move all the resource definitions of that name (which must be for different resource types) in the group to the new group. For example:

```
MOVE ALL(XMPL) GROUP(EXAMPLE) TO(EXAMPLE2)
```

To merge definitions from group X in with the definitions in group Y, keeping the Y version of any duplicates:

```
MOVE GROUP(X) TO(Y) MERGE
```

To combine definitions from group X in with definitions in group Y, keeping the X version of any duplicates:

```
MOVE GROUP(X) TO(Y) REPLACE
```

REMOVE

Remove a group name from a list.

REMOVE syntax

```
►►──REMove──Group(groupname)──List(listname)──────────────────────────────────►◄
```

Description

The group, and all its resource definitions, still exists on the CSD file. When the last group is removed from a list, the list no longer exists on the CSD file.

A generic list name is not accepted.

A generic group name can be specified to remove many or all groups from a list with one command.

| When a group is deleted, the user may have requested that the group be removed from all lists that had contained it. When the last group is removed from a list, the list is deleted.

Options

Group(*groupname***)**
specifies the group to be removed.

List(*listname***)**
specifies the list from which the group is to be removed.

Examples

A list LL02 contains the following groups:
G001
X001
XG001
G002
G003
X002
G004.

To remove all groups beginning with G:

```
REMOVE GROUP(G*) LIST(LL02)
```

This leaves:
X001
XG001
X002.

To remove the list completely:

```
REMOVE GROUP(*) LIST(LL02)
```

RENAME

Rename a resource definition to the new name specified in the AS option.

RENAME syntax

```
►►──REName──┬─ALl(name)──────────┬──┬──────────────────┬──┬──────────────┬──┬─────────────────────┬──────►
            ├─Connection(name)───┤  └─Group(groupname)─┘  └─AS(newname)─┘  └─TO(newgroupname)─┘
            ├─DB2Conn(name)──────┤
            ├─DB2Entry(name)─────┤
            ├─DB2Tran(name)──────┤
            ├─DOctemplate(name)──┤
            ├─Enqmodel(name)─────┤
            ├─File(name)─────────┤
            ├─Journalmodel(name)─┤
            ├─Lsrpool(name)──────┤
            ├─Mapset(name)───────┤
            ├─PARTItionset(name)─┤
            ├─PARTNer(name)──────┤
            ├─PROCesstype(name)──┤
            ├─PROFile(name)──────┤
            ├─PROGram(name)──────┤
            ├─REQuestmodel(name)─┤
            ├─Sessions(name)─────┤
            ├─TCpipservice(name)─┤
            ├─TDqueue(name)──────┤
            ├─TErminal(name)─────┤
            ├─TRANClass(name)────┤
            ├─TRANSaction(name)──┤
            ├─TSmodel(name)──────┤
            └─TYpeterm(name)─────┘

►──┬──────────┬──────────────────────────────────────────────────────────────────────────────────►◄
   └─REMove─┘
```

Description

You can also rename a resource definition by using the DISPLAY command or the EXPAND command with the RENAME option. See "DISPLAY" on page 52 and "EXPAND" on page 54 for information about these commands.

Options

AS(*newname*)
If you copy a definition within a group, you must use AS to rename it. You can also use AS if you want to copy a definition to another group and rename it at the same time. You cannot use a generic name when using AS.

Group(*groupname*)
specifies the group containing the definitions to be copied. A generic group name is not accepted.

| **REMove**
| specifies that, when a group is deleted, the group is to be removed from all
| lists that had contained it.

Resource(*name*)
specifies the resource you want to copy. The default is ALL, which copies all the resource definitions in a group to another group. A generic resource definition name is not accepted.

TO(*newgroupname*)
You can copy definitions to a different group, using TO to specify the new group.

Examples

To rename a resource and keep it in its current group:

```
RENAME PROFILE(PROF1) AS(NEWPROF) GROUP(PROFS)
```

You can rename all resource definitions which share the same name, to a new name, using the ALL option instead of a resource type. For example:

```
RENAME ALL(TVA) AS(XTVA) GROUP(XTVA1)
RENAME ALL(USER) AS(OLDU) GROUP(USERDEF)
```

You can move a resource definition to a new group, which you specify in the TO option, at the same time as renaming it. (You can also do this with the MOVE command.) For example:

```
RENAME PROGRAM(N20ZA) AS($SOSERR) GROUP(N20) TO($MODULES)
```

You can move all the resource definitions of the same name from one group to another, using the TO option, at the same time as renaming them. For example:

```
RENAME ALL(USER) GROUP(USERDEF) AS(TEMP) TO(TEMPGRP)
```

You cannot rename a resource definition to a name that already exists in the target group.

UNLOCK

Remove the lock from a group or a list of definitions.

UNLOCK syntax

```
►►──UNLock──┬─Group(groupname)─┬──────────────────────────────────────────────►◄
            └─List(listname)───┘
```

Description

The UNLOCK command is the only RDO command that can remove a lock on a list or group put there by use of the RDO LOCK command.

You can UNLOCK a nonexistent group or list.

You must specify either the GROUP or the LIST option, even if you are unlocking the current group or list, because the UNLOCK command can be used with both.

For more information about UNLOCK, see “Controlling access to a group or list—LOCK and UNLOCK” on page 19.

Controlling access to a group or list—LOCK and UNLOCK

The LOCK and UNLOCK commands enable you to control update access to a group or list so that only operators with the same operator identifier can make changes.

The lock is held on the CSD file and remains in effect across restarts of CICS. The lock is owned by the user, who is identified by a combination of the CICS generic applid (specified by the APPLID system initialization parameter), and the user’s operator identifier (OPIDENT).

The OPIDENT is the one associated with the user when he or she signs on to the terminal used for RDO. For further information on OPIDENT, see the *CICS RACF Security Guide*. Any user who is not signed on or who has a different OPIDENT is not allowed to perform any operation that would change the locked group. However, any user is allowed to do the following things to a locked group:
- CHECK
- COPY
- DISPLAY
- INSTALL
- VIEW

The lock can be removed, using the UNLOCK command, only by a user on the same system and with the same operator identifier.

It would be wise to put a lock on your group of TYPETERMs and on your group of AUTINSTMODEL TERMINALs.

Options

Group(*groupname*)
specifies the group to be unlocked. A generic group name is not accepted.

List(*listname*)
specifies the list to be unlocked. A generic list name is not accepted.

Examples

To unlock a group G1:

```
UNLOCK GROUP(G1)
```

To unlock a LIST L1:

```
UNLOCK LIST(L1)
```

USERDEFINE

Create a new resource definition.

USERDEFINE syntax

```
►►──USerdefine──┬─Connection(name)───┬──Group(groupname)──attribute list(newvalue)──────────►◄
                ├─DB2Conn(name)──────┤
                ├─DB2Entry(name)─────┤
                ├─DB2Tran(name)──────┤
                ├─DOctemplate(name)──┤
                ├─Enqmodel(name)─────┤
                ├─File(name)─────────┤
                ├─Journalmodel(name)─┤
                ├─Lsrpool(name)──────┤
                ├─Mapset(name)───────┤
                ├─PARTItionset(name)─┤
                ├─PARTNer(name)──────┤
                ├─PROCesstype(name)──┤
                ├─PROFile(name)──────┤
                ├─PROGram(name)──────┤
                ├─Requestmodel(name)─┤
                ├─Sessions(name)─────┤
                ├─TCpipservice(name)─┤
                ├─TDqueue(name)──────┤
                ├─TErminal(name)─────┤
                ├─TRANClass(name)────┤
                ├─TRANSaction(name)──┤
                ├─TSmodel(name)──────┤
                └─TYpeterm(name)─────┘
```

Description

USERDEFINE is an alternative to the DEFINE command. Instead of using CICS-supplied default values, USERDEFINE uses your own defaults. Otherwise it operates in exactly the same way as DEFINE.

To set up your own defaults, use DEFINE to create a dummy resource definition named USER in a group named USERDEF. Each dummy resource definition must be complete (for example, a transaction definition must name a program definition, even though you always supply a program name when you USERDEFINE a transaction). You need not install the dummy resource definitions before using USERDEFINE.

Do this for each type of resource for which you want to set default values. Each of them is named USER, but this does not matter because the fact that they are definitions of different resource types makes them unique.

So you could have the following resources in your USERDEF group:
- CONNECTION(USER)
- DB2CONN(USER)
- DB2ENTRY(USER)
- DB2TRAN(USER)
- DOCTEMPLATE(USER)
- ENQMODEL(USER)
- FILE(USER)
- JOURNALMODEL(USER)

- LSRPOOL(USER)
- MAPSET(USER)
- PARTITIONSET(USER)
- PARTNER(USER)
- PROCESSTYPE(USER)
- PROFILE(USER)
- PROGRAM(USER)
- REQUESTMODEL(USER)
- SESSIONS(USER)
- TCPIPSERVICE(USER)
- TDQUEUE(USER)
- TERMINAL(USER)
- TRANCLASS(USER)
- TRANSACTION(USER)
- TSMODEL(USER)
- TYPETERM(USER).

This example is reviewed in the 'Examples' section for this command.

Options

Attribute list
The attribute list depends on the resource type being defined; some resources have attributes that must be included in the definition. See "Part 5. RDO resource types and their attributes" on page 151 for a description of the attributes and default values of each resource type. Attributes that you do not specify are given default values.

Group(*groupname***)**
The name of the group to be defined.

Examples

Assembler programmers at an installation have created a dummy program definition called USER with Assembler as the default language. They use USERDEFINE to define their programs to CICS.

First you must define a program called USER in group USERDEF. You could do this with the command:

```
CEDA DEFINE PROGRAM(USER) GROUP(USERDEF)
```

The following figure shows the panel you would see as a result of this command:

```
  DEFINE PROGRAM(USER)
GROUP(USERDEF)            CICS RELEASE = 0530
  OVERTYPE TO MODIFY
   CEDA  DEFine
    PROGram        :
USER
    Group          :
USERDEF
    DEscription  ==>                                                    .
    Language     ==>                   CObol | Assembler | Le370 | C | Pli
    RELoad       ==> No                No | Yes
    RESident     ==> No                No | Yes
    USAge        ==> Normal            Normal | Transient
    USElpacopy   ==> No                No | Yes
    Status       ==> Enabled           Enabled | Disabled
    RSl            : 00                0-24 | Public
    Cedf         ==> Yes               Yes | No
    DAtalocation ==> Below             Below | Any
 I New group USERDEF created
                                                 SYSID=ABCD    APPLID=DBDCCICS
   DEFINE SUCCESSFUL                      TIME:  11.24.39   DATE:  97.359
 PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Type in `ASSEMBLER` as the LANGUAGE option and press ENTER:

```
  OVERTYPE TO MODIFY
   CEDA  USerdefine
    PROGram        :
USER
    Group          :
USERDEF
    DEscription  ==>
    Language     ==> Assembler         CObol | Assembler | Le370 | C | Pli
    RELoad       ==> No                No | Yes
    RESident     ==> No                No | Yes
    USAge        ==> Normal            Normal | Transient
    USElpacopy   ==> No                No | Yes
    Status       ==> Enabled           Enabled | Disabled
    RSl            : 00                0-24 | Public
    Cedf         ==> Yes               Yes | No
    DAtalocation ==> Below             Below | Any



                                                 SYSID=ABCD    APPLID=DBDCCICS
   DEFINE SUCCESSFUL                      TIME:  11.24.41   DATE:  97.359
 PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Now, each time you want to define a new program, you can use the USERDEFINE command to get the default value ASSEMBLER automatically. So, if you want to define a new program P2 in group GRP you enter the command:

```
CEDA USERDEFINE PROGRAM(P2) GROUP(GRP)
```

The following figure shows the panel resulting from this command.

```
  USERDEFINE PROGRAM(P2)
GROUP(GRP)
  OVERTYPE TO MODIFY
   CEDA  USerdefine
    PROGram        :
P2
    Group          :
GRP
    DEscription  ==>
    Language     ==> Assembler          CObol | Assembler | Le370 | C | Pli
    RELoad       ==> No                 No | Yes
    RESident     ==> No                 No | Yes
    USAge        ==> Normal             Normal | Transient
    USElpacopy   ==> No                 No | Yes
    Status       ==> Enabled            Enabled | Disabled
    RSl            : 00                 0-24 | Public
    Cedf         ==> Yes                Yes | No
    DAtalocation ==> Below              Below | Any
                                                              APPLID=DBDCCICS
   USERDEFINE SUCCESSFUL                  TIME:  11.25.48   DATE:  97.359
 PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

You see that the ASSEMBLER option has appeared for the LANGUAGE attribute. You can overtype the option values on this panel to complete the definition just as you can with the DEFINE command panel.

After you have set up your own defaults in a USER resource definition, anyone using the USERDEFINE command for that resource type gets those default values.

By renaming your USER to something else and defining your own dummy resource definition, you can use your own default values. Normally, however, your installation probably agrees on default values for standardization reasons, and puts a LOCK on the USERDEF GROUP.

VIEW

View the attributes of an existing resource definition.

VIEW syntax

```
►►──View──Group(groupname)──┬─────────────────────────┬────────────────────────────►◄
                            ├─ALl(name)───────────────┤
                            ├─Connection(name)────────┤
                            ├─DB2Conn(name)───────────┤
                            ├─DB2Entry(name)──────────┤
                            ├─DB2Tran(name)───────────┤
                            ├─DOctemplate(name)───────┤
                            ├─Enqmodel(name)──────────┤
                            ├─File(name)──────────────┤
                            ├─Journalmodel(name)──────┤
                            ├─Lsrpool(name)───────────┤
                            ├─Mapset(name)────────────┤
                            ├─PARTItionset(name)──────┤
                            ├─PARTNer(name)───────────┤
                            ├─PROFile(name)───────────┤
                            ├─PROCesstype(name)───────┤
                            ├─PROGram(name)───────────┤
                            ├─Requestmodel(name)──────┤
                            ├─Sessions(name)──────────┤
                            ├─TCpipservice(name)──────┤
                            ├─TDqueue(name)───────────┤
                            ├─TErminal(name)──────────┤
                            ├─TRANClass(name)─────────┤
                            ├─TRANSaction(name)───────┤
                            ├─TSmodel(name)───────────┤
                            └─TYpeterm(name)──────────┘
```

Description

The VIEW command lets you look at the resource definition attributes in the same way as the ALTER command. However, you cannot update any of the definitions.

Options

Group(*groupname***)**
specifies the group to be viewed. If no name is given, the current group is assumed.

Examples

```
VIEW TERMINAL(SZT1) GROUP(ZEMTERMS)

VIEW MAPSET(N20MAP01) GROUP(N20)
```

See Figure 8 on page 31 for an example of the VIEW panel.

Part 3. The CSDUP offline utility

This part explains what the CSDUP offline utility is, and how you use batch jobs submitted offline to change definitions in the CSD file. It contains the following chapters:

Chapter 5. CSD file utility program (DFHCSDUP)

The CSD file utility program, DFHCSDUP, is a component of resource definition online (RDO). DFHCSDUP is an offline utility program that allows you to read from and write to a CICS system definition (CSD) file, either while CICS is running or while it is inactive.

You can use the DFHCSDUP program to:
- ADD a group to the end of a named list in a CSD file
- ALTER attributes of an existing resource definition
- APPEND a group list from one CSD file to a group list in another, or in the same, CSD file
- COPY all of the resource definitions in one group or several generically named groups to another group or several other generically named groups in the same, or in a different, CSD file
- DEFINE a single resource, or a group of resources, on the CSD file
- DELETE from the CSD file a single resource definition, all of the resource definitions in a group, or all of the group names in a list
- EXTRACT data from the CSD file and pass it to a user program for processing
- INITIALIZE a new CSD file, and add to it CICS-supplied resource definitions
- LIST selected resource definitions, groups, and lists
- MIGRATE the contents of a table from a CICS load library to a CSD file
- | PROCESS to apply maintenance to the CSD file for a specific APAR
- REMOVE a single group from a list on the CSD file
- SCAN all of the IBM-supplied groups and user-defined groups for a resource. The definition of the matched resource in an IBM-supplied group is compared to the definition(s) of the corresponding matched resource in the user groups.
- SERVICE a CSD file when necessary
- UPGRADE the CICS-supplied resource definitions in a primary CSD file for a new release of CICS
- | USERDEFINE a single resource or a group of resources on the CSD file
- VERIFY a CSD file by removing internal locks on groups and lists

You can invoke the DFHCSDUP program in two ways:
- As a batch program (see page 77)
- From a user program running either in batch mode or in a TSO environment (see page 79)

Sharing the CSD file between this release and earlier releases

If you want to share the CSD file between CICS regions at different release levels, to enable you to share common resource definitions, you must update the CSD file from the higher level region—CICS Transaction Server for OS/390.

Any release of CICS can make some attributes of a previous release obsolete, and remove them from the CSD resource definitions. In CICS Transaction Server for OS/390, using the ALTER command on definitions that specify obsolete attributes does not cause the loss of these attributes, so you can safely update resource definitions from a CICS Transaction Server for OS/390 region.

If you intend sharing the CSD file between a CICS Transaction Server for OS/390 Release 3 region and a region running an earlier release, you can use the CICS Transaction Server for OS/390 CSD file utility, DFHCSDUP, to update resources that specify obsolete attributes. A compatibility option has been added for this purpose, which you must specify on the PARM parameter on the EXEC PGM=DFHCSDUP statement. You indicate the compatibility option by specifying COMPAT or NOCOMPAT. The default is NOCOMPAT, which means that you cannot update obsolete attributes. (See Figure 12 on page 77.)

"Appendix A. Obsolete attributes retained for compatibility" on page 497 discusses these obsolete attributes and their compatibility with earlier releases, and "Compatibility mode (CSD file sharing)" on page 12 has advice on CSD file sharing with several cross-references to other information that you may need.

Note: You cannot use the EXTRACT command of the CICS Transaction Server for OS/390 DFHCSDUP utility when the COMPAT option is specified.

Input to the DFHCSDUP program

Input to the DFHCSDUP program (see Figure 11) is from:
- A **primary CSD file**, which must be present, and have a ddname of DFHCSD
- Optionally, a **secondary CSD file**, for which you can specify any ddname
- A CICS table, as specified on the MIGRATE command

Output from the DFHCSDUP program

The result of running the DFHCSDUP program (see Figure 11) may be an updated primary file, or a print file.

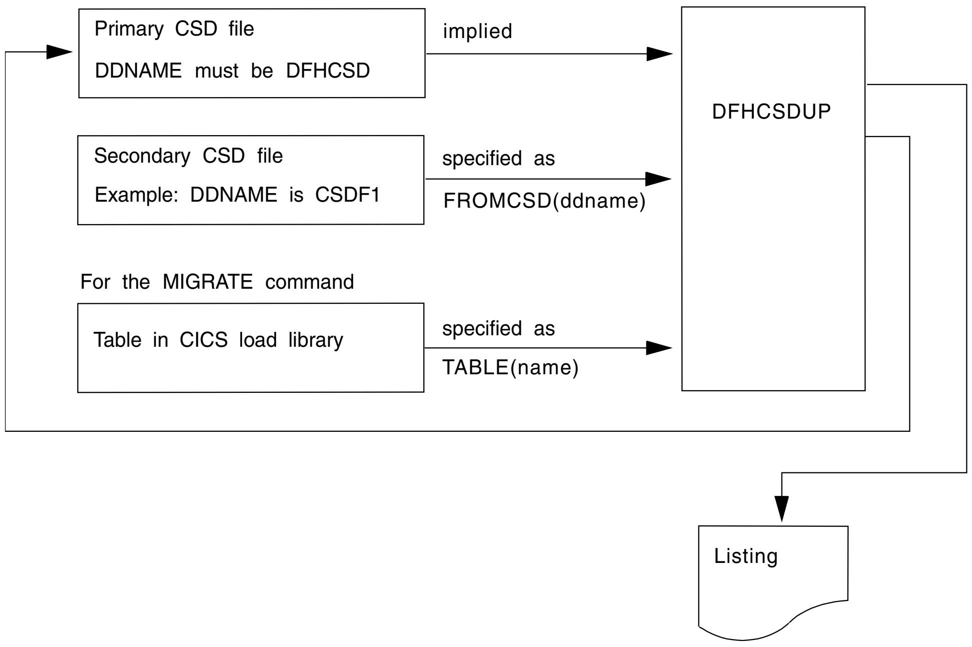


Figure 11. The DFHCSDUP offline utility program

Invoking DFHCSDUP as a batch program

The job in Figure 12 shows you an example of the job control statements you can use to invoke DFHCSDUP as a batch program.

```
//CSDJOB  JOB  accounting info,name,MSGLEVEL=1
//STEP1   EXEC PGM=DFHCSDUP,REGION=1024K,                          1
//             PARM='CSD(READWRITE),PAGESIZE(60),NOCOMPAT'
//STEPLIB DD   DSN=CICSTS13.CICS.SDFHLOAD,DISP=SHR
//*********************************************************************
//*       If you are running DFHCSDUP with the MIGRATE command,
//*       and your CICS load tables are not in CICSTS13.CICS.SDFHLOAD,
//*       concatenate your own private library here:
//*********************************************************************
//        DD  DSN=CICSTS13.CICS.userlib.tables,DISP=SHR
//DFHCSD   DD  UNIT=SYSDA,DISP=SHR,DSN=CICSTS13.CICS.DFHCSD
//SECNDCSD DD  UNIT=SYSDA,DISP=SHR,DSN=CICSTS13.CICS.SECNDCSD  2
//indd     DD  UNIT=SYSDA,DISP=SHR,DSN=extract.input.dataset  3
//outdd    DD  UNIT=SYSDA,DISP=SHR,DSN=extract.output.dataset  4  5
//* or
//outdd    DD  SYSOUT=A  4  5
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  *
   :
     DFHCSDUP commands  6
   :
/*
//
```

Figure 12. Sample job to run DFHCSDUP

Notes:

1 The EXEC statement should specify a suitable REGION size and a PARM parameter:

- **The REGION size**. A region size of 1024KB is generally recommended for the execution of the DFHCSDUP program. However, for the MIGRATE command, the table to be migrated is loaded into main storage, so the region size should be at least 1024KB plus the size of the largest table.
- **The PARM parameter**. Use this to specify any of the following options:

  UPPERCASE
  : specifies that you want all output from DFHCSDUP to be in uppercase. If you want all output to be in mixed case (the default), do not specify this option.

  CSD({READWRITE|READONLY})
  : specifies whether you want read/write or read-only access to the CSD file from this batch job. The default value is READWRITE. You must specify READONLY if you use the RLS override on the DD statement for the CSD file, and the CSD file is defined to be recoverable.

  PAGESIZE(nnnn)
  : specifies the number of lines per page on output listings. Values for nnnn are 4 through 9999. The default value is 60.

  NOCOMPAT|COMPAT
  : specifies whether the DFHCSDUP utility program is to run in compatibility mode (that is, whether it can update definitions that are obsolete in CICS Transaction Server for OS/390). The default is

NOCOMPAT, which means that you cannot update obsolete attributes. For further information about this option, see “Sharing the CSD file between this release and earlier releases” on page 75.

2 You need a DD statement for a secondary CSD file if you specify the FROMCSD parameter on an APPEND, COPY, or SERVICE command. The ddname for this DD statement is the name you specify on the FROMCSD parameter. The secondary CSD file must be a different data set from the primary; you must not define primary and secondary DD statements that reference the same data set.

3 If you specify the EXTRACT command, you may need to:

- Concatenate with STEPLIB the libraries that contain your USERPROGRAM programs.
- Include a DD statement for any input data set that is defined in your user program. For example, the CICS-supplied user program, DFH$CRFA, needs a DD statement with a ddname of CRFINPT.

  The input file specified by CRFINPT is needed by the user programs DFH$CRFx (where x=A for Assembler or x=P for PL/I) and DFH0CRFC (for COBOL) to supply the list of resource types or attributes for which you want a cross-reference listing. You can specify (in uppercase) any resource type known to CEDA, one resource type per line (starting in column 1). For example, your CRFINPT file may contain the following resource types (one per line) to be cross-referenced:

  ```
  PROGRAM
  TRANSACTION
  TYPETERM
  XTPNAME
  DSNAME
  ```

  For programming information about the use of the CRFINPT file by the programs DFH$CRFx or DFH0CRFC (for COBOL), see the *CICS Customization Guide*.

4 If you specify the EXTRACT command, you need to include the DD statements for any data sets that receive output from your extract program. The ddname is whatever ddname you define in the user program. The CICS-supplied sample programs need DD statements for the following ddnames:

Table 5. DD statements for the CICS-supplied sample programs

program name	ddname	example DD statement
DFH$CRFx or DFH0CRFC (VS COBOL II)	CRFOUT	`//CRFOUT DD SYSOUT=A`
DFH$FORx or DFH0FORC (VS COBOL II)	FOROUT	`//FOROUT DD SYSOUT=output.dataset`
DFH0CBDC	CBDOUT SYSABOUT	`//CBDOUT DD SYSOUT=A//SYSABOUT DD SYSOUT=A`

5 The output data sets in these examples are opened and closed for each EXTRACT command specified in SYSIN. If you are writing the output to a sequential disk data set, specify DISP=MOD to ensure that data is not overwritten by successive EXTRACT commands. Alternatively, provided you do not specify SYSOUT on the DD statement, you can change the OPEN statement in the program (for example, in the VS COBOL II versions, to OPEN EXTEND). For programming information about the CICS-supplied user programs, see the *CICS Customization Guide*.

6 Syntax

You can specify commands and options using abbreviations and mixed case, as given in the syntax box in the description of each command. If you enter an ambiguous command or option, the DFHCSDUP program issues a message indicating the ambiguity.

You can specify option values longer than one line, if you use the continuation character (an asterisk) at the end of a line (in column 72). Subsequent lines start in column 1. For example, you can use this facility to specify XTPNAME values of up to 128 hexadecimal characters.

General note

You can specify RLS=NRI or RLS=CR on the DD statement to specify that the CSD file is to be opened in RLS access mode. You must specify this RLS override if you want to run the DFHCSDUP utility while the CSD file is being accessed from CICS in RLS mode. You are advised to use the RLS=NRI option in preference to RLS=CR. If you specify the RLS override for a recoverable CSD file, you must also specify CSD(READONLY) in the PARM parameter.

Note: If you specify the RLS override, the data set that you specify must be defined for use in RLS mode (for example, the LOG parameter associated with the base cluster must be defined).

Invoking the DFHCSDUP program from a user program

By specifying the appropriate entry parameters, your program can cause the DFHCSDUP program to pass control to an exit routine at any of five exit points. The exits can be used, for example, to pass commands to the DFHCSDUP program, or to respond to messages produced by its processing.

You can run your user program:
- In batch mode
- Under TSO

Notes:
1. In a TSO environment, it is normally possible for the terminal user to interrupt processing at any time by means of an ATTENTION interrupt. In order to protect the integrity of the CSD file, the DFHCSDUP program does not respond to such an interrupt until after it has completed the processing associated with the current command. It then writes message number DFH5618 to the put-message exit, where this is available, and also to the default output file:

   ```
   AN ATTENTION INTERRUPT HAS BEEN REQUESTED DURING DFHCSDUP PROCESSING
   ```

   Your put-message exit routine can terminate the DFHCSDUP program, if desired. (You **must** supply a put-message routine if you want your operators to regain control after an ATTENTION interrupt.)

2. Suitably authorized TSO users can use the CEDA INSTALL transaction to install resources that have previously been defined with the DFHCSDUP program.

The CICS-supplied sample program, DFH$CUS1, illustrates how the DFHCSDUP program can be invoked from a user program. It is written as a command processor (CP) for execution under the TSO/E operating system.

The following sections outline the entry parameters of the DFHCSDUP program and the responsibilities of the user program. For programming information about invoking the DFHCSDUP program from a user program, see the *CICS Customization Guide*.

Entry parameters for the DFHCSDUP program

When invoking the DFHCSDUP program, your program passes a list of up to five parameters, as described below:

OPTIONS
A list of character strings, separated by commas. (The information passed here is that which would otherwise be passed on the PARM option of the EXEC statement of JCL.)

Note: A maximum of three options may be specified:

UPPERCASE
specifies that you want all output from DFHCSDUP to be in uppercase. If you want all output to be in mixed case (the default), do not specify this option.

CSD({READWRITE|READONLY})
specifies whether you require read/write or read-only access to the CSD file. The default value is READWRITE. You must specify READONLY if you use the RLS override on the DD statement for the CSD file, and the CSD file is defined to be recoverable.

PAGESIZE(nnnn)
specifies the number of lines per page on output listings. Valid values for nnnn are 4 through 9999. The default value is 60.

NOCOMPAT|COMPAT
specifies whether the DFHCSDUP utility program is to run in compatibility mode (that is, whether it can update definitions that are obsolete in CICS Transaction Server for OS/390). The default is NOCOMPAT, which means that you cannot update obsolete attributes. For further information about this option, see "Sharing the CSD file between this release and earlier releases" on page 75.

DDNAMES
A list of ddnames that, if specified, are substituted for those normally used by the DFHCSDUP program. You can specify RLS=NRI or RLS=CR on the DD statement to specify that the CSD file is to be opened in RLS access mode. You must specify the RLS override if you want to run the DFHCSDUP utility while the CSD file is being accessed from CICS in RLS mode. You are advised to use the RLS=NRI option in preference to RLS=CR. If you specify the RLS override for a recoverable CSD file, you must also specify CSD(READONLY) in the PARM parameter.

Note: If you specify the RLS override, the data set that you specify must be defined for use in RLS mode (for example, the LOG parameter associated with the base cluster must be defined).

HDING
The starting page number of any listing produced by the DFHCSDUP program. You can use this parameter to ensure that subsequent invocations produce logically numbered listings. If this parameter is not specified, the starting page number is set to 1.

The page number, if supplied, must be four numeric EBCDIC characters.

DCBs
The addresses of a set of data control blocks for use internally by the DFHCSDUP program. Any DCBs (or ACBs) that you specify are used internally, instead of those normally used by the DFHCSDUP program.

Note that if you specify both replacement DDNAMES and replacement DCBs, the alternative DCBs are used, but the alternative DDNAMES are disregarded.

EXITS
The addresses of a set of user exit routines to be invoked during processing of the DFHCSDUP program.

Responsibilities of the user program

Before invoking the DFHCSDUP program, your calling program must ensure that:

- AMODE(24) and RMODE(24) are in force
- S/370™ register conventions are obeyed
- If the EXITS parameter is passed, any programming environment needed by the exit routines has been initialized
- Any ACBs or DCBs passed for use by the DFHCSDUP program are OPEN

Commands for the DFHCSDUP program

This section describes the commands available with the DFHCSDUP utility program. Commands can be abbreviated, but the minimum abbreviation allowed differs from some of the CEDA command abbreviations.

Rules for the syntax and preparation of commands

Enter the commands in columns 1 through 71 of 80-character input records. You can specify option values longer than one line, if you use the continuation character (an asterisk) at the end of a line (in column 72). Subsequent lines start in column 1. For example, you can use this facility to specify XTPNAME values of up to 128 hexadecimal characters.

The command options can be specified by abbreviations and in mixed case, as shown in the command syntax under each command description. The minimum abbreviation is given in uppercase in the command syntax, with the optional characters given in lower case; for example:

```
ALter Connection(name) Group(groupname)
```

Leading blanks are ignored, and blanks between options and values are permitted.

Comment records are permitted; they must have an asterisk (*) in column 1. Comment material is not permitted on a record that contains a command.

Blank records between commands are ignored.

Follow the conventions for the names of groups and lists when coding the GROUP, LIST, TO, and TYPESGROUP parameters. If you use a generic specification for the GROUP or LIST parameter in the LIST command, you can use the symbols * and + in the same way as for CEDA.

The FROMCSD parameter must contain a valid ddname conforming to the rules for the JCL of the operating system.

An example of a valid sequence of commands is shown in Figure 13. Other examples of commands are given in the command descriptions that follow.

```
*                               SET UP INITIAL CSD FILE
INITialize
*
LIst LIst(DFHLIST) Objects
*                               UPGRADE FROM EARLIER RELEASE
UPgrade
*                               MIGRATE MAIN TABLES
MIgrate TAble(DFHFCTF1)
*
MIgrate TAble(DFHTCTT1)
*
MIgrate TAble(DFHDCTD1) TOGROUP(YOURGROUP)
*
LI Group(PPTM1)
LI G(SETM*)
*                                  CREATE GROUP PCTZ4
Copy G(PCTM1)  To(PCTZ4)
C G(SETMP3) T(PCTZ4) Replace
LI G(P++M+)
*                                  CREATE LIST MODLIST
APpend LIst(TESTLIST) TO(MODLIST) FRomcsd(CSDF1)
AP LI(SECLIST)  To(MODLIST) FR(CSDF1)
AP LI(DFHLIST)  To(MODLIST)
*
LI ALL OBJECTS
```

Figure 13. Sample commands of the DFHCSDUP program

Command processing following internal error detection

If you have provided a put-message-exit routine for the DFHCSDUP program, it is invoked whenever a message is issued. You can use this exit to respond to error messages produced by DFHCSDUP processing, when the DFHCSDUP program is invoked from a user program. The put-message-exit routine is not used if the DFHCSDUP program is running as a batch program. For programming information about the DFHCSDUP exits, see the *CICS Customization Guide*.

The reaction of the DFHCSDUP program to an error (with return code 8 or greater) depends on the nature of the error and on how the DFHCSDUP program is invoked.

If an error is detected while the DFHCSDUP program is running as a batch program, one of the following two reactions occurs:

1. If the error occurs during connection of the CSD file, no subsequent commands are completed.

2. If the error occurs elsewhere, no subsequent commands are executed other than LIST commands.

If an error is detected while the DFHCSDUP program is receiving commands from a get-command exit, all subsequent commands are processed if possible.

Generic naming

Generic names allow you to use one command to perform the same operation on many objects.

The asterisk can be used to mean 'all' resource names, groups, or lists. For example, DFHCSDUP EXTRACT TRANSACTION(DISC) GROUP(*) extracts transactions called DISC in any group where they occur in the CSD file. DFHCSDUP LIST GROUP(*) TRANSACTION(*) lists all the transactions in every group in the CSD file.

Another way to see groups without having to specify their exact names is to use the plus sign (+) as part of the group name to represent one character. For example, whereas DFHCSDUP LIST GROUP(DFH*) lists all groups beginning with DFH, DFHCSDUP LIST GROUP(DFH+++) lists only those groups that begin with DFH and are six characters long, DFHCSDUP LIST GROUP(DFH++++) lists those that begin with DFH and are seven characters long, and so on.

You can also use the ALL attribute in some commands, instead of specifying a resource type. The command then operates on all resource definitions that fit the type specified with ALL.

Which commands accept generic names?

Table 6 tells you which commands you can use generic naming with.

yes means that you may use a generic name or an actual name.

no means that you must use an actual name.

— means that generic naming is not applicable for this command.

Table 6. DFHCSDUP commands accepting generic names

Command	Generic resource name	Generic group name	Generic list name	ALL resource types
ADD	—	no	no	—
ALTER	yes	yes	—	no
APPEND	—	—	no	—
COPY	yes	no	—	yes
DEFINE	no	no	—	no
DELETE	no	no	no	yes
EXTRACT	—	yes	yes	—
INITIALIZE	—	—	—	—
LIST	—	yes	yes	yes
MIGRATE	—	—	—	—
PROCESS	—	—	—	—
REMOVE	—	no	no	—
SCAN	no	—	—	—

Table 6. DFHCSDUP commands accepting generic names (continued)

Command	**Generic resource name**	**Generic group name**	**Generic list name**	**ALL resource types**
SERVICE	—	—	—	—
UPGRADE	—	—	—	—
USERDEFINE	no	no	—	no
VERIFY	—	—	—	—

Chapter 6. DFHCSDUP commands

This chapter explains the syntax for each of the DFHCSDUP commands and shows examples of use. The commands are:

- "ADD" on page 86
- "ALTER" on page 87
- "APPEND" on page 89
- "COPY" on page 91
- "DEFINE" on page 93
- "DELETE" on page 95
- "EXTRACT" on page 97
- "INITIALIZE" on page 99
- "LIST" on page 100
- "MIGRATE" on page 102
- | "PROCESS" on page 106
- "REMOVE" on page 107
- "SCAN" on page 108
- "SERVICE" on page 110
- | "UPGRADE" on page 111
- | "USERDEFINE" on page 113
- "VERIFY" on page 115

Note: CICS may add new resource types and attributes across different releases. For similarly named fields, such as TRANSACTION and TRANCLASS, the minimum abbreviation may have to change from release to release. For example, at CICS/ESA 3.3 the minimum abbreviation for TRANSACTION was TR, whereas in this release it is now TRANS, to differentiate between that and TRANC for TRANCLASS.

RDO commands are validated online by the CEDA/B/C transactions, and confirmation for such ambiguities is required before the transaction can continue. This is not possible for input to DFHCSDUP however. To avoid batch operations failing because the minimum abbreviation has altered across CICS releases, it is recommended that the full resource type and attribute names are used when coding input to DFHCSDUP. While CICS cannot guarantee upward compatibility of the abbreviations, it always provides such compatibility for the full names of the fields.

ADD

Add a group to a list.

ADD syntax

```
►►──ADd──Group(groupname)──LIst(listname)──────────────────────────────────────►◄
```

Options

Group(*groupname*)
specifies the name of the group to be added. The name must not already exist in the list. A generic group name is not accepted. If you do not specify a group, the current group name is added.

LIst(*listname*)
specifies the name of the list to which the group is to be added. If the list does not already exist, a new one is created. If LIST is not specified, the group name is added to the current list if there is one. A generic list name is not accepted.

Examples

To create a list LA01, by adding a group to it:

```
ADD GROUP(GA001) LIST(LA01)
```

To add another group to list LA01:

```
ADD GROUP(GA002) LIST(LA01)
```

LA01 now looks like this:
GA001
GA002

ALTER

Change some or all of the attributes of an **existing** resource definition.

ALTER syntax

```
►►──ALter──┬─Connection(name)───┬──Group(groupname)──attribute list(new value)──────────────►◄
           ├─DB2Conn(name)──────┤
           ├─DB2Entry(name)─────┤
           ├─DB2Tran(name)──────┤
           ├─DOctemplate(name)──┤
           ├─Enqmodel(name)─────┤
           ├─File(name)─────────┤
           ├─Journalmodel(name)─┤
           ├─Lsrpool(name)──────┤
           ├─Mapset(name)───────┤
           ├─PARTItionset(name)─┤
           ├─PARTNer(name)──────┤
           ├─PROCesstype(name)──┤
           ├─PROFile(name)──────┤
           ├─PROGram(name)──────┤
           ├─Requestmodel(name)─┤
           ├─Sessions(name)─────┤
           ├─TCpipservice(name)─┤
           ├─TDqueue(name)──────┤
           ├─TErminal(name)─────┤
           ├─TRANClass(name)────┤
           ├─TRANSaction(name)──┤
           ├─TSmodel(name)──────┤
           └─TYpeterm(name)─────┘
```

Description

For information about the attributes that you can specify on the ALTER command for the various resource types, and for a description of the attributes and default values of each resource type, see "Part 5. RDO resource types and their attributes" on page 151.

Do not use ALTER to change the value of the attributes of a TYPETERM definition on which other attributes depend. If you make a mistake with DEVICE, SESSIONTYPE, or TERMMODEL, delete the definition and define a new one with the correct values.

You can specify null operand values, for example:

```
ALTER FILE(TEST) GROUP(ACT1) DESCRIPTION()
```

If an option for which you have specified a null value has a default, the default value is used. If an option does not have a default value, the definition acts as if the option has never been specified. In this example, if you list the resource definition for file TEST, it is shown with DESCRIPTION().

Changes to resource definitions in the CSD file do not take effect until you install the group in which the resource definition resides.

Generic naming in the ALTER command

The ALTER command accepts both generic resource names and group names.

For each resource in the CSD file matching the specified combination of resource name and group name, an ALTER is attempted. In the case of an individual ALTER failing, processing terminates when all attempts for the command have been processed.

Options

Attribute list
specifies the attributes to be altered.

Group(*groupname*)
specifies the name of the group containing the resource to be altered.

Resource(*name*)
specifies the resource whose attributes you want to alter. You can specify a generic name by using the characters + and *.

Examples

To make a program resident:

```
ALTER PROGRAM(ERR01) GROUP(GENMODS) RESIDENT(YES)
     DATALOCATION()
```

To make all programs in the group GENMOD resident:

```
ALTER PROGRAM(*) GROUP(GENMOD) RESIDENT(YES)
    DATALOC()
```

APPEND

Add the groups in one list to the end of another list.

APPEND syntax

```
►►──APpend──FRomcsd(ddname)──LIst(listname1)──To(listname2)──────────────────────►◄
```

Description

No duplicate group names are allowed in a list. If DFHCSDUP finds any duplicate names during the APPEND operation it ignores them, and they are not appended. The DFHCSDUP output listing contains a warning message if this happens.

Options

FRomcsd(*ddname***)**
specifies the ddname of the secondary CSD file from which you are appending *listname1*.

LIst(*listname1***)**
specifies the name of the list that is appended. Do not use a generic list name.

The list being appended can be on the primary CSD file, or on another CSD file. If you are appending from another CSD file, you must identify it by specifying the FROMCSD parameter.

To(*listname2***)**
specifies the name of the list to which you want the group names appended. If you are appending from another CSD file, you can give this list the same name as the one you are appending from. Do not use a generic list name.

If this target list already exists, the source list is appended to the end of it. If the target list does not exist, it is created. (In effect, you are copying the source list.)

Examples

A list called LISTA contains the following groups:
GB001
GB002
GB003

A list called LISTB contains the following groups:
G001
G002
G003

Append LISTB to LISTA, like this:

```
APPEND LIST(LISTB) TO(LISTA)
```

After this, LISTA contains the following groups, in this order:
GB001
GB002
GB003
G001
G002

G003

and LISTB still contains:
G001
G002
G003

COPY

Copy a resource definition, either within the same group or to a different group.

Note: Single resources cannot be copied as in the CEDA version of the COPY
command.

#

COPY syntax

```
►►──Copy──Group(groupname1)──To(groupname2)──┬─────────┬──FRomcsd(ddname)──────────────►◄
                                             ├─Replace─┤
                                             └─MErge───┘
```

Description

The COPY command copies all the resource definitions in **groupname1** to **groupname2**. The group to be copied (*groupname1*) can be on the primary CSD, or it can be on the CSD file specified by the FROMCSD parameter.

The group is copied to the group named on the TO parameter (*groupname2*) in the primary file. If this group already exists, the definitions from the source group (*groupname1*) are added to those already in the *groupname2* group. If the group specified on the TO parameter does not already exist, a new group of that name is created. However, if duplicate definitions exist in the two groups, the whole copy operation fails unless you specify REPLACE or MERGE to indicate how duplicates should be handled.

Generic naming in the COPY command

The COPY command accepts generic group names, both on the GROUP option and on the TO option, subject to the following rules:

- The only generic character permitted on the COPY command is the asterisk (*) symbol.
- The prefix length of *groupname1* must be equal to or greater than the prefix length of *groupname2*. Thus COPY GROUP(DFHCOMP*) TO(USRCMP*) is valid, but COPY GROUP(DFHCO*) TO(USRCOMP*) is not.

You can use the asterisk (*) symbol to copy from generically named groups to other generically named groups or from generically named groups to a specific group, as shown on page 92.

Note: There is no AS parameter as in the CEDA version of the COPY command.

The DFHCSDUP output listing tells you which definitions were copied, and what happened if duplicates were found.

Options

FRomcsd(*ddname*)
: specifies the ddname of the secondary CSD file from which you are copying *groupname1*.

Group(*groupname1*)
: specifies the name of the group to be copied. You can specify a generic name by using an asterisk (*). See “Generic naming in the COPY command” for details.

MErge
If *groupname2* already exists and duplicate definitions occur, the original definitions in *groupname2* are preserved.

Replace
If *groupname2* already exists and duplicate definitions occur, the definitions in *groupname1* replace those in *groupname2*.

To(*groupname2***)**
specifies the name of the group to which the definitions are copied. If you are copying from another CSD file, you can give this group the same name as the one you are copying from. You can specify a generic name by using an asterisk (*). See "Generic naming in the COPY command" on page 91 for details.

Examples

The following example copies a group named GA001 to a group named GA002, which already exists, replacing any duplicate resource definitions with those in group GA001.

```
COPY GROUP(GA001) TO(GA002) REPLACE
```

The following example copies group GA003 to group GA004, but if any duplicate definitions occur, preserves the group GA004 definitions.

```
COPY GROUP(GA003) TO(GA004) MERGE
```

The following example copies all the CICS-supplied groups to user-named groups with a prefix of USR, with the result that DFHOPER becomes USROPER, DFHSTAND becomes USRSTAND, and so on.

```
COPY GROUP(DFH*) TO(USR*)
```

The following example copies every group starting with ABCD to the group called NEWGROUP:

```
COPY GROUP(ABCD*) TO(NEWGROUP)
```

DEFINE

Create new resource definitions.

DEFINE syntax

```
►►──DEFine──┬─Connection(name)───┬─Group(groupname)──attribute list(newvalue)──────────────►◄
            ├─DB2Conn(name)──────┤
            ├─DB2Entry(name)─────┤
            ├─DB2Tran(name)──────┤
            ├─DOctemplate(name)──┤
            ├─Enqmodel(name)─────┤
            ├─File(name)─────────┤
            ├─Journalmodel(name)─┤
            ├─LSRpool(name)──────┤
            ├─Mapset(name)───────┤
            ├─PARTItionset(name)─┤
            ├─PARTNer(name)──────┤
            ├─PROCesstype(name)──┤
            ├─PROFile(name)──────┤
            ├─PROGram(name)──────┤
            ├─Requestmodel(name)─┤
            ├─Sessions(name)─────┤
            ├─TCpipservice(name)─┤
            ├─TDqueue(name)──────┤
            ├─TErminal(name)─────┤
            ├─TRANClass(name)────┤
            ├─TRANSaction(name)──┤
            ├─TSmodel(name)──────┤
            └─TYpeterm(name)─────┘
```

Options

Attribute list
The attribute list depends on the resource type being defined; some resources have attributes that must be included in the definition. For a description of the attributes and default values of each resource type, see "Part 5. RDO resource types and their attributes" on page 151. Attributes that you do not specify are given default values.

Group(*groupname*)
specifies the name of the group containing the resource definition to be altered. Do not use a generic group name. If you specify the name of a group which does not already exist, the group is created.

Resource(*name*)
specifies the name of the resource you want to define. Do not use a generic resource name. The resource option must always be the first operand of the DEFINE command.

Examples

You can use the same name for more than one resource definition in a group, if the definitions are for different resource types. For example:

```
DEFINE PROGRAM(N28A) GROUP(N28APPL)
DEFINE TRANSACTION(N28A) GROUP(N28APPL)

DEFINE TERMINAL(USER) GROUP(USERDEF)
DEFINE PROGRAM(USER) GROUP(USERDEF)
```

The next example defines two consoles to CICS. (You do not need continuation symbols if a definition spans several lines).

```
DEFINE TERMINAL(CON0)     GROUP(CONTERMS)
       CONSOLE(00)        TYPETERM(DFHCONS)
       DESCRIPTION(MVS CONSOLE ID 00, FOR ISSUING
          JCL COMMANDS)
DEFINE TERMINAL(CON1)     GROUP(CONTERMS)
       CONSOLE(01)        TYPETERM(DFHCONS)
       DESCRIPTION(MVS CONSOLE ID 01, MVS MASTER
          CONSOLE)
```

The INITIALIZE command generates a TYPETERM definition, but not a TERMINAL definition, for a console. You must have at least one console defined in order to issue MVS MODIFY commands to CICS. Console id 00 is used to issue commands using MVS job control language, and by authorized programs that use that MGCR macro to issue MVS commands.

DELETE

Delete a single resource definition in a group, all the resource definitions in a group, or all the group names in a group list.

DELETE syntax

```
►►──DELete──┬─List(listname)──────────────────────────────────────────┬──────────────────►◄
            └─Group(groupname)────┬─All─────────────────┬──┬────────┬─┘
                                  ├─Connection(name)────┤  └─REMove─┘
                                  ├─DB2Conn(name)───────┤
                                  ├─DB2Entry(name)──────┤
                                  ├─DB2Tran(name)───────┤
                                  ├─DOctemplate(name)───┤
                                  ├─Enqmodel(name)──────┤
                                  ├─File(name)──────────┤
                                  ├─Journalmodel(name)──┤
                                  ├─Lsrpool(name)───────┤
                                  ├─Mapset(name)────────┤
                                  ├─PARTItionset(name)──┤
                                  ├─PARTNer(name)───────┤
                                  ├─PROCesstype(name)───┤
                                  ├─PROFile(name)───────┤
                                  ├─PROGram(name)───────┤
                                  ├─Requestmodel(name)──┤
                                  ├─Sessions(name)──────┤
                                  ├─TCpipservice(name)──┤
                                  ├─TDqueue(name)───────┤
                                  ├─TErminal(name)──────┤
                                  ├─TRANClass(name)─────┤
                                  ├─TRANSaction(name)───┤
                                  ├─TSmodel(name)───────┤
                                  └─TYpeterm(name)──────┘
```

Description

Deleting a resource definition is different from removing a group from a list (see "REMOVE" on page 107). A deleted resource definition really does disappear from the CSD file.

When you DELETE the last resource in a group, the group is automatically deleted. An empty group cannot exist.

You cannot delete the definitions of groups and lists supplied by IBM.

If you delete a list, the definitions of the resources within the groups contained in the list are not deleted. To do this, you must also delete each group individually.

Options

Group(*groupname*)
: If this is specified alone, it indicates the name of the group to be deleted. If a resource is also specified, it indicates the group to which the resource belongs. Do not use a generic group name.

List(*listname*)
: specifies the name of the list to be deleted. Do not use a generic list name.

REMove
specifies that, when a group is deleted, the group is to be removed from all lists that had contained it.

RESource(*name*)
specifies the name of the resource to be deleted. Do not use a generic resource name.

This operand can be used only with the GROUP option.

Examples

A list in the primary CSD file called LISTA contains the following groups:
GB001
GB002

Group GB001 contains the following resource definitions:

TERMINAL(CON0)
TERMINAL(CON1)
TERMINAL(TEST)

The following command deletes the resource definition for the terminal TEST from group GB001:

```
DELETE TERMINAL(TEST) GROUP(GB001)
```

The following command deletes all the resource definitions in group GB002:

```
DELETE GROUP(GB002)
```

This leaves only group GB001 in the group list LISTA. The following command deletes all group names in the group list LISTA:

```
DELETE LIST(LISTA)
```

Note: The resource definitions in the groups in LISTA are not deleted.

EXTRACT

Extract a resource definition, group, or list from the CSD file.

EXTRACT syntax

```
►►──EXtract──┬─Group(groupname)─┬──┬─USerprogram(DFHxCRFy)───────────────┬──┬─────────┬──────────►◄
             └─LIst(listname)───┘  ├─USerprogram(DFHxFORy)───────────────┤  └─Objects─┘
                                   ├─USerprogram(DFH0CBDC)───────────────┤
                                   └─USerprogram(user-written program)───┘
```

Description

You can use the EXTRACT command to extract resource definition data from the CSD file, either from a list or from a group, and invoke a user program to process the extracted data. You specify the user program on the USERPROGRAM parameter.

Note: For programming information about coding user programs for the EXTRACT command, see the *CICS Customization Guide*.

Options

Group(*groupname***)**
specifies only those resource definitions within the named group. You can specify a generic group name.

LIst(*listname***)**
specifies only those resource definitions within the groups contained in the named list. You can use a generic list name only if you are not using the OBJECTS option.

Objects
returns the detail of each resource definition. You can extract resource definition data at two levels of detail:

- Without the OBJECTS option, the command extracts either the names of all the groups within a specified list, or the names of all the resource definitions within a specified group.
- With the OBJECTS option, all the resource definition attributes are also extracted.

You must specify OBJECTS for the CICS-supplied sample user programs DFHxCRFy and DFHxFORy. It is optional for DFH0CBDC and user-written user programs.

USerprogram(*user-written program***)**
specifies the name of the user-written program that is to process the data retrieved by the EXTRACT command. You must supply a USERPROGRAM value.

CICS supplies three types of sample user program: DFHxCRFy, DFHxFORy, and DFH0CBDC. The letter x in the program name is $ for assembler or PL/I and 0 for COBOL. The letter y in the program name denotes the programming language, where y=A is the assembler version, y=C is the COBOL version, and y=P is the PL/I version. Note that DFH0CBDC is supplied as VS COBOL II only.

All other user programs are available in source form, in CICSTS13.CICS.SDFHSAMP, and the assembler versions are also available in pregenerated form in CICSTS13.CICS.SDFHLOAD.

Examples

The following command uses the CICS-supplied user program, DFH0CBDC, to extract the resource definitions in group DFHTYPE and create the DEFINE commands needed to create them. It stores these commands in the file specified by the CBDOUT DD statement.

```
EXTRACT GROUP(DFHTYPE) USERPROGRAM(DFH0CBDC) OBJECTS
```

INITIALIZE

Prepare a newly defined data set for use as a CSD file.

INITIALIZE syntax

```
►►──INITialize──────────────────────────────────────────────────────────►◄
```

Description

You must initialize your CSD file before you can use any of the other DFHCSDUP commands, or the RDO transactions. After you have initialized your CSD file, you do not need to execute this function again.

The standard entries for the CICS-supplied resource definitions are created on the CSD file. The INITIALIZE command arranges these definitions into groups, and defines these groups in a group list named DFHLIST. This list contains only the CICS-supplied groups that are required by a CICS system.

CICS supports RDO for transient data. The DFHDCTG group contains sample definitions of all the CICS-supplied queues. You can add the names of other queues that you want to be installed at the same time to DFHDCTG. Place DFHDCTG at the top of DFHLIST so that the queues become available for use at the earliest possible point during CICS initialization.

If you use another group to install the CICS-supplied queues, make sure that this group is at the top of the first list to be installed using GRPLIST as part of an initial or cold start.

You can put other transient data resource definitions into different groups, from which they can be installed either during an initial or cold start, or at some point after initialization has completed.

INITIALIZE also creates a control record at the start of the CSD file. This record contains fields identifying the CICS release and the current level of service applied to the CSD. It also has fields containing the date and time of creation of the CSD file, and the date and time the file was last updated. Both these fields appear on the hard copy listing of the CSD file produced by the LIST command.

If you want to prepare a newly defined recoverable data set for use as a CSD file, you must INITIALIZE it using non-RLS mode, because a recoverable data set cannot be opened for output from batch in RLS mode, but the data set needs to be opened for output in order to initialize it.

LIST

Produce listings of the current status of the CSD file.

LIST syntax

```
        ┌─All─┐
►►─LIst─┼─────┼─────────────┬─────────┬──────────────────────────────────────►◄
        ├─Group(groupname)─┤ └─Objects─┘
        └─LIst(listname)───┘
```

Description

The listings are output to the SYSOUT data set, along with the messages issued by the command processing. The result is to print the contents of all the qualifying groups or lists.

Options

Group(*groupname*)
specifies only those resource definitions within the named group. You can specify a generic group name.

LIst(*listname*)
specifies only those resource definitions within the groups contained in the named list. You can use a generic list name only if you are not using the OBJECTS option (the only command where a generic list name is not acceptable is LIST LIST(*listname*) OBJECTS).

Objects
specifies the level of detail required for each resource definition. You can extract resource definition data at two levels of detail:

- Without the OBJECTS option, the command extracts either the names of all the groups within a specified list, or the names of all the resource definitions within a specified group.
- With the OBJECTS option, all the resource definition attributes are also extracted.

Examples

The listings produced by the various commands are as follows:

- LIST ALL
  – Names of defined lists and groups
  – Summary of lists
  – Summary of groups

  This prints summaries of all the definitions of lists and groups that exist on the CSD file.
- LIST ALL OBJECTS
  – Names of defined lists and groups
  – Summary of lists
  – Summary of groups
  – Objects in groups

  This prints summaries of all the definitions of lists and groups that exist on the CSD file, together with the properties of the resources in all the groups.

- LIST GROUP(*groupname*) (group name may be generic)
  - Summary of groups

  This summarizes the names of all the resources in one or more groups. They are organized within each group into resource type categories (for example, map sets, programs, and so on).
- LIST GROUP(*groupname*) OBJECTS (group name may be generic)
  - Summary of groups (see above)
  - Objects in groups

  This enables you to tabulate the properties of the resources, again organized according to resource type. The creation time for each resource is given, together with all its attributes, as originally set up by using DEFINE and ALTER commands, or by migrating it from a CICS table. The properties of transactions and profiles are arranged in the same subcategories that appear on the CEDA DEFINE screen.
- LIST LIST(*listname*) (list name may be generic)
  - Summary of lists

  The contents of one or more group lists are tabulated. The groups appear in the same sequence as their position in the list. This order is set by the commands ADD and APPEND, which were used in the CEDA transaction to build the list.
- LIST LIST(*listname*) OBJECTS (generic list name not allowed)
  - Summary of lists (see above)
  - Objects of groups in list

  This enables you to tabulate the properties of all the resources to be defined in a CICS system at startup time. These are identified by the list name or names specified in the GRPLIST=(*list1*,*list2*,*list3*,*list4*) system initialization parameter. The names of all the groups in the list appear in the summary of lists. Then, for each group contained in the list, the properties of the individual resources in the group are tabulated.

  The ‘Objects in Groups in Lists’ tabulation arranges the groups in the same order as they were added to the group list. This order matters if duplication occurs, when definitions of the same resource may exist in more than one group. If a list of this type is used at system startup time, the resource definitions used when there is duplication are those belonging to the group that is latest in the list.

MIGRATE

Transfer the contents of a DCT, an FCT, an RCT, a TCT or a TST from a CICS load library to the CSD file.

MIGRATE syntax

```
►►──MIgrate──TAble(tablename)──┬─────────────────────────────┬──┬──────────────────────┬──────────►◄
                               └─TYpesgroup(typesgroupname)─┘  └─TOGROUP(groupname)─┘
```

Description

The contents of a table are transferred as one group, or as a set of several groups, containing definitions.

When migrating large tables, make sure you allocate a sufficiently large region for the largest table to be loaded.

If you are migrating a table from an earlier release of CICS, you must specify the COMPAT option on the DFHCSDUP PARM of the EXEC statement to prevent migration errors.

- **To transfer a DCT**, the format is:

  ```
  MIGRATE TABLE(tablename) [TOGROUP(groupname)]
  ```

  where TABLE(*tablename*) identifies the name of the table in the load library (DFHDCTxx).

  The contents of a table are transferred as one group, or as a set of several groups, containing definitions. When migrating large tables, make sure you allocate a sufficiently large region for the largest table loaded. For migration purposes, DCTs must be link-edited with AMODE(24) RMODE(24). To ensure this, you must specify a DFHDCT TYPE=(INITIAL,MIGRATE) statement in your DCT. Failure to do so causes the DFHDCT macro to force AMODE(31), which results in errors when running DFHCSDUP.

  The result is a set of groups containing TDQUEUE resource definitions. You can specify each group using the macro:

  ```
  DFHDCT TYPE=GROUP,GROUP=xxxxxxxx
  ```

  which you insert in the DCT source instructions before you assemble them for migration. All definitions **after** such a TYPE=GROUP macro (up to the next TYPE=GROUP macro) go into the group named by GROUP=xxxxxxxx. Definitions that occur **before** the first such TYPE=GROUP macro are migrated to the default group. You can also specify that definitions are to be migrated to the default group by inserting the following macro in the DCT before the definition entries:

  ```
  DFHDCT TYPE=GROUP,GROUP=*DEFAULT
  ```

  You can use the TOGROUP parameter of the MIGRATE command to assign a specific name to the default group. If you do not specify TOGROUP, the name of the default group is taken from the table name. For example, if the table name is DFHDCT24, the name of the group created is DCT24.

- **To transfer an FCT**, the format is:

  ```
  MIGRATE TABLE(tablename)
  ```

The result is a set of groups containing file and LSR pool definitions. You can define each group using the macro:

```
DFHFCT TYPE=GROUP,GROUP=xxxxxxxx
```

which you insert in the FCT source instructions before you assemble the FCT for migration. Any file or LSR pool definitions that come before the first such TYPE=GROUP macro are migrated into a group named after the table name: for example, if the table name is DFHFCTxx, the group name is FCTxx.

- **To transfer an RCT**, the format is:

  ```
  MIGRATE TABLE(tablename) [TOGROUP(groupname)]
  ```

  where TABLE(*tablename*) identifies the name of the table in the load library (DFHRCTxx).

  The contents of a table are transferred as one group, or as a set of several groups, containing definitions. When migrating large tables, make sure you allocate a sufficiently large region for the largest table loaded. For migration purposes, RCTs must be link-edited with RMODE(24).

  The result is a set of groups containing DB2CONN, DB2ENTRY and DB2TRAN resource definitions. You can define each group using the macro:

  ```
  DSNCRCT TYPE=GROUP,GROUP=xxxxxxxx
  ```

  which you insert in the RCT source instructions before you assemble the RCT for migration. All definitions **after** such a TYPE=GROUP macro (up to the next TYPE=GROUP macro) go into the group named by GROUP=xxxxxxxx. Definitions that occur **before** the first such TYPE=GROUP macro are migrated to the default group. You can also specify that definitions are to be migrated to the default group by inserting the following macro in the RCT before the definition entries:

  ```
  DSNCRCT TYPE=GROUP,GROUP=*DEFAULT
  ```

  You can use the TOGROUP parameter of the MIGRATE command to assign a specific name to the default group. If you do not specify TOGROUP, the name of the default group is taken from the table name. For example, if the table name is DFHRCT24, the name of the group created is RCT24.

- **To transfer a TCT**, the format is:

  ```
  MIGRATE TABLE(tablename)   [TYPESGROUP(typesgroupname)]
  ```

  where TYpesgroup(*typesgroupname*) specifies the name of the group to contain the TYPETERM definitions obtained from the TCT.

  If this parameter is not specified, the TYPETERM definitions are put in the GROUP currently being created, with the TERMINAL definitions.

  The result is:

  1. A set of groups containing terminal definitions. You can define each group using the macro:

     ```
     DFHTCT TYPE=GROUP,GROUP=xxxxxxxx
     ```

     which you insert in the TCT source instructions before you assemble the TCT for migration. Any terminal definitions that come before the first TYPE=GROUP macro are migrated into a group named after the table name. If the table name is DFHTCTxx, the group name is TCTxx.

2. A group of TYPETERM definitions. These are derived from attributes of TYPE=TERMINAL macros which are often identical for many terminals. They are put into the CSD GROUP named in the TYPESGROUP parameter.

   The typeterm attributes of each TYPE=TERMINAL table macro are checked with existing TYPETERM definitions and if they don't match with any of these, a new TYPETERM is added to the CSD file.

   The existing TYPETERMs checked are:
   – TYPETERMs in the GROUP currently being created
   – TYPETERMs in the group specified in the TYPESGROUP parameter of the MIGRATE command.

   However, the scope of the checking is never extended to include any other TYPETERMs in other groups already on the CSD file. (Such groups may have been created using RDO or by a previous MIGRATE command.) For this reason, it is a good idea to use the TYPESGROUP parameter to avoid creating duplicate TYPETERMs in different groups. It is convenient to keep the TYPETERMs in a separate group anyway.

   TYPETERMs created on the CSD file during the migration are named systematically, in a way related to the TRMTYPE parameter of the original terminal definition. The name consists of a prefix (3–5 characters) with a 3-character suffix. For example, a TYPETERM defining attributes for a 3270 printer is named 3270P001. Variants with the same TRMTYPE are named 3270P002, and so on. The migration process ensures that this name is used as the TYPETERM parameter of every terminal definition that references it.

Note: Migrating your TCT does not cause an error if the destination group already exists. Only definitions that already exist are flagged by an error message; any new or additional definitions are added to the existing group.

- **To transfer a TST**, the format is:

```
MIGRATE TABLE(tablename)   [TOGROUP(groupname)]
```

where TABLE(*tablename*) identifies the name of the table in the load library (DFHTSTxx) and TOGROUP(*groupname*) specifies the name of the group to contain the definitions obtained from the TST.

The content of a table is transferred as a group containing TSMODEL definitions. When migrating large tables, make sure that you allocate a sufficiently large region for the largest table.

For migration purposes, TSTs must be link-edited with AMODE(24) RMODE(24). To ensure this, you must specify a DFHTST TYPE=(INITIAL,MIGRATE) statement in your TST. Failure to do so causes the DFHTST macro to force AMODE(31), which leads to errors when running DFHCSDUP.

You can use the TOGROUP parameter of the MIGRATE command to assign a specific name to the default group. If you do not specify TOGROUP, the name of the default group is taken from the TABLENAME. For example, if the tablename is DFHTSTJP, the name of the group created is TSTJP.

Notes:

1. TSMODEL definitions have a location attribute, either MAIN or AUXILIARY. Migration sets this to auxiliary (although you can change it later by updating the TSMODEL definition).

Before you define TSMODEL resource definitions to replace TST macros, you are able to specify MAIN or AUXILIARY on the WRITEQ TS API command, but this is ignored if a TSMODEL resource definition with a matching prefix is installed; the value supplied by the TSMODEL is used instead.

2. The TYPE=SHARED macro in the TST is different from the other TST macros in that it does not have a DATAID parameter on which you can specify a TS queue prefix. Thus, to map a TS request to a TS data sharing pool, CICS requires one of the following to be specified in addition to a TYPE=SHARED macro:
   - A TYPE=REMOTE macro that specifies a SYSIDNT that matches the SYSIDNT in a corresponding TYPE=SHARED macro.
   - A SYSID specified explicitly, either on the TS API command or set by an XTSEREQ global user exit program.

   If you use the second of these two methods, and do not specify a supporting TYPE=REMOTE entry in your TST, DFHCSDUP cannot migrate a TST TYPE=SHARED entry because it has no means of knowing the DATAID from which to create the corresponding PREFIX attribute in the TSMODEL. In this case DFHCSDUP issues message DFH5139 to indicate that a TYPE=SHARED entry has been ignored.

   DFHCSDUP also issues message DFH5139 if a TYPE=SHARED macro has a supporting TYPE=REMOTE entry and has been successfully migrated to a TSMODEL with the POOLNAME shared attribute. The reason for the message in this case is that application programs that explicitly specify a SYSID, or which rely on a SYSID being specified in a global user exit program, cannot use TSMODELs, and continue to require a TST to route the request to a data sharing pool. Check that your application programs work with migrated TSMODELs for shared queues in the same way as with the migrated TST. For more information about the specific use of a SYSID on TS commands, see the *CICS Application Programming Reference*.

Options

TAble(*tablename***)**
specifies the name in the load library of the table you want to migrate (that is, DFHDCTxx, DFHFCTxx, DFHRCTxx, DFHTCTxx, or DFHTSTxx).

TOgroup(*groupname***)**
specifies the name of the group to which the definitions are to be migrated. This is for use with DCT, RCT and TST only.

TYpesgroup(*typesgroupname***)**
specifies the name of the group to which the TYPETERM definitions are to be migrated. For use with TCT migration only.

PROCESS

Apply maintenance to the CSD file for a specific APAR.

PROCESS syntax

```
►►──PROCESS──Apar(aparnumber)──────────────────────────────────────────────►◄
```

Description

The PROCESS APAR command is used to apply maintenance to your CSD file for a specific APAR. Only use this command in accordance with the instructions in the associated PTF cover letter.

Options

Apar(*aparnumber***)**
The number of the APAR providing the maintenance, for example, PROCESS APAR(PQ12417) is used to apply maintenance for APAR PQ12417.

REMOVE

Remove a group name from a list.

REMOVE syntax

```
►►──Remove──Group(groupname)──LIst(listname)──────────────────────────────────►◄
```

Description

The group, and all its resource definitions, still exists on the CSD file.

| When the last group is removed from a list, the list is deleted.

Options

Group(*groupname*)
specifies the name of the group to be removed. Do not use a generic group name.

LIst(*listname*)
specifies the name of the list from which a group is to be removed. Do not use a generic list name. When the last group is removed from a list, the list no longer exists on the CSD file.

Examples

A list LL02 contains the following groups:

G001 G002 G003 G004

To remove group G003:

```
REMOVE GROUP(G003) LIST(LL02)
```

This leaves:

G001 G0023 G004

SCAN

SCAN all the IBM-supplied groups and user-defined groups for a specified resource. The definition of the matched resource in an IBM-supplied group is compared with the definition(s) of the corresponding matched resource in the user groups.

SCAN syntax

```
►►──SCAN──┬─Connection(name)───┬──┬───────────────────┬──────────────────────────────────►◄
          ├─DB2Conn(name)──────┤  └─ALIAS(aliasname)─┘
          ├─DB2Entry(name)─────┤
          ├─DB2Tran(name)──────┤
          ├─DOctemplate(name)──┤
          ├─Enqmodel(name)─────┤
          ├─File(name)─────────┤
          ├─Journalmodel(name)─┤
          ├─Lsrpool(name)──────┤
          ├─Mapset(name)───────┤
          ├─PARTItionset(name)─┤
          ├─PARTNer(name)──────┤
          ├─PROCesstype(name)──┤
          ├─PROFile(name)──────┤
          ├─PROGram(name)──────┤
          ├─Requestmodel(name)─┤
          ├─Sessions(name)─────┤
          ├─TCpipservice(name)─┤
          ├─TDqueue(name)──────┤
          ├─TErminal(name)─────┤
          ├─TRANClass(name)────┤
          ├─TRANSaction(name)──┤
          ├─TSmodel(name)──────┤
          └─TYpeterm(name)─────┘
```

Description

The SCAN command searches all the IBM-supplied groups in the CSD for a resource definition of a specified name and type. A message is issued with the results of the search. The user-defined groups are then searched for the same resource definition. The outcome of this can be one of the following:

- If an IBM-supplied group and one or more user-defined groups contain the resource definition, a comparison is made between the definition in the IBM-supplied group and the user group(s). A message is issued indicating whether the definition in the IBM-supplied group matches the definition(s) in the user group(s).
- If the resource definition is not found in the user-defined groups a message is issued.
- If the resource definition is not found in an IBM-supplied group but is found in one or more user-defined groups, a message is issued indicating the group(s) that contained it.

If ALIAS(*aliasname*) is specified, the user groups are searched using *aliasname*.

You can use the SCAN command to check for differences between IBM-supplied definitions that you have modified and the latest IBM-supplied versions after an upgrade.

Notes:

1. The compatibility groups DFHCOMPx are not scanned as part of the IBM-supplied groups but as user-defined groups.
2. The DESCRIPTION attribute is not used in the comparison.

Options

Alias(*aliasname*)
specifies the alias name of the resource type to be searched for in the user-defined groups.

This operand is optional.

Resource(*name*)
specifies the name of the resource type to be searched for in the IBM-supplied groups, and in the user-defined groups if ALIAS(*aliasname*) is not specified.

The resource option must always be the first operand of the SCAN command.

Examples

To search the CSD for transaction CEDA:

```
SCAN TRANSACTION(CEDA)
```

The result of this could look like:

```
 DFH5120 I PRIMARY CSD OPENED;  DDNAME: DFHCSD
 DFH5633 I TRANSACTION CEDA FOUND IN GROUP DFHSPI
 DFH5631 I TRANSACTION CEDA FOUND IN GROUP A1
           MATCHES THE IBM-SUPPLIED DEFINITION
           IN GROUP DFHSPI
 DFH5631 I TRANSACTION CEDA FOUND IN GROUP A2
           MATCHES THE IBM-SUPPLIED DEFINITION
           IN GROUP DFHSPI
 DFH5632 I TRANSACTION CEDA FOUND IN GROUP DFHCOMP1
           DOES NOT MATCH THE IBM-SUPPLIED DEFINITION
           IN GROUP DFHSPI
 DFH5101 I SCAN COMMAND EXECUTED SUCCESSFULLY.
```

To search the CSD for transaction CEDA with an alias name of AEDA:

```
SCAN TRANSACTION(CEDA) ALIAS(AEDA)
```

The result of this could look like:

```
 DFH5120 I PRIMARY CSD OPENED;  DDNAME: DFHCSD
 DFH5633 I TRANSACTION CEDA FOUND IN GROUP DFHSPI
 DFH5631 I TRANSACTION AEDA FOUND IN GROUP A3
           MATCHES THE IBM-SUPPLIED DEFINITION
           IN GROUP DFHSPI
 DFH5101 I SCAN COMMAND EXECUTED SUCCESSFULLY.
```

SERVICE

Carry out maintenance to your CSD file.

SERVICE syntax

```
►►──Service──FRomcsd(ddname)──LEvel(nnn)──────────────────────────────────────────►◄
```

Description

You might occasionally (between CICS releases) have to apply a service routine to carry out preventive or corrective maintenance to your CSD file. You do this by loading and running a special service program (DFHCUS1), which is supplied with CICS as a separately loadable module.

You can use the SERVICE command to create a new copy of the CSD file, from the existing CSD file. All the definitions are preserved, with the corrections (if any) applied.

Options

FRomcsd(*ddname*)
: specifies the ddname of the current CSD file, which for the purposes of the command is treated as the secondary CSD file.

LEvel(*nnn*)
: Associated with your CSD file is a current service level, initially set to 000 when the file was initialized. Applying the service routine causes the service level to be incremented in steps of one, from a “current level” to a “target level”.

  This operand specifies the target service level to which the CSD file is to be upgraded, and must be 1 higher than the current level of FROMCSD. Specify it as a 3-character integer; for example, LEVEL(001).

UPGRADE

Change the CICS-supplied resource definitions in a primary CSD file.

UPGRADE syntax

```
►►──UPgrade──┬──────────────────┬──┬─────────┬─────────────────────────────────►◄
             └─USing(filename)──┘  └─Replace─┘
```

Description

If your existing CSD file was initialized at CICS Transaction Server for OS/390 Release 2 or earlier, you must UPGRADE it using DFHCSDUP to obtain the RDO extensions in CICS Transaction Server for OS/390 Release 3. Upgrading ensures that the definitions in the DFH-groups are brought up to the level of CICS Transaction Server for OS/390 Release 3 function.

The upgrade command can also be used to apply any package of IBM-supplied resource definitions to the CSD file. For example, the definitions for the CICS sample programs and transactions can be transferred to the CSD file with the UPGRADE statement.

Note: If you have copied any CICS-supplied definitions to your own groups, these copies do **not** have any changes applied to them as a result of this command.

When you run the UPGRADE function of the CSD utility program (DFHCSDUP), you must ensure that you manually upgrade any IBM-supplied definitions that you may have modified. The safest way to do this is to copy the upgraded IBM-supplied definitions and reapply your modifications. This action is required because the UPGRADE command does not operate on your own groups, or on IBM groups that you have copied.

It is important to upgrade these modified definitions to ensure that they are defined correctly with non-default values for attributes that are new. If you fail to upgrade modified definitions, CICS assigns default values to any new attributes, and these may be inappropriate for IBM-supplied resource definitions.

If you are not sure whether your CSD contains any modified IBM definitions, use the DFHCSDUP SCAN function to compare the IBM-supplied resource definitions with any user-modified versions.

Options

Replace
You must specify the REPLACE option when you need to rerun the UPGRADE command (for example, because of a previous failure).

USing(*filename*)
Upgrading a CSD file for CICS Transaction Server for OS/390 does not require you to use the USING operand. All IBM-supplied definitions from **any** release are deleted and then the CSD file is initialized, so you do not need to say which release you came from. However, UPGRADE USING(*filename*) is used to

install IBM® features onto CICS. For example, UPGRADE USING(DFHRDJPN) is used to place the double-byte character set feature definitions onto the CSD file.

USERDEFINE

Create a new resource definition.

USERDEFINE syntax

```
►►──USerdefine──┬─Connection(name)────┬─Group(groupname)──attribute list(newvalue)──────────────►◄
                ├─DB2Conn(name)───────┤
                ├─DB2Entry(name)──────┤
                ├─DB2Tran(name)───────┤
                ├─DOctemplate(name)───┤
                ├─Enqmodel(name)──────┤
                ├─File(name)──────────┤
                ├─Journalmodel(name)──┤
                ├─Lsrpool(name)───────┤
                ├─Mapset(name)────────┤
                ├─PARTItionset(name)──┤
                ├─PARTNer(name)───────┤
                ├─PROCesstype(name)───┤
                ├─PROFile(name)───────┤
                ├─PROGram(name)───────┤
                ├─Requestmodel(name)──┤
                ├─Sessions(name)──────┤
                ├─TCpipservice(name)──┤
                ├─TDqueue(name)───────┤
                ├─TErminal(name)──────┤
                ├─TRANClass(name)─────┤
                ├─TRANSaction(name)───┤
                ├─TSmodel(name)───────┤
                └─TYpeterm(name)──────┘
```

Description

USERDEFINE is an alternative to the DEFINE command. Instead of using CICS-supplied default values, USERDEFINE uses your own defaults. Otherwise it operates in exactly the same way as DEFINE.

To set up your own defaults, use DEFINE to create a dummy resource definition named USER in a group named USERDEF. Each dummy resource definition must be complete (for example, a transaction definition must name a program definition, even though you always supply a program name when you USERDEFINE a transaction). You need not install the dummy resource definitions before using USERDEFINE.

Do this for each type of resource for which you want to set default values. Each of them is named USER, but this does not matter because the fact that they are definitions of different resource types makes them unique.

So you could have the following resources in your USERDEF group:
- CONNECTION(USER)
- DB2CONN(USER)
- DB2ENTRY(USER)
- DB2TRAN(USER)
- DOCTEMPLATE(USER)
- ENQMODEL(USER)
- FILE(USER)
- JOURNALMODEL(USER)

- LSRPOOL(USER)
- MAPSET(USER)
- PARTITIONSET(USER)
- PARTNER(USER)
- PROCESSTYPE(USER)
- PROFILE(USER)
- PROGRAM(USER)
- REQUESTMODEL(USER)
- SESSIONS(USER)
- TCPIPSERVICE(USER)
- TDQUEUE(USER)
- TERMINAL(USER)
- TRANCLASS(USER)
- TRANSACTION(USER)
- TSMODEL(USER)
- TYPETERM(USER).

This example is reviewed in the 'Examples' section for this command.

Options

Attribute list
The attribute list depends on the resource type being defined; some resources have attributes that must be included in the definition. See "Part 5. RDO resource types and their attributes" on page 151 for a description of the attributes and default values of each resource type. Attributes that you do not specify are given default values.

Group(*groupname*)
The name of the group to be defined.

Examples

Assembler programmers at an installation have created a dummy program definition called USER with Assembler as the default language. They use USERDEFINE to define their programs to CICS.

First you must define a program called USER in group USERDEF. You could do this with the command:

```
DEFINE PROGRAM(USER) GROUP(USERDEF) LANGUAGE(ASSEMBLER)
```

Now, each time you want to define a new program, you can use the USERDEFINE command to get the default value ASSEMBLER automatically. So, if you want to define a new program P2 in group GRP you enter the command:

```
USERDEFINE PROGRAM(P2) GROUP(GRP)
```

After you have set up your own defaults in a USER resource definition, anyone using the USERDEFINE command for that resource type gets those default values.

By renaming your USER to something else and defining your own dummy resource definition, you can use your own default values. Normally, however, your installation probably agrees on default values for standardization reasons, and puts a LOCK on the USERDEF GROUP.

VERIFY

Remove internal locks on groups and lists.

VERIFY syntax

```
►►──VERIFY──────────────────────────────────────────────────────────────►◄
```

Description

Use the VERIFY command only when the CSD file is not in use and no backout processing is pending on the CSD file; preferably use it only when no CICS systems that may use the CSD file are running. In particular, do not use the VERIFY command while CICS systems could be accessing the CSD file in RLS access mode.

VERIFY acts on the whole CSD file, and is for use in the extreme condition where internal lock records have been left behind. These records are normally removed when a function that changes the CSD file has been completed. However, this may not have happened if there was a system failure when the CEDA transaction was running, or if an offline utility failed to finish. The locks may prevent CEDA users from accessing certain groups and lists on the CSD file.

Note that VERIFY removes only the internal locks. It does not affect the normal user locks applied by the LOCK command in the CEDA transaction.

Part 4. Autoinstall

This part discusses the automatic installation of resources in CICS. It consists of the following chapters:

Chapter 7. Overview of autoinstall

To use a resource without autoinstall, you must have a definition for that resource installed in your CICS system, created using CEDA, DFHCSDUP, or a macro table. This clearly uses up time (to define and install every resource) and storage, because every definition occupies storage whether the resource is being used or not.

With autoinstall, you do not need to define and install every resource that you intend to use. Instead, CICS dynamically creates and installs a definition for you when a resource is requested. CICS bases the new definition on a "model" definition provided by you.

Autoinstall can be used for the following resources:

- APPC (LU6.2) connections (see "Chapter 10. Autoinstall for APPC connections" on page 141).
- Journals (see "Chapter 11. Autoinstall for other resources" on page 145). Journals can be used only after they have been autoinstalled with reference to a journalmodel.
- Mapsets (see "Chapter 11. Autoinstall for other resources" on page 145).
- Partitionsets (see "Chapter 11. Autoinstall for other resources" on page 145).
- Programs (see "Chapter 11. Autoinstall for other resources" on page 145).
- VTAM terminals (see "Chapter 8. Autoinstall for VTAM terminals" on page 121).

Autoinstall models

You must provide at least one **model** resource definition for each type of resource to be autoinstalled. When a resource is requested that does not have an installed definition, CICS creates a definition based on what you have specified in the model. You can have more than one model, depending on what properties you want the autoinstalled resources to have; for example, if you had 500 terminals all with the same properties and another 500 with a different set of properties, you could have two model terminal definitions, one for each of the two sets of properties.

Autoinstall control program

You control autoinstall by means of an **autoinstall control program**, either user-written or supplied by CICS. This program is responsible for, among other things, providing the model name or names to CICS, providing VTAM information to CICS for autoinstall for terminals, and so on.

CICS provides three autoinstall control programs; one for terminals; one for both terminals and connections; and one for programs, mapsets, and partitionsets. You can use the CICS-supplied ones or you can customize them to suit your installation.

These chapters also describe in detail what you must do to use autoinstall, and tell you about other considerations, such as what happens at installation time, and what happens at CICS restart.

Chapter 8. Autoinstall for VTAM terminals

This chapter explains how to implement autoinstall for VTAM terminals. It contains the following information:

- "Getting started with autoinstall"
- "Autoinstall considerations" on page 124
- "Autoinstall and VTAM" on page 126
- "Autoinstall recovery and restart" on page 129
- "The autoinstall control program" on page 133

If you have not used autoinstall for any resources before, read "Chapter 7. Overview of autoinstall" on page 119.

| For information on autoinstalling MVS consoles, see "Chapter 9. Autoinstall for
| MVS consoles" on page 135.

For information on autoinstalling connections and parallel sessions, see "Chapter 10. Autoinstall for APPC connections" on page 141.

Getting started with autoinstall

This list explains how to set up autoinstall for your VTAM terminals.

1. **Are your terminals eligible for autoinstall?**

   The terminals that **can** be autoinstalled are:
   - VTAM locally attached 3270 terminals (non-SNA), both displays and printers
   - VTAM logical unit type 0 terminals
   - VTAM logical unit type 1 terminals, including SCS printers
   - VTAM logical unit type 2 terminals
   - VTAM logical unit type 3 terminals
   - VTAM logical unit type 4 terminals
   - VTAM logical unit type 6.2 single-session terminals
   - TLX or TWX terminals using NTO

   Terminals that **cannot** be autoinstalled are:
   - Pipeline terminals
   - Automatic teller machines (3614 and 3624)
   - Non-VTAM resources
   - VTAM logical unit type 6.1 ISC and MRO sessions
2. **Should you use autoinstall?**

   You are likely to benefit from autoinstall if your system has:
   - A significant number of VTAM terminals
   - Frequent changes to your network
   - Many VTAM terminals logged off much of the time
   - Many VTAM terminals using other applications much of the time
   - Many VTAM terminals that need access to multiple, but unconnected, CICS systems

   Autoinstall might be less beneficial if you have:
   - A small, static network
   - Many terminals more or less permanently logged on to one CICS system
   - Many terminals logging on and off frequently
   - Many terminals logging on and off at the same time

3. **Decide which devices to autoinstall**.

   This decision depends on how you use your VTAM terminals. For example, a terminal that is logged on all the time can be autoinstalled, but you might choose to define it individually.

   An autoinstall logon is slower than a logon to a terminal individually defined to CICS, so if you switch continually between applications and have to log on to CICS frequently, you may require individual definitions for some terminals.

   You should also consider your use of automatic transaction initiation (ATI), terminal list tables (TLTs), and the intercommunication methods in use in your installation. See “Autoinstall considerations” on page 124.
4. **Create your TYPETERM and model TERMINAL definitions**.

   CICS supplies some TERMINAL and TYPETERM definitions; these are listed in “TYPETERM definitions in group DFHTYPE” on page 517 and “Model TERMINAL definitions in group DFHTERM” on page 523. You can use these definitions if they are suitable; if not, create your own using CEDA or DFHCSDUP.

   Define an autoinstall model for each different kind of terminal to be autoinstalled. Try to keep the number of definitions to a minimum, so that the autoinstall control program can be as simple as possible.

   When you create your definitions, consider whether you want to use the QUERY structured field (see page 333). It can help the autoinstall control program to choose which model on which to base a definition, and so speed up the autoinstall process.
5. **Redefine DFHZCQ**.

   For every region using autoinstall, redefine DFHZCQ to be RESIDENT(YES). (DFHZCQ is in the CICS-supplied group DFHSPI). See the *CICS Performance Guide* for guidance on why you should consider making programs resident.
6. **Ensure that your VTAM LOGMODE table entries are correct**.

   “Autoinstall and VTAM” on page 126 explains the relationship between CICS autoinstall and VTAM. For programming information, including a list of VTAM LOGMODE table entries, see the *CICS Customization Guide*.
7. **Design and write an autoinstall control program**.

   The terminal autoinstall control program is invoked by CICS every time there is a valid request for a TCT entry to be autoinstalled, and every time an autoinstalled TCT entry is deleted.

   For programming information about the autoinstall control program, see the *CICS Customization Guide*. “The autoinstall control program” on page 133 provides a summary of what the program is about.

   Before beginning your program, look at the CICS-supplied autoinstall control program DFHZATDX in group DFHSPI to see if it is suitable for what you want to do with autoinstall.
8. **Enable terminal autoinstall**.

   You can enable autoinstall for terminals either by using the system initialization table (SIT) or by using the EXEC CICS or CEMT INQUIRE|SET SYSTEM command.

   Five system initialization parameters relate to terminal autoinstall:

   AIEXIT

   specifies the name of the autoinstall program to be used. It defaults to DFHZATDX, the name of the IBM-supplied autoinstall control program.

AIQMAX
specifies the maximum number of terminals that can be queued concurrently for autoinstall. When this limit is reached, CICS requests VTAM to stop passing LOGON and BIND requests to CICS until CICS has processed one more autoinstall request.

The purpose of the limit is to protect the system from uncontrolled consumption of operating system storage by the autoinstall process, as a result of some other abnormal event. Normally, in the process of autoinstall, the principal consumer of CICS storage is the autoinstall task (CATA) itself. The amount of CICS storage consumed by the autoinstall process during normal operation can therefore be controlled by creating an appropriate TRANCLASS definition to limit the number of autoinstall tasks that can exist concurrently.

AILDELAY
specifies the time interval, expressed as hours, minutes, and seconds (hhmmss), which elapses after an autoinstall terminal logs off before its TCTTE is deleted. The default value is 0, indicating that TCTTEs are deleted at logoff time and at warm shutdown as CLSDST is issued for the autoinstall terminals still in session. Specifying an AILDELAY interval permits the TCTTE to be reused should the terminal log back on before the interval has expired.

AIRDELAY
specifies the time interval, expressed as hours, minutes, and seconds (hhmmss), which elapses after emergency restart before terminal entries are deleted if they are not in session. The default value is 700, indicating a restart delay of 7 minutes.

GRPLIST
specifies the list or lists containing the group or groups of autoinstall models created.

For information on how to specify these system initialization parameters, see the *CICS System Definition Guide*.

Three options relate to terminal autoinstall on the INQUIRE|SET AUTOINSTALL command:

CUR(*value*)
specifies the number of autoinstall logon requests that are currently being processed.

MAXREQS(*value*)
specifies the largest number of autoinstall requests that are allowed to queue at one time, in the range 0-999.

You can prevent more terminals from logging on through autoinstall by setting this value to 0. This allows autoinstalled entries for terminals currently logged on to be deleted by the autoinstall program when they log off.

PROGRAM(*pgrmid*)
specifies the name of the user program that is controlling the autoinstall process. The default is the CICS-supplied program DFHZATDX.

Autoinstall considerations

This section should help you to decide which devices to autoinstall. It covers the following subjects:

- "Automatic transaction initiation"
- "The TCT user area (TCTUA)"
- "The terminal list table (TLT)" on page 125
- "Transaction routing" on page 125
- "Autoinstall and output-only devices" on page 126

Automatic transaction initiation

If a BMS ROUTE message is sent to a terminal that has a TCT entry but is not logged on, CICS saves the message for subsequent delivery. In addition, if the TCT entry so specifies, CICS attempts to acquire the terminal and log it on for that purpose. CICS attempts to satisfy other ATI requests (EXEC CICS START or a transient data trigger level) in the same way.

The use of autoinstall for printers is severely limited because almost all transactions for printers are initiated automatically. It is possible to autoinstall printers, however, if you arrange for them to be logged on. Entering VARY NET,...,LOGON as a console command does this.

For an autoinstalled terminal, a TCT entry **may** be available even when the terminal is logged off. This happens only if the terminal has been logged on earlier in the CICS run, and depends on the TYPETERM definition, the system initialization parameters, and the SNT entry for the last user of the terminal. For details, see "Automatic sign-off, logoff, and TCTTE deletion" on page 130. If a TCT entry exists, an autoinstalled terminal can accept an ATI request just like an individually defined terminal.

If you choose autoinstall for terminals that might receive ATI requests, make use of the AUTOCONNECT attribute on the TYPETERM definition for your models. AUTOCONNECT(YES) means that the terminal is logged on to CICS automatically at an emergency restart (see Figure 17 on page 132), by CICS requesting a VTAM SIMLOGON (simulated logon). (Because this requires a TCT entry, it does not happen at cold or warm start.)

You may find that setting up a printer-owning region is the best approach, especially if you have distributed printing with many small printers.

Whether or not you autoinstall your printers, you can associate a printer and an alternate printer with a display device. The association is made when the display device is autoinstalled. Definitions for these printers need not have been installed at the time the display device is autoinstalled, but they must exist at the time of use.

The TCT user area (TCTUA)

The TCT user area is an optional extension to the TCT entry. The TCTUA is available for application use (the rest of the TCT entry belongs to CICS). It has traditionally been used for two purposes:

- To pass data from one transaction of a pseudo-conversational sequence to the next.

- To maintain user profile information and statistics during a terminal session. (This is not necessarily a VTAM session, but a period of access to a particular application, as defined by the application.)

The first use has gradually been supplanted by COMMAREA and other CICS facilities, but the second is still fairly common. An application may store statistics, such as teller totals, in the TCTUA, which is initialized by a PLTPI program at the beginning of execution and retrieved by a PLTSD program at termination (shutdown). Autoinstall does not provide the ability to save this information between logoff and logon, because the TCTUA does not exist then. In addition, the TCTUA is not available to PLTPI and PLTSD programs at system initialization and termination. A new technique must be devised to allow the initialization of TCTUA and user data when the user logs on or logs off.

As noted earlier, the autoinstall process creates the TCT entry (including the TCTUA) before the first transaction is executed at the terminal, but after the autoinstall control program has done its initial execution. Thus you cannot access the TCTUA in the autoinstall control program and so any TCTUA initialization must be done later. You could write your own 'good morning' transaction to do this, or use the first transaction of the application in question.

Furthermore, the autoinstall control program does not have access to the TCTUA at logoff either, because CICS deletes the TCT entry (including the TCTUA) before invoking this program. Therefore, if an application needs to capture statistics or other information from such a TCTUA, it must get access before CICS does this. The place to do this is in the **node error program (NEP)**, the user-written component of the terminal error processing routines, because CICS drives the NEP exit before it deletes the TCT entry.

The terminal list table (TLT)

A terminal list table is a list of terminals, defined either by four-character CICS terminal names or by three-character CICS operator identifiers. It is used principally for routing messages to multiple destinations and for giving limited operational control of a group of terminals to a supervisor. Both of these uses must be rethought in an autoinstall environment. If a TLT lists terminals that do not have TCT entries, because they are not logged on at the time the TLT is used, supervisory operations against those terminals fails. For example, you cannot put a nonexistent TCT entry into or out of service.

Similarly, message routing works differently for individually defined terminals and for autoinstalled terminals. When a message is sent to an individually defined terminal that is not logged on, the message is saved in temporary storage, and delivered when the terminal logs on. When a message is sent to a terminal that is not defined because it is an autoinstalled terminal and is not logged on, CICS gives a route fail condition, indicating that it knows nothing of that terminal. Indeed, if terminal names are generated and assigned randomly, as they may be in an autoinstall environment, the whole TLT mechanism breaks down.

Transaction routing

An autoinstall request to a local system overrides an autoinstall request for the same definition shipped from a different system.

If transaction routing can occur between two CICS systems, any terminal that can log on to both should be installed in both in the same way. Such a terminal should be:

- Autoinstalled in both systems, or
- Defined to each system in its TCT setup at initialization

Autoinstall and output-only devices

Most of the benefits of autoinstall apply more to **display devices** than to printers. Displays initiate transactions in CICS; usually, any transaction output is returned to the input terminal, so that neither CICS nor the application needs to know any other NETNAME than that of the display, which identifies itself in the process of logging on.

On the other hand, when output is directed to output-only **printers**, either CICS or the application must know what NETNAME to use, and this implies that some knowledge of the network is maintained somewhere. The primary and alternate printer names in an individually-defined TCT entry constitute this kind of information, as maintained by CICS. In the case of autoinstalled terminals, corresponding information—if it is required—must be maintained in tables or files, embedded in the application, supplied by VTAM ASLTAB and ASLENT model terminal support (MTS), or supplied dynamically by the user.

Autoinstall and VTAM

This section explains how autoinstall works with VTAM. It is intended to help you understand the processing that takes place when you use autoinstall. It consists of the following topics:
- "The process of logging on to CICS using autoinstall"
- "What happens when the user logs off" on page 129

The process of logging on to CICS using autoinstall

To help you understand the process, consider what takes place when you log on to CICS through VTAM. (See also Figure 14 on page 128 and Figure 15 on page 129.) CICS supports the model terminal support (MTS) function of VTAM 3.3 and above. Using MTS, you can define the model name, the printer (PRINTER), and the alternate printer (ALTPRINTER) for each terminal in a VTAM table. CICS captures this information as part of autoinstall processing at logon, and uses it to create a TCTTE for the terminal. If you are using MTS, you must use a version of DFHZATDX that is suitable for use on CICS Transaction Server for OS/390. See the *CICS Customization Guide* for programming information about the user-replaceable autoinstall program.

1. VTAM receives your request, determines that you want to use CICS, and passes your request to CICS.
2. CICS extracts your terminal's NETNAME name from the logon data. CICS searches the TCT for an entry with the same NETNAME.
3. If it finds such an entry, CICS issues an OPNDST to VTAM to establish a session between CICS and the terminal. This is the normal CICS logon process.
4. If it fails to find a matching entry, CICS checks the system initialization parameters that were specified in the SIT, or reset using CEMT, to check whether it can allow an autoinstall.
5. If the system initialization parameters allow an autoinstall, CICS checks the terminal data passed by VTAM, to check whether the terminal is eligible for autoinstall.
6. If the terminal is eligible, CICS examines the bind image to see if it carries sufficient information.

7. If the VTAM bind image data proves sufficient, CICS searches the autoinstall model table (AMT) and autoinstalls the terminal in one of the following ways:
   - If VTAM has supplied CICS with a valid model name, CICS passes this name to the **autoinstall control program**. (If the logon request has come to CICS through VTAM 3.3, and if you have supplied VTAM with names of model terminals, CICS can obtain the name of the model terminal from the logon data.)
   - If VTAM has not supplied CICS with a valid model name, CICS searches the AMT for suitable autoinstall models and passes these to an **autoinstall control program**, together with VTAM logon data. If VTAM has supplied CICS with an invalid model name, message DFHZC6936 results.
8. The autoinstall control program (either the CICS-supplied program or one written by you) selects one of the models and provides the rest of the information necessary to complete a TCT entry for the terminal.
9. When the autoinstall control program returns control, CICS builds a TCT entry using the autoinstall model, the data returned by the autoinstall control program, and the VTAM logon data for the terminal. CICS then adds the new entry to the TCT and issues an OPNDST to VTAM to establish a session between CICS and the terminal.
10. If the TYPETERM definition so specifies, CICS uses the QUERY function to find out about some of the features of the terminal. (These features are listed on page 333.)

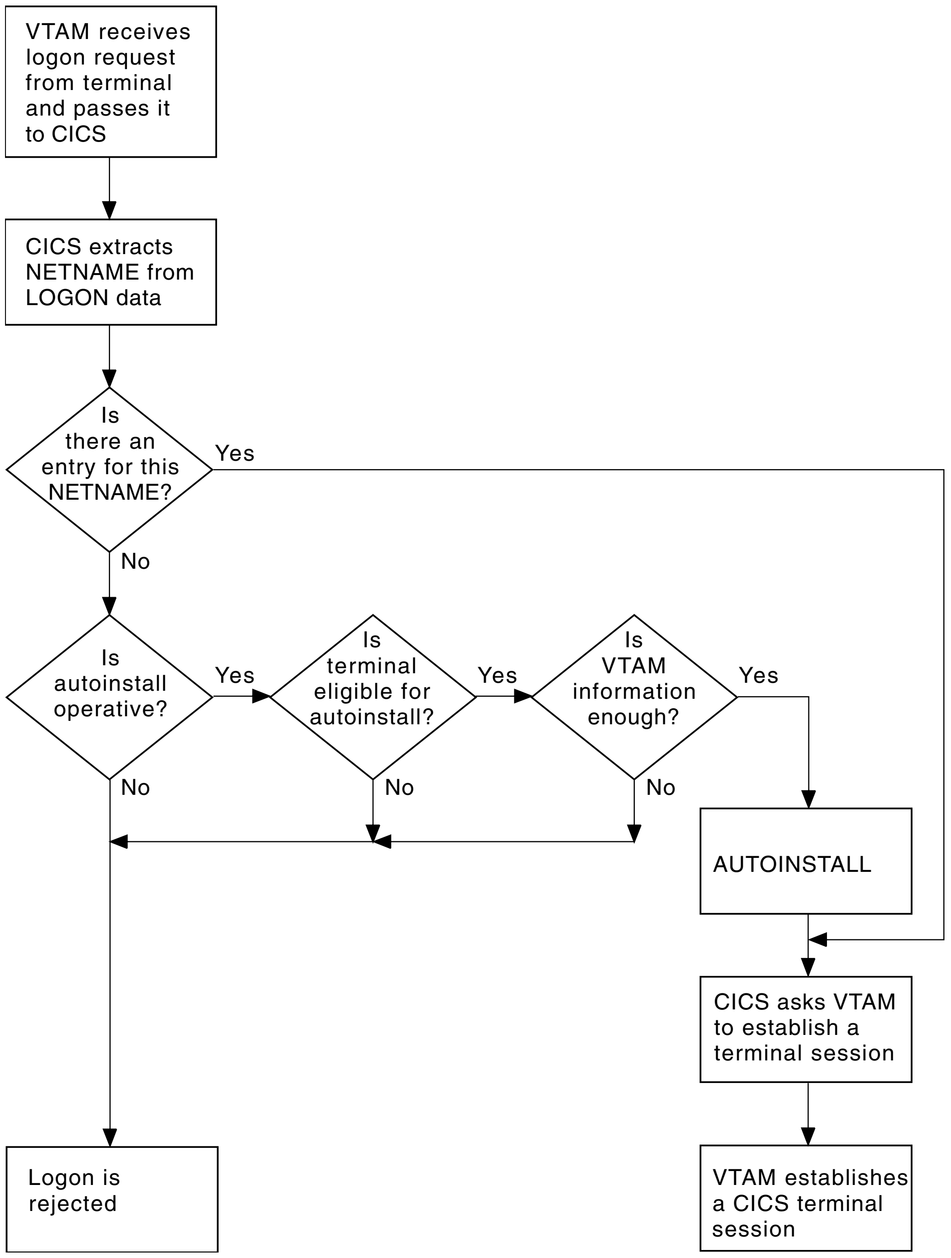

VTAM receives logon request from terminal and passes it to CICS
CICS extracts NETNAME from LOGON data
Is there an entry for this NETNAME?
Yes
No
Is autoinstall operative?
Yes
No
Is terminal eligible for autoinstall?
Yes
No
Is VTAM information enough?
Yes
No
AUTOINSTALL
CICS asks VTAM to establish a terminal session
Logon is rejected
VTAM establishes a CICS terminal session

Figure 14. The process of logging on to CICS using autoinstall. Note that the autoinstall process itself is shown as a single box. What happens inside this box is depicted in Figure 15 on page 129.

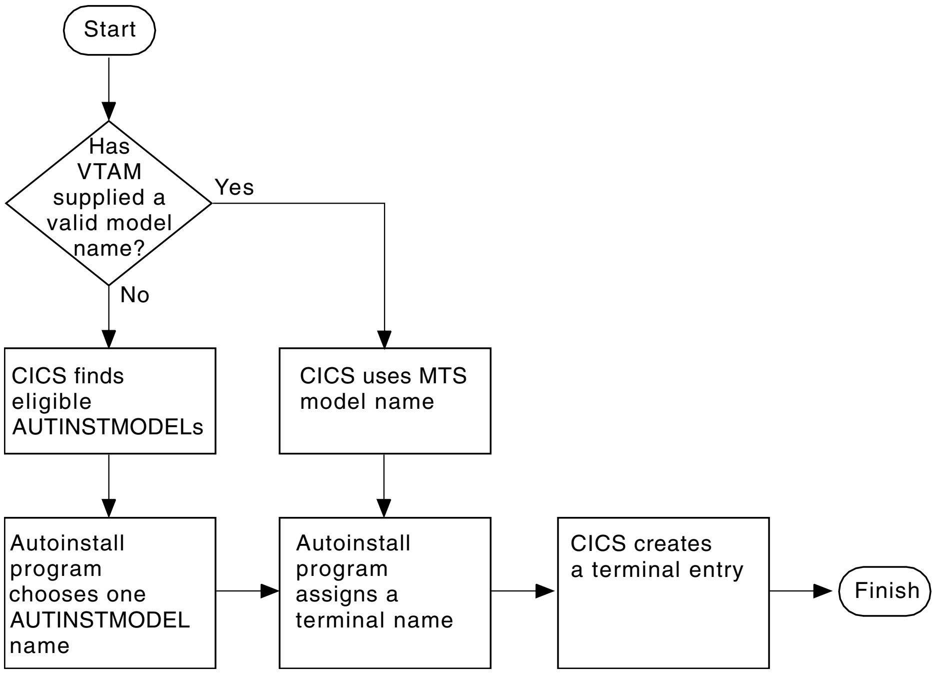

Start
Has VTAM supplied a valid model name?
Yes
No
CICS finds eligible AUTINSTMODELs
CICS uses MTS model name
Autoinstall program chooses one AUTINSTMODEL name
Autoinstall program assigns a terminal name
CICS creates a terminal entry
Finish

Figure 15. How CICS autoinstalls a terminal. Note that the overall process in which this fits is shown in Figure 14 on page 128.

What happens when the user logs off

When the terminal user finishes work and logs off:

1. CICS issues a CLSDST to ask VTAM to end the session.
2. CICS attempts to delete the TCT entry after a delay specified in the AILDELAY system initialization parameter.

   Note that the TCT entry cannot be deleted if there is a lock effective at the time. For instance, CEMT INQUIRE TERMINAL or an outstanding ATI request locks the TCT entry.
3. CICS gives control to the autoinstall control program in case it has any further processing to do; for example, to free the TERMINAL name it gave to the terminal.

If the terminal's TYPETERM definition specifies SIGNOFF(LOGOFF) **and** the RACF® segment specifies timeout, expiry of the timeout period causes the terminal to be logged off and the user to be signed off. After this automatic logoff, the delay period commences, leading to deletion of the TCT entry unless the terminal logs on again before the period ends. For details, see "Automatic sign-off, logoff, and TCTTE deletion" on page 130.

Autoinstall recovery and restart

This section explains what happens to autoinstalled terminal definitions at logoff and system restart times. It consists of:
- "What happens at CICS restart"
- "Automatic sign-off, logoff, and TCTTE deletion" on page 130

What happens at CICS restart

At an **emergency restart**, autoinstalled TCT entries are recovered unless you specify a restart delay period of zero in the AIRDELAY system initialization parameter. This means that users can log on again after an emergency restart without going through the autoinstall process. Those terminals with AUTOCONNECT(YES) specified in their TYPETERM definition are automatically logged on during the restart process, without the need for operator intervention. The recovery of autoinstalled TCT entries avoids the performance impact of many concurrent autoinstall requests following a CICS restart. A terminal user logging on after restart uses the TCT entry created for that terminal's NETNAME during the previous CICS run. It is just as if that terminal had an individual TERMINAL definition installed in the TCT.

Because this could pose a threat to security, CICS checks for operator activity after recovery. After a delay, all autoinstalled TCT entries that were recovered but are not in session again are deleted. As well as improving security, this ensures that CICS storage is not wasted by unused TCT entries. You can specify the length of the delay using the system initialization parameters.

If **persistent sessions** is in use and AIRDELAY is not equal to zero, autoinstalled TCT entries are treated exactly like other TCT entries. See the *CICS Recovery and Restart Guide* for more information about persistent sessions.

If **XRF** is in use, autoinstalled TCT entries are treated exactly like other TCT entries. That is, any TCT entry installed in the active CICS can be tracked, and a corresponding TCT entry is then installed in the alternate CICS. At an XRF takeover, for a terminal that is tracked, a TCT entry is already present in the new active CICS and the terminal is logged on. Therefore, the user does not have to go through the autoinstall process again.

At a **warm start**, TCT entries that were previously autoinstalled are lost, unless you logoff and CICS is shut down before the AILDELAY time has expired.

If a TCTTE is recovered during emergency restart, a specification of AUTOCONNECT(YES) prevents deletion of the TCTTE by AIRDELAY. If you want the TCTTE storage to be deleted in this case, specify AUTOCONNECT(NO).

Automatic sign-off, logoff, and TCTTE deletion

If a session ends through expiry of the user's TIMEOUT period, the terminal entry is deleted as described in the preceding section only if SIGNOFF(LOGOFF) is specified in the TYPETERM definition of the model. Table 7 summarizes the automatic deletion and recovery of TCTTEs for autoinstalled terminals.

Table 10 on page 132 shows how automatic sign-off, logoff, and TCTTE deletion occur if a session is timed out. Figure 17 on page 132 shows how logon and TCTTE deletion occur if, at a warm start or emergency restart, a TCTTE exists for an autoinstalled terminal. Table 11 on page 132 shows how automatic TCTTE deletion occurs if a session ends for any reason other than timeout.

Table 7. AUTOINSTALL—summary of TCTTE recovery and deletion

CICS RUNNING	**Restart delay = 0**	TCTTE entries are not cataloged and therefore cannot be recovered in a subsequent run.
	Restart delay > 0	TCTTE entries are cataloged. If they are not deleted during this run or at shutdown, they can be recovered in a subsequent emergency restart.

Table 7. AUTOINSTALL—summary of TCTTE recovery and deletion (continued)

SHUTDOWN	**Warm**	Terminals are logged off and their TCTTEs are deleted, except for those terminals which were logged off before shutdown but whose AILDELAY has not expired.
	Immediate	No TCTTEs are deleted. All can be recovered in a subsequent emergency restart.
	Abnormal termination	No TCTTEs are deleted. All can be recovered in a subsequent emergency restart.
STARTUP	**Cold**	No TCTTEs are recovered.
	Warm	No TCTTEs are recovered.
	Emergency restart	**Restart delay = 0.** No TCTTEs are recovered (but see note in Figure 17 on page 132). This session is unbound if it persists.
	Restart delay > 0. TCTTEs are recovered. For details see Figure 17 on page 132. This session is recovered if it persists.	

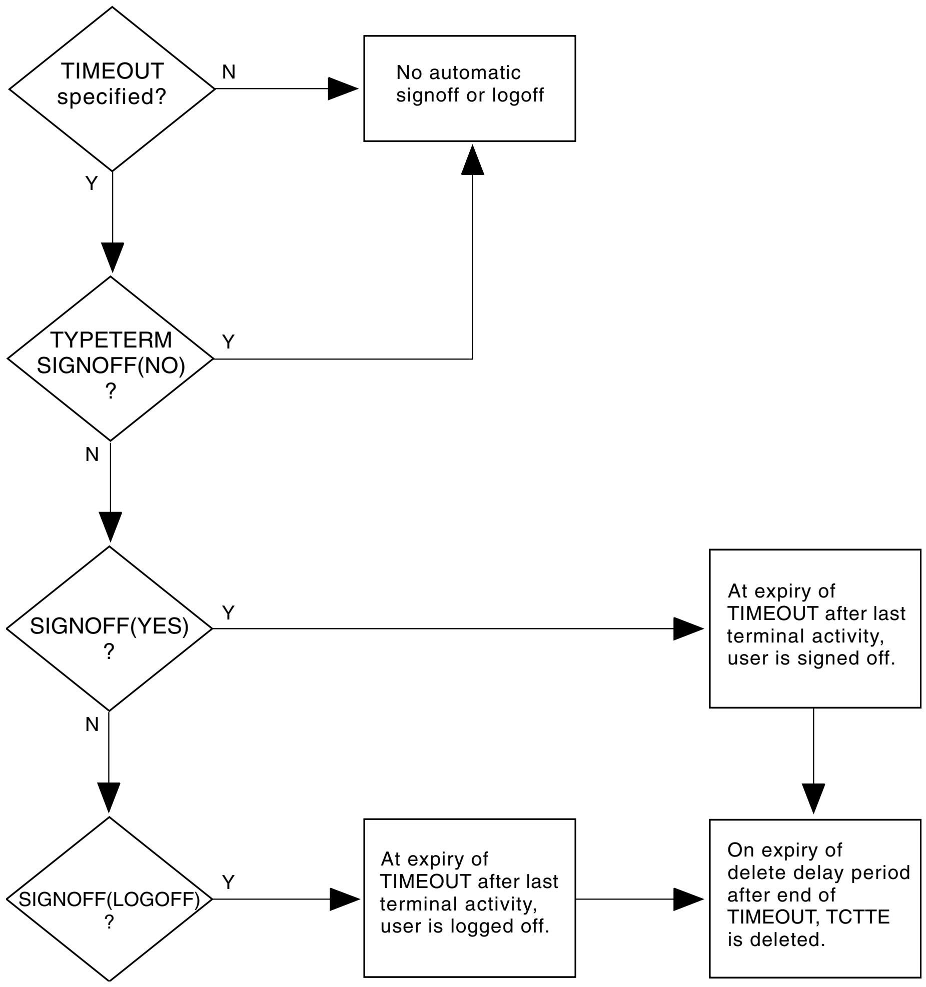

TIMEOUT specified?
N
No automatic signoff or logoff
Y
TYPETERM SIGNOFF(NO) ?
Y
N
SIGNOFF(YES) ?
Y
At expiry of TIMEOUT after last terminal activity, user is signed off.
N
SIGNOFF(LOGOFF) ?
Y
At expiry of TIMEOUT after last terminal activity, user is logged off.
On expiry of delete delay period after end of TIMEOUT, TCTTE is deleted.

Figure 16. AUTOINSTALL—automatic sign-off, logoff, and TCTTE deletion

Table 8. Chronological sequence for SIGNOFF(YES)

SIGNOFF(YES) Last activity at terminal.	<-----**TIMEOUT period**-----> User can resume session at any time in this period.	User signed off.

Table 9. Chronological sequence for SIGNOFF(LOGOFF)

SIGNOFF (LOGOFF) Last activity at terminal.	<---**TIMEOUT period**---> User can resume session at any time in this period.	User signed off, terminal logged off.	<--**Delete delay period**--> The terminal can log on during this period without repeating the autoinstall process.	TCTTE entry deleted.

Table 10. Chronological sequence for AUTOCONNECT(NO)

Emergency restart	<---------**Restart delay period**---------> For recovered autoinstalled TCTTE entries, the terminal can log on during this period without repeating the autoinstall process.	TCTTE entry deleted.

Table 11. AUTOINSTALL—automatic TCTTE deletion after non-automatic logoff

Non-automatic LOGOFF VTAM informs CICS of a session outage, terminal logs off, or transaction disconnects terminal	<---------**Delete delay period**---------> The terminal can log on during this period without repeating the autoinstall process.	TCTTE entry deleted.

Note

Successive CICS runs are assumed to use the same restart delay period.
Note that, if a run with restart delay > 0 is followed by an emergency restart with restart delay = 0, undeleted TCTTEs are recovered. If the recovered TCTTE specifies AUTOCONNECT(YES), the associated terminal is logged on; otherwise the TCTTE is deleted after startup.

Emergency Restart

If restart delay period = 0, there is no TCTTE recovery during emergency restart.

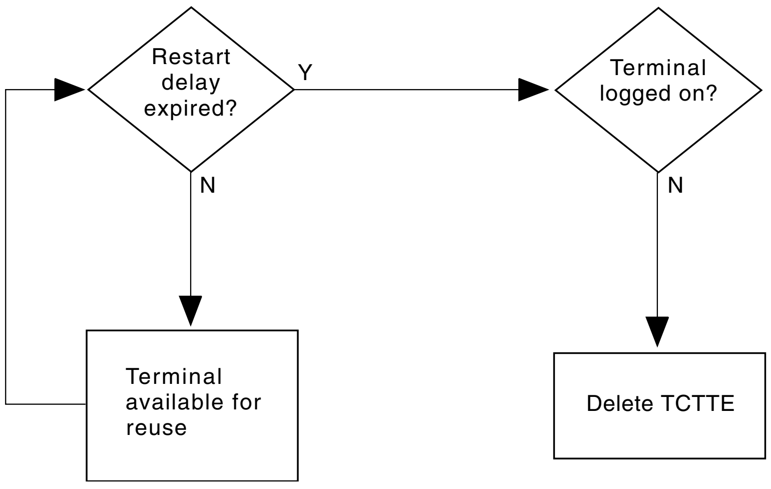


Figure 17. AUTOINSTALL—automatic logoff and TCTTE deletion after an emergency restart

The autoinstall control program

The autoinstall control program is invoked by CICS every time there is a valid request for a TCT entry to be autoinstalled, and every time an autoinstalled TCT entry is deleted.

For programming information about the autoinstall control program, see *CICS Customization Guide* This section is a summary of that information, and consists of:
- "Autoinstall with model terminal support"
- "Autoinstall program functions"

Autoinstall with model terminal support

If you are using VTAM 3.3 or later, you can define, in a VTAM table, the model name as well as other information for each terminal. During logon, VTAM sends this information to CICS, which may use it to create a TCTTE for the terminal. Therefore you do not need to specify routines to select a model terminal, optional printer, and alternate printer, but your autoinstall control program still needs to provide a TERMINAL name. However, if you are already using an autoinstall control program that selects a model terminal, optional printer, and alternate printer, CICS can continue to use this program.

Autoinstall program functions

If you are using a level of VTAM earlier than VTAM 3.3, the NETNAME and other VTAM data about the terminal are not sufficient to build a TCT entry for the terminal. This program must create a CICS identifier for the TERMINAL, and must choose a suitable model from among those passed to it by CICS.

IBM supplies a program (DFHZATDX) that performs the basic functions, but it may not perform all the functions that you require. For example, you may have your own conventions for TERMINAL names and their relationship to NETNAMEs. (Note that TERMINAL names are up to four characters long, and NETNAMEs are up to eight characters long, so that it is often not possible to derive one from the other.)

In addition, you could code your program to perform other functions associated with terminal logon and logoff. For example:
- Security checking
- Providing associated printer names
- Monitoring the number of terminals currently logged on through autoinstall

The autoinstall control program runs in a transaction environment, rather than as a CICS exit. This means that you can read files and issue other CICS commands, to determine the TERMINAL name, or the associated PRINTER and ALTPRINTER names. The TCT entry, however, does not exist at either of the times that this program is invoked, because at logon it has not yet been created, and at logoff it has already been deleted. The program therefore runs in transaction-without-terminal mode.

You can write the autoinstall control program in any CICS language: assembler language, C/370, COBOL II, or PL/I. The CICS-supplied autoinstall program is available in all these languages; the assembler version is used by default. If you decide to write your own program, you can use one of the CICS-supplied programs as a pattern. Note that for COBOL II and C/370, you need extra program definitions in the CSD file. See the *CICS System Definition Guide* for information about defining the CSD file.

You specify the name of the program you want to use in the AIEXIT system initialization parameter.

When you test your autoinstall control program, the CADL transient data destination records each installation and each deletion of TCT entries. Message DFHZC6987 is useful for indicating which model came closest to being chosen, when a null list of models is passed to the autoinstall control program.

Note: You can have only one autoinstall control program active at one time for terminals and connections. The active program is specified on the AIEXIT system initialization parameter. The DFHZATDY program described in “Chapter 10. Autoinstall for APPC connections” on page 141 provides the same function for terminal autoinstall as DFHZATDX, but also provides function to autoinstall APPC connections initiated by BIND requests. Therefore, if you want to autoinstall APPC connections as well as terminals, use a customized version of DFHZATDY rather than DFHZATDX.

For programming information on implementing the CICS-supplied autoinstall control program, or designing and writing your own program, see the *CICS Customization Guide*.

Chapter 9. Autoinstall for MVS consoles

This chapter explains how to implement autoinstall for MVS consoles. It contains the following information:
- "Overview"
- "Implementing autoinstall for consoles" on page 136
- "The autoinstall control program" on page 138

If you have not used autoinstall for any resources before, read "Chapter 7. Overview of autoinstall" on page 119.

Overview

Commands issued at an MVS console (or in a job stream) can be directed to a CICS region running as a started task, or job, using the MVS MODIFY command. Before CICS can accept the MVS command, it needs an entry in the terminal control table for the console issuing the command, which CICS terminal control can use to display a response. The following discussion describes how CICS handles commands (transaction invocations) it receives from an MVS operator console.

Using pre-installed console definitions

When MVS receives your request, it identifies the CICS region from the task or job name, and passes your request to CICS.

CICS extracts the console's name from the MODIFY data, and searches the terminal control table (TCT) for a CONSNAME entry that matches the console name. If CICS cannot find a matching name, but the console has a migration ID (range 100 - 250) or a non-extended id (range 0-99), CICS tries to find a matching CONSOLE id. If CICS finds a matching entry, it starts the transaction specified on the MODIFY command, and the transaction can send the results to the console using the termid of the console's entry in the terminal control table.

Using autoinstalled console definitions

If CICS fails to find a matching entry, it checks the autoinstall status for consoles to determine whether it can perform an autoinstall for the console.

If autoinstall for consoles is active, CICS searches the autoinstall model table (AMT) and initiates the autoinstall process for the console. CICS either:
- Passes a list of autoinstall model definitions to the autoinstall control program, together with information about the console, or
- Automatically uses the first console model definition it finds, or a console model with the same name as the console, and autoinstalls the console using a CICS-generated termid, without calling your autoinstall control program.

Which of these options CICS takes is determined by the autoinstall status for consoles. The autoinstall status for consoles is either set at startup by the AICONS system initialization parameter, or dynamically by a CEMT (or EXEC CICS) SET AUTOINSTALL CONSOLES command.

The terminal autoinstall control program

You use the same autoinstall control program for console autoinstall as for VTAM terminals and APPC connections, specifying the name of the control program on the AIEXIT system initialization parameter.

If the autoinstall control program is invoked (either the CICS-supplied program or your own) it selects one of the models and provides the rest of the information necessary to complete a TCT terminal entry for the console. When the autoinstall control program returns control, CICS builds a terminal control table terminal entry (TCTTE) for the console using the autoinstall model, the termid, and other data returned by the autoinstall control program, and MVS console data. CICS then adds the new entry to the TCT and starts the transaction specified on the MODIFY command.

Preset security for autoinstalled consoles

If the model terminal specifies USERID(*FIRST) or USERID(*EVERY), CICS uses the user ID passed by MVS on the MODIFY command to sign on the console, in effect using the MVS-passed user ID as the preset userid for the new console.

Automatic deletion of autoinstalled consoles

CICS automatically deletes autoinstalled consoles if they are unused for a specified delay period (the default is 60 minutes). As part of the install function, the autoinstall control program can set a ‘delete-delay’ value for the console. The delete-delay period is the length of time (in minutes) that an autoinstalled console can remain installed without being used before CICS deletes it. Setting this value to 0 inhibits automatic deletion. Autoinstalled consoles are not recorded on the catalog and not recovered at restart. Note that a console is deleted even if there is a currently signed-on user.

Implementing autoinstall for consoles

CICS autoinstall support for consoles is not provided automatically—there are some tasks to be completed to enable the support. Basically, you need to specify that you want the support, and ensure that the required model resource definitions are installed. This is because the autoinstall models supplied in DFHLIST do not contain models suitable for console autoinstall. The IBM-supplied group DFHTERMC contains an autoinstall model definition for a console, but this is not included in DFHLIST. Also, an optional requirement is the support of an autoinstall control program

The following steps describe how to enable autoinstall for consoles:

1. Define a console terminal definition that:
   - Specifies AUTINSTMODEL(YES), or AUTINSTMODEL(ONLY)
   - Specifies either CONSNAME(name) or CONSOLE(number).
   - References a TYPETERM that specifies DEVICE(CONSOLE)

   You can use the model console definition defined in the group DFHTERMC, which is added to your CSD by the INITIALIZE and UPGRADE commands.

   Note: When a model terminal definition is used by the console autoinstall function, CICS ignores the console name or console id specified on the model definition.
2. Install the model console definition either by adding its group to a group list and perform a cold start, or use the CEDA INSTALL command.

3. If you decide that you want the autoinstall control program to be invoked for console autoinstall, modify your autoinstall control program to handle console install and delete requests. To reactivate your modified program, either restart CICS or use the CEMT, or EXEC CICS, SET PROGRAM(...) NEWCOPY command.

   To ensure CICS invokes your autoinstall control program, specify system initialization parameter AICONS=YES, or use the CEMT, or EXEC CICS, SET AUTOINSTALL CONSOLES(PROGAUTO) command to specify console autoinstall dynamically.
4. If you decide to let CICS autoinstall consoles without invoking your autoinstall control program, specify system initialization parameter AICONS=AUTO, or use the CEMT, or EXEC CICS, SET AUTOINSTALL CONSOLES(FULLAUTO) command to specify console autoinstall dynamically. With the AUTO option, CICS allocates the termid automatically.
5. As in the case of VTAM terminal autoinstall, ensure that the necessary autoinstall programs and transactions are installed. These are your autoinstall control program, the transactions CATA and CATD, and the programs DFHZATD and DFHZATA. The CICS-supplied autoinstall control program, DFHZATDX or DFHZATDY, accepts a request from any console, provided an autoinstall model for a console is found in the AMT. Use the model definition supplied in group DFHTERMC, or alternatively create your own autoinstall console models (see “The autoinstall control program” on page 138)

If, when your CICS system is in production, you want to restrict the consoles that are allowed to be autoinstalled, control this in the autoinstall control program. There are other reasons why you might write your own autoinstall control program, such as security requirements or varying the default delete-delay period. See the *CICS Customization Guide* for information about including support for consoles in your autoinstall console program. You may have to change the way you use console names and terminal names, or you may have to make special arrangements in the autoinstall control program to allow you to continue to use the names in the way that you do.

Defining model TERMINAL definitions for consoles

Whenever CICS receives a command from an unknown console, and autoinstall for consoles is active, CICS searches the AMT for autoinstall models that describe a console. These models must specify either a CONSOLE id or a CONSNAME, and reference a TYPETERM definition that specifies DEVICE(CONSOLE). The console id and console name in a model definition are dummy values in the case of autoinstall, and are ignored by CICS, which passes a list of AUTINSTNAME attribute values to the autoinstall control program, so that it can choose one of them.

Specifying automatic preset security

One attribute that your autoinstall control program may require in a model definition is a USERID that provides the correct preset security. Selecting a model that has preset security defined ensures that the operator’s security authority is predetermined. This avoids an operator having to logon to CICS, using the CESN signon transaction, in order to issue a MODIFY command. Two special operands of the USERID parameter provide for an automatic signon of consoles:

- USERID(*EVERY) means that CICS is to use the userid passed on the MVS MODIFY command every time a MODIFY command is received. The console is signed on using the MVS userid as the preset userid for the console being autoinstalled. The console remains signed on with this userid until the console is deleted or another MODIFY command is received with another userid. If a

MODIFY command is received without a userid, CICS signs on the default CICS userid until a MODIFY command is received that has a valid userid. For non-console terminals, or if security is not enabled, this value is ignored.
- USERID(*FIRST) means that CICS is to use the userid passed on the first MVS MODIFY command that requires the console to be autoinstalled. The console is signed on with the MVS userid as the preset userid. The console remains signed on with this userid until the console is deleted. If a MODIFY command is received without a userid, CICS signs on the default CICS userid. For non-console terminals, or if security is not enabled, this value is ignored.

The autoinstall control program

The console name and other MVS data about the console are not sufficient to build a terminal control table terminal entry (TCTTE) for the console. The autoinstall control program must create a CICS terminal identifier (termid) for the console, and choose a suitable model from the list of models passed by CICS in the communications area.

IBM supplies two assembler versions of an autoinstall control program (DFHZATDX and DFHZATDY in SDFHLOAD) that perform the basic functions, but it these may not perform all the functions that you require. For example, you may have your own conventions for terminal names and their relationship to console names. Terminal names are up to four characters long, and console names are up to eight characters long, hence it is often not possible to derive one from the other.

In addition to providing the termid, you can code your program to perform other functions associated with console definition. For example, your program could:
- Perform security checks
- Monitor the number of autoinstalled consoles currently logged on.

The autoinstall control program runs in a transaction environment as a user-replaceable module, rather than as a CICS exit. This means that you can read files, and issue other CICS commands, to help you determine the terminal name. The TCTTE entry for the console, however, does not exist at either of the times that the program is invoked, because (1) for the install function it has not yet been created, and (2) for the delete function it has already been deleted. The program therefore runs in transaction-without-terminal mode.

You can write an autoinstall control program in any CICS supported language: assembler language, C/370, COBOL II, or PL/I. The CICS-supplied autoinstall program source is available in all four languages. The assembler version, which is used by default, is also shipped in executable form in SDFHLOAD. If you decide to write your own program, you can use one of the IBM-supplied programs as a pattern.

You specify the name of your autoinstall control program on the AIEXIT system initialization parameter. CICS supports only one autoinstall control program at a time, and therefore your program must handle console and terminal autoinstall requests if support for both is required.

When you test your autoinstall control program, you will find that CICS writes install and delete messages to the transient data destination, CADL, each time CICS installs and deletes a TCT entry. If there are no autoinstall models installed for consoles, CICS does not call the autoinstall control program, and fails the autoinstall request.

For information on implementing the CICS-supplied autoinstall control program, or designing and writing your own program, see the *CICS Customization Guide*.

Chapter 10. Autoinstall for APPC connections

This chapter describes how to implement autoinstall for APPC parallel-session or single-session connections when a BIND is received from a primary. It consists of the following:

If you have not used autoinstall for any resources before, read "Chapter 7. Overview of autoinstall" on page 119 before going any further.

If autoinstall is enabled, and an APPC BIND request is received for an APPC service manager (SNASVCMG) session that does not have a matching CICS CONNECTION definition, or a BIND is received for a single session, a new connection is created and installed automatically.

Getting started with APPC connection autoinstall

1. **Decide whether to use autoinstall for connections**.

   You are most likely to benefit from autoinstall for connections if you have large numbers of workstations all with the same characteristics. The main **benefits** of using autoinstall for connections are that cold, warm, and emergency restarts are faster, you have fewer CSD file definitions to manage, and less storage is taken up by unused definitions.

   However, some possible restrictions are discussed in "Security" on page 142 and "Connection autoinstall and recovery and restart" on page 144.
2. **Decide which sessions to autoinstall**.

   You can use autoinstall for CICS-to-CICS connections, but it is intended primarily for workstations.
3. **Create your model connection definitions**.

   A model definition provides CICS with one definition that can be used for all connections with the same properties. See "Model definitions for connection autoinstall" on page 143.
4. **Design and write an autoinstall control program**.

   The autoinstall control program provides CICS with the extra information it needs to complete an autoinstall request, such as the autoinstall model name.

   For programming information about the autoinstall control program, see the *CICS Customization Guide* and "The autoinstall control program for connection autoinstall" on page 143.
5. **Enable autoinstall for connections**.

   Autoinstall for connections is enabled when you enable autoinstall for terminals; see page 122 for information about the system initialization parameters and the CEMT and EXEC CICS INQUIRE and SET options used.

   If terminal autoinstall has been enabled but you want to prevent autoinstall for connections, set the model connection out of service by using the CEMT or EXEC CICS SET CONNECTION(connection-name) OUTSERVICE command. (See the *CICS Supplied Transactions* manual for information about the CEMT commands, and the *CICS System Programming Reference* manual for programming information about the EXEC CICS commands.) When the model connection is out of service, the autoinstall control program cannot access it and copy it, so the autoinstall function for connections is effectively disabled.

Considerations for connection autoinstall

This section should help you in deciding whether to use autoinstall for connections. It covers the following subjects:
- "Security"
- "Model terminal support (MTS)"
- "Deletion of autoinstalled connections"

Security

Before using autoinstall for connections, consider whether the security considerations are suitable for your purposes.

The autoinstalled connection inherits the security attributes specified in the model. The security attributes from the CONNECTION definition are:
- SECURITYNAME
- ATTACHSEC
- BINDSECURITY

If you are using compatibility mode to share a CSD file with an earlier release of CICS, the BINDPASSWORD attribute is also inherited. See "CONNECTION" on page 157 for information on these attributes.

From the SESSIONS definition, the autoinstalled connection inherits the preset security attribute USERID. See page 263 for a description of this attribute. If you are attempting to autoinstall an attachsec identify connection from a CICS system prior to CICS/ESA 4.1 or you are attempting to autoinstall an attachsec verify connection from a non-EBCDIC based system, then you should refer to *'Attach Time Security and the USEDFLTUSER option'* in the *CICS RACF Security Guide*.

Model terminal support (MTS)

MTS is not supported for connection autoinstall. This means that you must code the routines yourself to select the model definition name; you cannot get VTAM to do it for you. MTS is described on page 133.

Deletion of autoinstalled connections

Unlike autoinstalled terminal definitions, autoinstalled connection definitions are **not** deleted when they are no longer in use. This means that they continue to occupy storage.

The rules for cataloging and deletion of autoinstalled APPC connections have changed in CICS Transaction Server for OS/390. All autoinstalled connections are now cataloged, depending on the AIRDELAY system initialization parameter. If AIRDELAY=0, synclevel 1 connections are not cataloged.

The rules for deletion are:
- Autoinstalled synclevel 1 connections are deleted when they are released.
- If CICS is **not** registered as a generic resource, autoinstalled synclevel 2 connections are not deleted.

- If CICS **is** registered as a generic resource, autoinstalled synclevel 2 and limited resources connections are deleted when the affinity is ended.

Model definitions for connection autoinstall

Model definitions for connection autoinstall are different from those for terminal autoinstall in that they do not have to be defined explicitly as models. Any installed connection definition can be used as a "template" for an autoinstalled connection.

For performance reasons, use an installed connection definition that is not otherwise in use. The definition is locked while CICS copies it and, if you have a very large number of sessions autoinstalling, the delay may be noticeable.

You can set the model connection definition out of service by using the CEMT or EXEC CICS SET CONNECTION OUTSERVICE command. This effectively disables autoinstall for connections, because the autoinstall control program cannot access and copy the model definition.

For CICS-to-CICS connection autoinstall, the MAXIMUM attribute in the SESSIONS definition of the model connection should be large enough to accommodate the largest device using the model.

When creating model connections, you must ensure that the usergroup modenames specified in the session's MODENAME field match the usergroup modenames in the connection to be autoinstalled.

The autoinstall control program for connection autoinstall

The autoinstall control program is invoked at installation for:
- APPC parallel-session connections initiated by a BIND
- APPC single-session connections initiated by a BIND

The autoinstall control program provides CICS with any extra information it needs to complete an autoinstall request. For APPC parallel sessions, the control program provides a SYSID for the new definition.

When an APPC BIND request is received by CICS, CICS receives the partner's VTAM NETNAME and passes it to the autoinstall control program. The control program uses the information contained in the partner's NETNAME and in the VTAM BIND to select the most appropriate model on which to base a new connection. In order to return the name of the most suitable model to CICS, the control program must know the NETNAME or SYSID of every model.

CICS supplies a sample control program, DFHZATDY, for connection autoinstall. You can use DFHZATDY unchanged if both of the following conditions are met:
- Your model connections are called CCPS, CBPS, or CBSS
- You use the last four characters of the NETNAME as the SYSID or terminal name

If these conditions are not met, you have to change DFHZATDY to suit your installation. Its source is supplied in CICSTS13.CICS.SDFHSAMP. DFHZATDY is defined as follows:

```
DEFINE  PROGRAM(DFHZATDY)   GROUP(DFHAI62)   LANGUAGE(ASSEMBLER)
        RELOAD(NO)   RESIDENT(NO)   STATUS(ENABLED)   CEDF(NO)
        DATALOCATION(ANY)   EXECKEY(CICS)
```

The definitions for the supplied model connections and sessions are:

```
DEFINE CONNECTION(CBPS)  GROUP(DFHAI62)  NETNAME(TMPLATE1)
       ACCESSMETHOD(VTAM)  PROTOCOL(APPC)  SINGLESESS(NO)
DEFINE SESSION(CBPS)  GROUP(DFHAI62)  CONNECTION(CBPS)
       MODENAME(LU62PS)  PROTOCOL(APPC)  MAXIMUM(10,5)

DEFINE CONNECTION(CBSS)  GROUP(DFHAI62)  NETNAME(TMPLATE2)
       ACCESSMETHOD(VTAM)  PROTOCOL(APPC)  SINGLESESS(YES)
DEFINE SESSION(CBSS)  GROUP(DFHAI62)  CONNECTION(CBSS)
       MODENAME(LU62SS)  PROTOCOL(APPC)  MAXIMUM(1,0)

DEFINE CONNECTION(CCPS)  GROUP(DFHAI62)  NETNAME(TMPLATE3)
       ACCESSMETHOD(VTAM)  PROTOCOL(APPC)  SINGLESESS(NO)
DEFINE SESSION(CCPS)  GROUP(DFHAI62)  CONNECTION(CCPS)
       MODENAME(LU62PS)  PROTOCOL(APPC)  MAXIMUM(10,5)
```

If you want to use these definitions, you must add group DFHAI62 to your group list.

Note: Do not try to use the terminal autoinstall exit DFHZATDX to autoinstall connections; any sessions installed by DFHZATDX are terminated and message DFHZC6921 is issued.

For programming information on customizing the autoinstall control program, see the *CICS Customization Guide*.

Connection autoinstall and recovery and restart

This section explains how autoinstalled connections are handled in different restart situations.

Persistent sessions support: Because autoinstalled connections are not cataloged, they cannot benefit from persistent sessions support. This means that when a session does persist, any autoinstalled connections relating to it are unbound.

XRF support: Autoinstalled connections are tracked to XRF.

Autoinstalled connections are recovered only at a cold start of CICS; they are not recovered at warm and emergency restarts.

Unit-of-recovery descriptors (URDs) for autoinstalled connections are recovered after cold, warm, and emergency restarts, as long as the SYSIDs of the connections are consistent.

The SYSIDs of the autoinstalled connections must be consistent if you are using recoverable resources.

Chapter 11. Autoinstall for other resources

This chapter describes how to implement autoinstall for programs, mapsets, partitionsets, and journals. It consists of the following:

If you have not used autoinstall for any resources before, read “Chapter 7. Overview of autoinstall” on page 119 before going any further.

If autoinstall for programs is active, and an implicit or explicit load request is issued for a previously undefined program, mapset, or partitionset, CICS dynamically creates a definition, and installs it and catalogs it as appropriate. An implicit or explicit load occurs when:

- CICS starts a transaction
- An application program issues one of the following commands:
  EXEC CICS LINK
  EXEC CICS XCTL
  EXEC CICS LOAD
  EXEC CICS ENABLE (for global user exit, or task-related user exit, program)
- When, after an EXEC CICS HANDLE ABEND PROGRAM(...), a condition is raised and CICS transfers control to the named program.
- CICS calls a user-replaceable module for the first time
- An application program issues one of the following commands:
  EXEC CICS RECEIVE or SEND MAP
  EXEC CICS SEND PARTNSET
  EXEC CICS RECEIVE PARTN

Getting started with program autoinstall

1. **Decide whether your programs are eligible for autoinstall**.

   The only program that cannot be autoinstalled is the program autoinstall control program.

   All other programs can be autoinstalled, including:
   - All other user-replaceable programs
   - Global user exits (GLUEs) and task-related user exits (TRUEs)
   - PLT programs

   User-replaceable programs and PLT programs are autoinstalled on first reference. GLUEs and TRUEs are autoinstalled when enabled.
2. **Decide whether to use autoinstall for programs**.

   Using autoinstall for programs can save time spent on defining individual programs, mapsets, and partitionsets. Savings can also be made on storage, because the definitions of autoinstalled resources do not occupy space until they are referenced.

Use of autoinstall for programs can reduce the number of definitions to be installed on a COLD start, thereby reducing the time taken.

3. **Decide which programs to autoinstall**.

   Depending on your programs, you can choose to use a mixture of RDO and autoinstall. A suggested way to manage program autoinstall is to continue to use RDO for existing program definitions and for installing groups containing related programs. Use autoinstall as you develop and install new applications, and in test environments, where you might otherwise install large numbers of programs at CICS startup.

4. **Enable autoinstall for programs**.

   You can enable autoinstall for programs either by using the system initialization table (SIT) or by using the EXEC CICS or CEMT INQUIRE|SET SYSTEM command.

   Three system initialization parameters relate to program autoinstall:

   PGAICTLG
      specifies whether autoinstalled program definitions should be cataloged. See "Cataloging for program autoinstall" on page 147.

   PGAIPGM
      specifies whether the program autoinstall function is active or inactive.

   PGAIEXIT
      specifies the name of the program autoinstall exit.

   For information on how to specify these system initialization parameters, see the *CICS System Definition Guide*.

   Three operands relate to program autoinstall on the INQUIRE|SET SYSTEM command:

   PROGAUTOCTLG
      specifies whether autoinstalled program definitions are to be cataloged. See "Cataloging for program autoinstall" on page 147.

   PROGAUTOEXIT
      specifies the name of the program autoinstall exit.

   PROGAUTOINST
      specifies whether the program autoinstall function is active or inactive.

   For programming information on how to specify EXEC CICS commands, see the *CICS System Programming Reference* manual, and for information on how to use CEMT, see the *CICS Supplied Transactions* manual.

5. **Create your model program definitions**.

   A model definition provides CICS with one definition that can be used for all programs with the same properties. See "Model definitions for program autoinstall" on page 147

6. **Design and write an autoinstall control program**.

   The autoinstall control program provides CICS with the extra information it needs to complete an autoinstall request, such as the autoinstall model name. You can write your autoinstall program in any language supported by CICS, with full access to the CICS application programming interface. See "The autoinstall control program for program autoinstall" on page 147.

Cataloging for program autoinstall

You can specify whether an autoinstalled program definition is cataloged or not (that is, whether the definition is retained over a warm or emergency start) by using either the PGAICTLG system initialization parameter or the PROGAUTOCTLG option on the CEMT INQUIRE|SET SYSTEM command. The values you can specify are:

ALL
Autoinstalled program definitions are written to the global catalog at the time of the autoinstall, and following any subsequent modification.

MODIFY
Autoinstalled program definitions are cataloged only if the program definition is modified by a SET PROGRAM command subsequent to the autoinstall.

NONE
Autoinstalled program definitions are not cataloged. This gives a faster CICS restart (warm and emergency) compared with the MODIFY or ALL options, because CICS does not reinstall definitions from the global catalog. Definitions are autoinstalled on first reference.

The effects of specifying cataloging for program autoinstall apply mainly to recovery and restart. See "Program autoinstall and recovery and restart" on page 148.

Model definitions for program autoinstall

Model definitions for program autoinstall are different from those for terminal autoinstall in that they do not have to be defined explicitly as models. Any installed program, mapset, or partitionset definition can be used as a "template" for program autoinstall.

If you do not want to use your own definitions, you can use the CICS-supplied model definitions. These are:
- DFHPGAPG for programs
- DFHPGAMP for mapsets
- DFHPGAPT for partitionsets

They are listed in "Appendix B. CICS-supplied resource definitions, groups, and lists" on page 503.

Note: Although the DFHPGAPG definition does not specify a program language, the CICS autoinstall routine detects the program language from the program being loaded.

The autoinstall control program for program autoinstall

You specify the name of the control program you want to use in the PGAIEXIT system initialization parameter, or use the CEMT or EXEC CICS INQUIRE|SET SYSTEM command.

For detailed programming information about the autoinstall control program for programs, see the *CICS Customization Guide*. This section is a summary of that information.

When it is invoked

For **programs**, the autoinstall control program is invoked when:

- Any of these commands references a previously undefined program:
  - EXEC CICS LINK
  - EXEC CICS XCTL
  - EXEC CICS LOAD
- The program is the first program in a transaction.
- An EXEC CICS ENABLE is issued for a GLUE or a TRUE.
- An abend occurs after an EXEC CICS HANDLE ABEND PROGRAM command is issued and CICS invokes the named program.
- CICS calls a user-replaceable module.

For **mapsets**, the autoinstall control program is invoked when an EXEC CICS SEND MAP or EXEC CICS RECEIVE MAP refers to a previously undefined mapset.

For **partitionsets**, the autoinstall control program is invoked when an EXEC CICS SEND PARTNSET or EXEC CICS RECEIVE PARTN command refers to a previously undefined partitionset.

Sample programs

The following sample programs are supplied by CICS:

- DFHPGADX—assembler program for program autoinstall exit
- DFHPGAHX—C program for program autoinstall exit
- DFHPGALX—PL/I program for program autoinstall exit
- DFHPGAOX—COBOL definition for program autoinstall exit

The source for these programs is supplied in the CICSTS13.CICS.SDFHSAMP sample library, but only the assembler version is supplied in executable form, in the CICSTS13.CICS.SDFHLOAD load library.

Program autoinstall functions

The program autoinstall facility uses model definitions, together with a user-replaceable module (URM), to create explicit definitions for programs, mapsets, and partitionsets that need to be autoinstalled. CICS calls the URM with a parameter list that gives the name of the appropriate model definition. On return from the URM (depending on the return code), CICS creates a resource definition from information in the model and from parameters returned by the URM.

Note: CICS does not call the URM for any CICS programs, mapsets, or partitionsets (that is, any objects beginning with the letters DFH).

For programming information on how to code a program autoinstall URM, see the *CICS Customization Guide*.

Program autoinstall and recovery and restart

There is a difference in performance between warm and emergency restarts using program autoinstall without cataloging, and warm and emergency restarts using autoinstall with cataloging. (See the *CICS Recovery and Restart Guide* for information on cataloging.)

If you are using autoinstall **with** cataloging, restart times are similar to those of restarting a CICS region that is not using program autoinstall. This is because, in both cases, resource definitions are reinstalled from the catalog during the restart. The definitions after the restart are those that existed before the system was terminated.

If you are using autoinstall **without** cataloging, CICS restart times are improved because CICS does not install definitions from the CICS global catalog. Instead, definitions are autoinstalled as required whenever programs, mapsets, and partitionsets are referenced following the restart.

Journal autoinstall

CICS writes journal data to journals or logs on log streams managed by the MVS system logger, or to SMF. You must define JOURNALMODELs so that the connection between the journal name, or identifier, and the log stream, or SMF log, can be made. You can use RDO, DFHCSDUP, or an EXEC CICS CREATE JOURNALMODEL command to define a journalmodel.

CICS resolves the link between the log stream and the journal request by using user-defined JOURNALMODELs or, in the situation where an appropriate JOURNALMODEL does not exist, by using default names created by resolving symbolic names. See "JOURNALMODEL" on page 210 for more information about this.

Journals are not user-definable resources.

Part 5. RDO resource types and their attributes

This part describes the resource types that you can define, manage, and install using RDO commands. The resource types are in alphabetical order.

For each resource type, this part shows the panels that you get when you type a **DEFINE** command correctly, followed by a resource type, and with no resulting error messages. For example:

```
DEFINE FILE(FILEB) GROUP(MYGROUP)
```

Everything but the information area has been removed, and panels have been joined together when there is more than one.

- Attributes that you **must** fill in are highlighted
- Capital letters indicate the minimum command line abbreviation for each attribute and option

  Note: CICS may add new resource types and attributes across different releases. For similarly named fields such as TRANSACTION and TRANCLASS, the minimum abbreviation may have to change from release to release. For example, at CICS/ESA 3.3 the minimum abbreviation for TRANSACTION was TR, whereas in this release it is now TRANS, to differentiate between that and TRANC for TRANCLASS.

  RDO commands are validated online by the CEDA/B/C transactions, and confirmation for such ambiguities is required before the transaction can continue. This is not possible for input to DFHCSDUP however. To avoid batch operations failing because the minimum abbreviation has altered across CICS releases, it is recommended that the full resource type and attribute names are used when coding input to DFHCSDUP. While CICS cannot guarantee upward compatibility of the abbreviations, it always provides such compatibility for the full names of the fields.
- Dots indicate the maximum length of each name.
- The choice of possible values is shown on the right. (The basic default values are shown following the ==>, but note that some default values may differ, depending on other values that you type in.)
- Where you see a colon (:) instead of the ==>, between an attribute and its value, it means that you cannot change the value. This can be for several reasons:
  – You do not have authority to change the values (for example, if you are a CEDC user).
  – You are using the VIEW command (explained later in this tutorial).
  – The length of the attribute is such that the whole of the attribute does not fit in the current screen view. In order to update the attribute value you must first scroll forwards (PF8 or PF11) or backwards (PF7 or PF10) until the whole attribute is seen.
  – The attribute is the resource name or the group name (in this example, MAPSET name NEW1 and GROUP name AAA1).
  – The attribute is obsolete for the current release of CICS (for example, the RSL attribute). To change the value of obsolete attributes, you must use the **compatibility mode** option, as described in the PF keys below.

Resource types and their attributes

Following the panel for each resource type, there is an alphabetical list of the attributes, with an explanation of each attribute and the possible values that it can take.

If you are familiar with resource definition macro operands, you may also want to refer to “Appendix C. Attribute cross-reference tables” on page 531.

Chapter 12. CICS RDO resources

This chapter introduces the CICS resources that you can define by RDO. For a description of the resources you can define by macro, see "Chapter 13. Resource definition with macros" on page 367.

Table 12. CICS RDO resources

Resource	Description	Page reference
CONNECTION	A CONNECTION is the definition of a remote system that your CICS system communicates with using intersystem communication (ISC) or multiregion operation (MRO).	157
DB2CONN	A DB2CONN is the definition of the global attributes of the connection to be established between CICS and DB2 as well as the attributes of pool threads and command threads to be used with the connection.	171
DB2ENTRY	A DB2ENTRY defines resources to be used by a specific CICS transaction, or a group of transactions, when using the CICS DB2 interface.	179
DB2TRAN	A DB2TRAN resource definition defines additional transactions to be associated with a particular DB2Entry. DB2Tran allows a DB2Entry to have an unrestricted number of transactions associated with it,	183
DOCTEMPLATE	A DOCTEMPLATE resource definition defines a document template. Document templates allow you to perform variable substitution on documents in a manner similar to that done by BMS for 3270 screens.	185
ENQMODEL	An ENQMODEL resource definition defines a resource for which ENQ|DEQ is to have a sysplex-wide scope	187
FILE	CICS file control uses the FILE definition for information on the physical and operational characteristics of a file, such as types of operations allowed on the file, recovery attributes, and operations to be journaled. Remote files (that is, files on another system) are also defined using this RDO resource type.	190
JOURNALMODEL	This resource definition provides the connection between a CICS journal name (or identifier) and the associated log streams managed by the MVS system logger, or between the journal name and the SMF log.	210
LSRPOOL	The local shared resources (LSR) pool is a reserve of data buffers, strings, and Hiperspace™ buffers (see the *CICS Performance Guide* for more information on Hiperspace) that VSAM uses when processing access requests for certain files.	217

Table 12. CICS RDO resources (continued)

Resource	Description	Page reference
MAPSET	Each interactive application using a display device can use specific screen layouts, or maps. Each map can be used by multiple invocations of the same program, or by different programs. You use either basic mapping support (BMS) or Screen Definition Facility (SDF) to create maps. Every map must belong to a mapset.	223
PARTITIONSET	The screen areas of some display devices (for example, the 8775 Display Terminal, and the IBM 3290 Information Panel) can be divided into **partitions**, each of which can be treated as a separate display. Different programs or transactions can write to or receive input from different partitions.	226
PARTNER	You use the PARTNER definition to enable CICS application programs to communicate via APPC protocols to a partner application program running on a remote logical unit. This interaction is called a **conversation**. The PARTNER definition also facilitates the use of the call to the interface with the communications element of the System Application Architecture.	229
PROCESSTYPE	Using the CICS business transaction services (BTS) API, you can define and execute complex business applications called *processes*. A PROCESSTYPE resource definition defines a BTS process-type. It names the CICS file which relates to the physical VSAM data set (repository) on which details of all processes of this type (and their activity instances) are to be stored. A PROCESSTYPE resource definition	232
PROFILE	The PROFILE definitions is used to specify options that control the interactions between transactions and terminals or logical units. It is a means of standardizing the use of, for example, screen size and printer compatibility. Each TRANSACTION definition names the PROFILE to be used.	235
PROGRAM	You use the PROGRAM definition to describe the control information for a program that is stored in the program library and used to process a transaction.	242
REQUESTMODEL	A REQUESTMODEL resource definition provides the connection between an Internet Inter-ORB Protocol (IIOP) inbound request and the identifier of the CICS transaction that is to be initiated.	251
SESSIONS	Before two systems can communicate using ISC or MRO, they must be logically linked through one or more sessions. The nature of the link determines how they can communicate. You specify the link in the SESSIONS definition.	253

Table 12. CICS RDO resources (continued)

Resource	Description	Page reference
TCPIPSERVICE	Use this resource to define which TCP/IP services are to use CICS internal sockets support. The internal CICS services that can be defined are IIOP and the CICS Web interface.	264
TDQUEUE	CICS transient data uses definitions for its resources to outline the physical and operational characteristics of the associated data sets. Other attributes, for example, associated transactions, terminals, and remote systems can also be included in the definition.	268 for RDO for transient data, and 376 for the DFHDCT macro
TERMINAL	CICS needs a definition for each terminal with which it communicates. A terminal's unique properties are in its TERMINAL definition. Properties that it has in common with other terminals (usually static) are in the TYPETERM definition.	285
TRANSACTION	A CICS application consists of one or more programs written to perform a specific function. A particular invocation of such an application is a **transaction**. In the TRANSACTION definition you specify options related to functions provided by CICS itself, such as transaction priority, security key, and the length of the transaction work area (TWA).	314
TRANCLASS	By putting your transactions into transaction classes (TRANCLASSes), you can control how CICS dispatches tasks. For example, you can separate transactions into those that are heavy resource users and those that are of lesser importance, such as the "Good morning" broadcast messages. You can then use the attributes on the TRANCLASS definition to control the number of active and new tasks allowed from each transaction class.	311
TSMODEL	A TSMODEL resource definition allows you to specify a Temporary Storage queue name prefix, and associate attributes with that name. You can also map names directly to a shared TS pool (without the need for a shared sysid).	330
TYPETERM	A TYPETERM is a partial terminal definition that identifies a set of common terminal properties or attributes. Every TERMINAL definition must specify a TYPETERM. TYPETERMS make it easier to define your terminals if you have many terminals of the same kind.	333

How the resource types relate to each other

Every resource is defined with a set of **attributes**. The attributes are the properties of the resource, telling CICS, for example, whether a file can be updated, what security level should be given to a transaction, or what remote systems CICS can communicate with. Some of the attributes relate resources to each other. For example:

- Every TRANSACTION definition must refer to either a PROGRAM definition or a REMOTESYSTEM definition

- Every SESSIONS definition must refer to a CONNECTION definition
- Every TERMINAL definition must refer to a TYPETERM definition

CONNECTION

A CONNECTION is the definition of a remote system that your CICS system communicates with using ISC or MRO. When the CONNECTION definition is installed in the CICS system, this information is stored in the terminal control table (TCT) as a system entry (TCTSE).

When you define a CONNECTION, you give enough information to identify the system and specify its basic attributes. You put details in the SESSIONS definition about the sessions you use to communicate with the system. CICS uses the CONNECTION name to identify the other system when the definition has been installed. For other CICS systems connected via MRO, this name is typically the same as that specified in the other CICS system as the SYSIDNT system initialization parameter. For other systems connected via ISC, this name is typically based on an acronym that describes the location of or the organization that owns the system (for example, USA1 or IBMC).

The REMOTESYSTEM name on a TRANSACTION definition, or on a TERMINAL definition, refers to a CONNECTION definition through its CONNECTION name. These attributes are used for transaction routing.

The REMOTESYSTEM name on a PROGRAM definition refers to a CONNECTION definition through its CONNECTION name. This attribute is used for distributed program link.

The CONNECTION definition does **not** name associated SESSIONS.

Installing CONNECTION definitions

To install new CONNECTION definitions, put them in a group of their own which does not contain CONNECTION definitions that have already been installed, then use CEDA INSTALL to install the whole group. You cannot install single CONNECTION definitions.

To modify and reinstall existing MRO CONNECTION definitions, or if you want new and existing MRO CONNECTION definitions to be in the same group, you must close down all interregion communication (IRC) and open it again, before you can use the definition. If you do not close IRC and open it again, you get message DFHIR3788 when you try to bring up the region with the new connection.

Use the following procedure:

1. Close IRC down:
   ```
   CEMT SET IRC CLOSED
   ```
2. Install the resource definitions:
   ```
   CEDA INSTALL GROUP(groupname)
   ```
3. When you have successfully installed the group containing the definitions, open IRC again:
   ```
   CEMT SET IRC OPEN
   ```

Definitions for various links and sessions

Here is some guidance on creating resource definitions for:

- MRO links and sessions
- APPC (LUTYPE6.2) links and parallel sessions
- APPC (LUTYPE6.2) single session terminals
- LUTYPE6.1 links and sessions
- LUTYPE6.1 CICS-IMS links and sessions
- INDIRECT connections

MRO links and sessions: You define an MRO link using one CONNECTION definition, and its associated parallel sessions using one SESSIONS definition.

ACCESSMETHOD
On the CONNECTION definition, specify this as IRC (for interregion communication), or XM (for cross-memory services). IRC is used to open and close the links.

PROTOCOL
On the SESSIONS definition, specify LU61 as the PROTOCOL. On the CONNECTION definition, leave the PROTOCOL value blank.

SENDPFX

SENDCOUNT

RECEIVEPFX

RECEIVECOUNT
In one SESSIONS definition, you specify a number of send sessions and a number of receive sessions. The values that you specify in these attributes are used to determine the names of the TCT entries created when the definition is installed. (See the section on installing "MRO links and sessions" on page 253.)

USEDFLTUSER
If you have a connection or APPC terminal definition to an earlier CICS release or CICS on another platform then you may need to code this as USEDFLTUSER(YES), See *'Attach Time Security and the USEDFLTUSER option'* in the *CICS RACF Security Guide*.

APPC links and parallel sessions: For APPC, the sessions are grouped into modesets. You define each modeset with a SESSIONS definition, so you have as many SESSIONS definitions as you require modesets. You define the link as a CONNECTION definition.

The following attributes are significant:

ACCESSMETHOD
On the CONNECTION definition, specify this as VTAM.

MAXIMUM
Use this to control the number of sessions in the modeset.

MODENAME
On the SESSIONS definition for each modeset, name the modeset with the MODENAME. This is the name by which the modeset is known to CICS when the definition is installed in the active system.

PROTOCOL
On both the CONNECTION and SESSIONS definitions, specify APPC as the protocol.

USEDFLTUSER
If you have a connection or APPC terminal definition to an earlier CICS release or CICS on another platform then you may need to code this as USEDFLTUSER(YES), See *'Attach Time Security and the USEDFLTUSER option'* in the *CICS RACF Security Guide*.

APPC (LUTYPE6.2) single session terminal: You can define an APPC terminal as a CONNECTION-SESSIONS pair or as a TERMINAL-TYPETERM pair. The TERMINAL-TYPETERM method is described on page 290.

If you want to use the CONNECTION-SESSIONS method, the following attributes are significant:

ACCESSMETHOD
On the CONNECTION definition, specify this as VTAM.

MAXIMUM
For a single session terminal, specifying 1,0 or 1,1 has the same effect. (For further information, see page 257.)

MODENAME
On the SESSIONS definition, specify the MODENAME. This is the name that CICS uses to identify the session when the definition is installed in the active system.

PROTOCOL
On both the CONNECTION and SESSIONS definitions, specify APPC as the protocol.

SINGLESESS
YES indicates that the CONNECTION definition is for a single session terminal.

| **USEDFLTUSER**
| If you have a connection or APPC terminal definition to an earlier CICS release
| or CICS on another platform then you may need to code this as
| USEDFLTUSER(YES), See *'Attach Time Security and the USEDFLTUSER option'*
| in the *CICS RACF Security Guide*.

LUTYPE6.1 links and sessions: LUTYPE6.1 links and sessions can be defined in one of two ways:

- In one CONNECTION and one SESSIONS definition
- In one CONNECTION and a number of SESSIONS definitions: one for each session needed

If your sessions are all to have **identical** attributes, define each link in one CONNECTION definition and all its associated sessions in one SESSIONS definition.

ACCESSMETHOD
On the CONNECTION definition, specify this as VTAM.

PROTOCOL
On the SESSIONS definition and on the CONNECTION definition, specify this as LU61.

RECEIVECOUNT

RECEIVEPFX

SENDCOUNT

SENDPFX
These attributes are used as for MRO links and sessions.

If your sessions are to have **different** attributes from each other, you must create a separate SESSIONS definition for each one. With the exception of NETNAMEQ, this method is the same as that for CICS-IMS sessions, described below.

Note: For CICS-CICS ISC links and sessions, you are recommended to use APPC rather than LUTYPE6.1.

LUTYPE6.1 CICS-IMS links and sessions: IMS needs each session to be defined in a separate SESSIONS definition, because each session must have a different NETNAMEQ.

You define the link as a CONNECTION definition, and create a number of SESSIONS definitions: one for each SEND session and one for each RECEIVE session.

ACCESSMETHOD
On the CONNECTION definition, specify this as VTAM.

NETNAMEQ
This is the name that the remote IMS™ system uses to identify the session.

PROTOCOL
On both the CONNECTION and SESSIONS definitions, specify LU61 as the protocol.

SESSNAME
This is the name that CICS uses to identify the session when the definition is installed in the active system.

RECEIVECOUNT

SENDCOUNT
Use these attributes to specify whether a session is a SEND session or a RECEIVE session.

A RECEIVE session is one in which the local CICS is the primary and is the contention loser. It is specified by defining RECEIVECOUNT(1) and leaving SENDCOUNT to default to blank. (You do not need to specify a SENDPFX or a RECEIVEPFX.)

A SEND session is one in which the local CICS is the secondary and is the contention winner. Specify it by defining SENDCOUNT(1) and leaving RECEIVECOUNT to default to blank.

INDIRECT connections: An INDIRECT connection is a remote system for which you have not defined a direct link with the local system. Instead, the two systems communicate with each other by way of one or more intermediate systems. You can use this method for transaction routing. The remote system, indirectly connected, is always the terminal-owning region; the local system is always the application-owning region or an intermediate region on the transaction routing path.

Note: In releases of CICS before 4.1, indirect connections were required for transaction routing across intermediate systems. They were needed on the application-owning region and on intermediate regions, to:
- Identify the NETNAME of the terminal-owning region
- Identify the next system in the path to the terminal-owning region

In CICS Transaction Server for OS/390, indirect connections are required only if you use non-VTAM terminals for transaction routing across intermediate systems. Optionally, you can use them with VTAM terminals, where several transaction routing paths are possible, to identify the preferred path to the terminal-owning region. For information about why you might want to define indirect connections, and about the resource definitions required for transaction routing, see the *CICS Intercommunication Guide*.

Defining indirect connections: In the local system, you must have ordinary CONNECTION and SESSIONS definitions for the intermediate systems to which you are directly connected. The ACCESSMETHOD should be IRC or XM with PROTOCOL(LU61), or VTAM with PROTOCOL(APPC).

For the INDIRECT connection (also known as an indirect link or an indirect system) you need, in the local system, a CONNECTION definition only. You do not need a SESSIONS definition: the sessions that are used are those of the intermediate system. The following attributes of the CONNECTION definition are significant:

ACCESSMETHOD
Specify this as INDIRECT.

INDSYS
Specify the CONNECTION definition for the MRO or APPC link that is the start of a path to the terminal-owning system.

NETNAME
Specify the APPLID of the terminal-owning system.

Before you start creating definitions for intercommunication resources, see the *CICS Intercommunication Guide* for further guidance. There you can find many useful examples of the attributes you must specify for different types of links and sessions.

Defining a CONNECTION

```
 Connection   ==>
 Group        ==>
 DEscription  ==>
CONNECTION IDENTIFIERS
 Netname      ==>
 INDsys       ==>
REMOTE ATTRIBUTES
 REMOTESYSTem ==>
 REMOTEName   ==>
 REMOTESYSNet ==>
CONNECTION PROPERTIES
 ACcessmethod ==> Vtam              Vtam | INdirect | IRc | Xm
 PRotocol     ==>                   Appc | Lu61 | Exci
 Conntype     ==>                   Generic | Specific
 SInglesess   ==> No                No | Yes
 DAtastream   ==> User              User | 3270 | SCs | STrfield | Lms
 RECordformat ==> U                 U | Vb
 Queuelimit   ==> No                No | 0-9999
 Maxqtime     ==> No                No | 0-9999
OPERATIONAL PROPERTIES
 AUtoconnect  ==> No                No | Yes | All
 INService    ==> Yes               Yes | No
SECURITY
 SEcurityname ==>
 ATtachsec    ==> Local             Local | Identify | Verify | Persistent
                                    | Mixidpe
 BINDPassword   :                   PASSWORD NOT SPECIFIED
 BINDSecurity ==> No                No | Yes
 Usedfltuser  ==> No                No | Yes
RECOVERY
 PSrecovery   ==>                   Sysdefault | None
 Xlnaction    ==> Keep              Keep | Force
```

Figure 18. The DEFINE panel for CONNECTION

Options

ACCESSMETHOD({VTAM|INDIRECT|IRC|XM})
specifies the access method to be used for this connection.

VTAM
Communication between the local CICS region and the system defined by this CONNECTION is through VTAM. You can use VTAM intersystem communication (ISC) for systems that are in different MVS images or in different address spaces in the same MVS image.

INDIRECT
Communication between the local CICS system and the system defined by this CONNECTION is through the system named in the INDSYS operand.

IRC Communication between the local CICS region and the region defined by this CONNECTION is through the interregion communication (IRC) program DFHIRP, using the SVC (as opposed to cross-memory (XM)) mode of DFHIRP.

Note: This use of the term IRC is more specific than its general use. You can use IRC for multiregion operation (MRO) for regions that are in the same MVS image or in different MVS images within a sysplex.

XM MRO communication between the local CICS region and the region defined by its CONNECTION definition uses MVS cross-memory services. Initial connection is through the interregion communication (IRC) program DFHIRP, using the cross-memory (XM) (as opposed to the SVC) mode of DFHIRP. You can use XM for multiregion operation for regions that are in the same MVS image, or in different MVS images within a sysplex.

Note: The CICS type 3 SVC is still required with XM because DFHIRP is used when the link is opened. For further information about SVCs, see the *CICS Transaction Server for OS/390 Installation Guide*.

MVS cross-memory services are used only if the ACCESSMETHOD of the other end of the link is also defined as XM.

If the MRO partners reside in different MVS images within a sysplex, and the CONNECTION specifies IRC or XM, CICS automatically uses XCF as the access method, and ignores the IRC or XM specification.

Note: You cannot define XCF explicitly; if you want to use XCF, you must specify IRC or XM. See the *CICS Intercommunication Guide* for more information about XCF.

ATTACHSEC({LOCAL|IDENTIFY|VERIFY|PERSISTENT| MIXIDPE})
specifies the level of attach-time user security required for the connection.

IDENTIFY
Incoming attach requests must specify a user identifier. Enter IDENTIFY when the connecting system has a security manager; for example, if it is another CICS system.

LOCAL
The authority of the user is taken to be that of the link itself, and you

rely on link security alone to protect your resource. If the PROTOCOL attribute on the CONNECTION definition is LU6.1, you must specify LOCAL.

MIXIDPE
Incoming attach requests may be using either or both IDENTIFY or PERSISTENT security types. The security type actually used depends on the incoming attach request.

PERSISTENT
Incoming attach requests must specify a user identifier and a user password on the first attach request. Subsequent attach requests require only the user identifier. This should be used only between a PWS (programmable workstation, for example, an IBM Personal System/2) and CICS.

VERIFY
Incoming attach requests must specify a user identifier and a user password. Enter VERIFY when the connecting system has no security manager and hence cannot be trusted.

Do not specify VERIFY for CICS-to-CICS communication, because CICS does not send passwords.

AUTOCONNECT({NO | YES | ALL})
For systems using ACCESSMETHOD(VTAM), you specify with AUTOCONNECT(YES) or (ALL) that sessions are to be established (that is, BIND is to be performed). Such sessions are set up during CICS initialization, or when you use the CEMT or EXEC CICS SET VTAM OPEN command to start communication with VTAM. If the connection cannot be made at these times because the remote system is unavailable, you must subsequently acquire the link by using the CEMT or EXEC CICS SET CONNECTION(sysid) INSERVICE ACQUIRED command, unless the remote system becomes available in the meantime and itself initiates communications.

For APPC connections with SINGLESESS(NO) specified, CICS tries to bind, on system start-up, the LU services manager sessions in mode group SNASVCMG.

For CONNECTIONs with SINGLESESS(YES) specified, the AUTOCONNECT operand is ignored. Use the AUTOCONNECT operand of the SESSIONS definition instead.

ALL On this definition, ALL is equivalent to YES, but you can specify ALL to be consistent with the SESSIONS definition.

AUTOCONNECT(ALL) should not be specified for connections to other CICS systems, because this can cause a bind-race.

NO CICS does not attempt to bind sessions when the connection is established.

YES CICS attempts to bind only contention-winning sessions when the connection is established.

The LU6.1 option is not applicable on an LU6.1 CONNECTION. For LU6.1 connections, specify AUTOCONNECT(YES) on the SESSIONS if you want the connection to be established at initialization or CEDA install. Specify AUTOCONNECT(NO) on the SESSIONS if you do not want the connection to be established at initialization or CEDA install.

BINDPASSWORD(*password*) (APPC only)
The BINDPASSWORD attribute is not valid for CICS Transaction Server for OS/390. See "Appendix A. Obsolete attributes retained for compatibility" on page 497 for information.

BINDSECURITY({NO|YES}) (APPC only)
specifies whether an ESM is being used for bind-time security.

No No external bind-time security is required.

Yes If security is active and the XAPPC system initialization parameter is set to YES, CICS attempts to extract the session key from RACF in order to perform bind-time security. If no RACF profile is available, the bind fails.

CONNECTION(*name*)
specifies the name of this CONNECTION definition. The name can be up to four characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

This is the name specified as REMOTESYSTEM on FILE, TERMINAL,
TRANSACTION, and PROGRAM definitions. You should not have a
TERMINAL definition and a CONNECTION definition with the same name.

CONNTYPE({SPECIFIC|GENERIC})
For external CICS interface connections, this specifies the nature of the connection.

GENERIC
The connection is for communication from a non-CICS client program to the CICS system, and is generic. A generic connection is an MRO link with a number of sessions to be shared by multiple EXCI users. For a generic connection you cannot specify the NETNAME attribute.

SPECIFIC
The connection is for communication from a non-CICS client program to the CICS region, and is specific. A specific connection is an MRO link with one or more sessions dedicated to a single user in a client program. For a specific connection, NETNAME is mandatory.

DATASTREAM({USER|3270|SCS|STRFIELD|LMS})
specifies the type of data stream.

LMS The data stream is a Logical Message Services (LMS) data stream consisting of FMH4s and FMH8s as defined in the LUTYPE6.1 architecture.

SCS The data stream is an SCS data stream as defined in the LUTYPE6.1 architecture.

STRFIELD
The data stream is a structured field data stream as defined in the LUTYPE6.1 architecture.

USER Let DATASTREAM default to USER if the data stream is user-defined. If you are communicating between multiple CICS systems, always let DATASTREAM default to USER.

3270 The data stream is a 3270 data stream as defined in the type 6.1 logical unit (LUTYPE6.1) architecture.

DESCRIPTION(*text*)
describes the resource you are defining. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see "How the CSD file is organized—groups and lists" on page 8.

The GROUP name can be up to eight characters in length. The characters allowed are A-Z 0-9 $ @ and #. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

INDSYS(*name*)
specifies the name of an intermediate system that is used to relay communications between this system and the remote system. The name can be up to four characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

You may name an intermediate system only if you specify ACCESSMETHOD(INDIRECT). The name must be the name of a CONNECTION definition.

INSERVICE({YES|NO})
specifies the status of the connection that is being defined.

NO The connection can neither receive messages nor transmit input.

YES Transactions may be initiated and messages may automatically be sent across the connection.

MAXQTIME({No|*number*})
specifies a time control on the wait time for queued allocate requests waiting for free sessions on a connection that appears to be unresponsive. The maximum queue time is used only if a queue limit is specified on QUEUELIMIT, and then the time limit is applied only when the queue length has reached the QUEUELIMIT value.

NO CICS maintains the queue of allocate requests that are waiting for a free session. No time limit is set for the length of time that requests can remain queued (though the DTIMOUT mechanisms can apply to individual requests). In this case, a value of X'FFFF' is passed on the XZIQUE parameter list (in field UEPEMXQT).

number
The approximate upper limit on the time that allocate requests can be queued for a connection that appears to be unresponsive. The number represents seconds in the range 0 through 9999.

CICS uses the maximum queue time attribute to control a queue of allocate requests waiting as follows. When the number of queued allocate requests reaches the queue limit (QUEUELIMIT), and a new allocate request is received for the connection, if the rate of processing for the queue indicates that, on average, the new allocate takes more than the maximum queue time, the queue is purged, and message DFHZC2300 is issued.

No further queuing takes place until the connection has successfully freed a session. At this point, CICS issues DFHZC2301 and resumes normal queuing.

You can also control the queuing of allocate requests through an XZIQUE global user exit program. This allows you to use statistics provided by CICS, which report the state of the link. You can use these statistics, in combination with the queue limit and maximum queue time values you specify, to make more specialized decisions about queues.

The MAXQTIME value is passed to an XZIQUE global user exit program on the XZIQUE parameter list, if the exit is enabled. See the *CICS Customization Guide* for programming information about writing an XZIQUE global user exit program.

You can also specify the NOQUEUE|NOSUSPEND option on the ALLOCATE command to prevent an explicit request being queued. See the *CICS Application Programming Reference* manual for programming information about these API options.

NETNAME(*name*)
specifies the network name that identifies the remote system. The name can be up to eight characters in length. The name follows assembler language rules. It must start with an alphabetic character. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

The NETNAME is the APPLID of the remote system or region, *unless* you are defining an APPC or LUTYPE6.1 link to a VTAM generic resource group. If you are defining an LUTYPE6.1 link to a generic resource, NETNAME must specify the generic resource name, not the APPLID of one of the group members. If you are defining an APPC link to a generic resource, NETNAME can specify either the group's generic resource name or the APPLID (member name) of one of the group members. However, if you specify a member name, and this CICS is not itself a member of a CICS Transaction Server for OS/390 generic resource, the connection must always be acquired by this CICS.

For VTAM, the APPLID is the label of the remote ACF/VTAM® VBUILD TYPE=APPL statement.

If you do not supply a NETNAME, the CONNECTION name is used by default.

There are some rules about duplicate NETNAMEs. You **cannot** have:
- Two or more APPC links with the same NETNAME
- An APPC link and an LUTYPE6.1 link with the same NETNAME
- Two or more IRC connections with the same NETNAME
- | Two or more remote APPC connections with the same NETNAME.
- | A remote APPC connection with the same NETNAME as any other connection or local terminal.

You **can** have:

- An IRC connection and an LUTYPE6.1 connection with the same NETNAME
- An IRC connection and an APPC connection with the same NETNAME
- Two or more LUTYPE6.1 connections with the same NETNAME
- Any connection with the same NETNAME as a remote terminal.

You cannot use an ALIAS when naming an APPC or LUTYPE6.1 link. CICS requires you to use the real NETNAME of the connecting partner. Any attempt to use an alias causes the BIND to be rejected.

PROTOCOL({APPC | LU61 | EXCI | blank})
specifies the type of protocol that is to be used for the link.

APPC (LUTYPE6.2 protocol)
Advanced program-to-program communication, or APPC protocol. This is the default value for ACCESSMETHOD(VTAM). Specify this for CICS-CICS ISC.

blank MRO between CICS regions. You must leave the PROTOCOL blank for MRO, and on the SESSIONS definition you must specify LU6.1 as the PROTOCOL.

EXCI The external CICS interface. Specify this to indicate that this connection is for use by a non-CICS client program using the external CICS interface.

LU61 LUTYPE6.1 protocol. Specify this for CICS-CICS ISC or CICS-IMS ISC, but not for MRO.

PSRECOVERY({SYSDEFAULT | NONE})
In a CICS region running with persistent sessions support, this specifies whether, and how, LU6.2 sessions are recovered on system restart within the persistent session delay interval.

NONE
All sessions are unbound as out-of-service with no CNOS recovery.

SYSDEFAULT
If a failed CICS system is restarted within the persistent session delay interval, the following actions occur:

- User modegroups are recovered to the SESSIONS RECOVOPTION value.
- The SNASVCMG modegroup is recovered.
- The connection is returned in ACQUIRED state and the last negotiated CNOS state is returned

QUEUELIMIT({No | *number*})
specifies the maximum number of allocate requests that CICS is to queue while waiting for free sessions. You can specify either NO, or a number:

NO There is no limit set to the number of allocate requests that CICS can queue while waiting for a free session. In this case, a value of X'FFFF' is passed on the XZIQUE parameter list (in field UEPQUELM).

number
The maximum number of allocate requests, in the range 0 through 9999, that CICS can queue on the connection while waiting for a free session. When the number of queued allocate requests reaches this limit, CICS rejects subsequent allocate requests until the queue drops below the limit.

This queue limit is passed to an XZIQUE global user exit program on the XZIQUE parameter list if the exit is enabled.

You can also control the queuing of allocate requests through the MAXQTIME attribute, and through an XZIQUE global user exit program. See the MAXQTIME attribute for more information about controlling queues.

Note:

1. The queueing of requests applies to Allocate requests only. On an LU62 session, Function Shipping requests will be failed with a SYSIDERR immediately if the MAXIMUM() value is reached. This option should be used with caution on a link where both Allocate and Function Shipping requests occur.
2. BIND re-negotiation is not triggered, even if there are unused secondary sessions. Unless the CEMT SET MODE command is used to force re-negotiation, the queuelimit will come into play as soon as all the primary sessions are in use.

RECORDFORMAT({U | VB})
specifies the type of SNA chain.

U Let RECORDFORMAT default to U if the SNA chain is a single, unblocked stream of data. You can have private block algorithms within the SNA chain.

Let RECORDFORMAT default to U if you are communicating between multiple CICS systems.

VB The SNA chain is formatted according to the VLVB standard as defined in the LUTYPE6.1 architecture.

REMOTENAME(*name*)
specifies the name by which the APPC connection for transaction routing is known in the system or region that owns the connection. The name can be up to four characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command. The remote system or region can be an APPC device (see "APPC devices for transaction routing" on page 292).

REMOTESYSNET(*name*)
specifies the network name (APPLID) of the system that owns the connection. The name can be up to eight characters in length. It follows assembler language rules, and must start with an alphabetic character. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

Use REMOTESYSNET when transaction routing to remote APPC systems or devices, and there is no direct link between the region in which this definition is installed and the system that owns the connection to the remote device. You do not need to specify REMOTESYSNET if:

- You are defining a local connection (that is, REMOTESYSTEM is not specified, or specifies the sysid of the local system).
- REMOTESYSTEM names a direct link to the system that owns the connection. However, there is one special case: if the connection-owning region is a member of a VTAM generic resources group and the direct link to it is an APPC connection, you may need to specify REMOTESYSNET. REMOTESYSNET is needed in this case if the NETNAME specified on the

CONNECTION definition for the direct link is the generic resource name of the connection-owning region (not the applid).

REMOTESYSTEM(*name***)**

specifies the name that identifies the intercommunication link to the system that owns the connection. The name can be up to four characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

This is the CONNECTION name on the CONNECTION definition for the intercommunication link.

REMOTESYSTEM is used for transaction routing to remote APPC systems or devices. If it is not specified, or if it is specified as the sysid of the local system, this connection is local to this system. If the name is that of another system, the connection is remote. You can therefore use the same definition for the connection in both the local system and a remote system.

If there are intermediate systems between this CICS and the region that owns the (connection to the) device, REMOTESYSTEM should specify the first link in the path to the device-owning region. If there is more than one possible path, it should specify the first link in the preferred path.

SECURITYNAME(*name***)**

For APPC and LU6.1 links only, this is the security name of the remote system.

In a CICS system with security initialized (SEC=YES or MIGRATE), the security name is used to establish the authority of the remote system.

Note: If USERID is specified in the session definition (DEFINE SESSIONS command) associated with the connection definition, it overrides the userid specified in the SECURITYNAME attribute, and is used for link security.

The security name (or USERID on the sessions definition) must be a valid RACF userid on your system. Access to protected resources on your system is based on the RACF user profile and its group membership.

For further information on defining connections for MRO, LUTYPE6.1, and APPC, see the *CICS RACF Security Guide*.

SINGLESESS({NO|YES})

specifies whether the definition is for an APPC terminal on a single session APPC link to CICS.

NO The definition is not for a single session APPC link to CICS.

YES The definition is for an APPC terminal on a single session APPC link to CICS.

The MODENAME attribute of the SESSIONS definition can be used to supply a modename for the single session mode set.

An APPC single session terminal can also be defined as a TERMINAL-TYPETERM definition. Both the TERMINAL-TYPETERM definition and the CONNECTION definition can be autoinstalled. If you are considering using autoinstall, see “Chapter 8. Autoinstall for VTAM terminals” on page 121 and “Chapter 10. Autoinstall for APPC connections” on page 141.

USEDFLTUSER ({NO|YES}) (APPC and MRO only)
specifies the kind of security checking that will take place for each inbound attach FMH.

NO Indicates that each inbound attach FMH will be checked for the presence of those fields required by the ATTACHSEC option and if the required fields are not present a protocol violation message will be issued and the attach will fail. NO is the default.

YES Indicates that some checks on the validity of the attach FMH are bypassed. This provides the same level of security as in releases of CICS prior to CICS/ESA 4.1. See *'Attach Time Security and the USEDFLTUSER option'* in the *CICS RACF Security Guide*.

XLNACTION({KEEP|FORCE}) (APPC and MRO only)
specifies the action to be taken when a new logname is received from the partner system. Receipt of a new logname indicates that the partner has deleted its recovery information.Recovery information is deleted when an INITIAL start is performed or a SET CONNECTION NORECOVDATA command is issued. This parameter is effective only in one of these circumstances.

Note: MRO here covers connections with ACCESSMETHOD set to either IRC or XM.

FORCE
The predefined decisions for in-doubt UOWs (as defined by the indoubt attributes of the transaction definition) are implemented, before any new work with the new logname is started. CICS also deletes any information retained for possible resolution of UOWs that were in-doubt at the partner system.

Attention: Data integrity may be compromised if you use this option.

KEEP Recovery information is kept, and no action is taken for in-doubt units of work.

For IRC, the connection continues with new work. Resolve in-doubt UOWs using the CEMT or SPI interface.

For APPC, the connection is unable to perform new work that requires synclevel 2 protocols until all outstanding recoverable work with the partner (that is, in-doubt UOWs, or information relevant to UOWs that were in-doubt on the partner system under the old logname) is completed using the CEMT or SPI interface.

Note: On IRC connections to pre-CICS Transaction Server for OS/390 systems, and on LU6.1 connections to all levels of CICS, lognames are not used and the XLNACTION attribute is ignored. For detailed information about IRC connections to back-level systems, see the *CICS Intercommunication Guide*.

DB2CONN

DB2CONN defines the global attributes of the connection to be established between CICS and DB2 as well as the attributes of pool threads and command threads to be used with the connection.

Note: DB2CONN definitions can be installed using a non-CSD definition with the EXEC CICS CREATE command, and EXEC CICS DISCARD enables you to discard DB2CONN definitions from a running system.

Installing and discarding a DB2CONN

This section describes the guidelines for installing and discarding DB2CONN definitions and the implications of interruptions in partial activity.

- Only one DB2CONN can be installed in a CICS system at any one time. An install of a second DB2CONN can implicitly DISCARD the existing DB2CONN and its associated DB2ENTRYs and DB2TRANs (unless reinstalling a DB2CONN of the same name) before proceeding with the installation.
- A DB2CONN must be installed before any DB2ENTRY or DB2TRAN definitions. DB2ENTRY and DB2TRAN definitions cannot exist on their own, and can only be associated with a DB2CONN that is already installed. Also, if you discard a DB2CONN, the associated DB2ENTRY and DB2TRAN resource definitions are also discarded. Note that there is no parameter on a DB2ENTRY or DB2TRAN that explicitly specifies the DB2CONN to which they belong. This allows DB2ENTRY and DB2TRAN definitions to be shared by DB2CONN definitions without alteration.

  Note: When DB2CONN, DB2ENTRYs, and DB2TRANs are defined in the same group, CICS automatically installs the DB2CONN first. If you install DB2 definitions from multiple groups (by means of a list or multiple INSTALL GROUP commands), the first group you install must contain the DB2CONN definition. Successive groups should not have any DB2CONN definitions. CICS issues an error message when installing a DB2ENTRY or DB2TRAN when no DB2CONN is installed. If multiple DB2CONN definitions are installed, all DB2 definitions installed before the final DB2CONN definition are discarded. CICS issues messages for all discards.
- A DB2CONN must be installed before the CICS DB2 connection can be started. Because it contains information regarding pool threads and command threads, as well as global type information, a DB2CONN represents the minimum required to start the CICS DB2 connection. There are no entry threads, and all transactions use the pool. You can add DB2ENTRYs and DB2TRANs after the CICS DB2 connection is active.
- A DB2CONN can be re-installed only if the CICS DB2 attachment facility is not connected to DB2, and therefore inactive.
- A DB2CONN can be discarded only when the CICS DB2 attachment facility is not connected to DB2.
- The discard of a DB2CONN implicitly discards all installed DB2ENTRYs and DB2TRANs.

  Note: There is no group commit or group discard of DB2CONNs, DB2ENTRYs, and DB2TRANs. However when a DB2CONN is discarded, the underlying control block is marked stating that a discard is in progress. The DB2ENTRYs and DB2TRANs are discarded before the DB2CONN. If the discard fails when half completed, a DB2CONN control block results, and a message is issued that a discard is in progress. A start of the CICS DB2 attachment facility fails with a message

```
DFHDB2074 CICS DB2 ATTACHMENT FACILITY STARTUP
          CANNOT PROCEED AS THE CURRENTLY
          INSTALLED DB2CONN IS NOT USABLE
```

when using a partially discarded DB2 resource definition. You must re-issue the discard. When a CICS system restarts after a failure when discarding, it knows that a discard took place. CICS does not recover the blocks from the catalog, and this effectively completes the discard. (Note that the definitions are removed from the catalog as well).

When you are installing, parts of any group or list install can fail, but messages are displayed that identify which resources have failed. You can proceed with a start of the CICS DB2 attachment facility when this happens.

Defining a DB2CONN

```
 DB2Conn      ==>
 Group        ==>
 DEscription  ==>
CONNECTION ATTRIBUTES
 CONnecterror ==> Sqlcode           Sqlcode | Abend
 DB2id        ==>
 MSGQUEUE1    ==> CDB2
 MSGQUEUE2    ==>
 MSGQUEUE3    ==>
 Nontermrel   ==> Yes               Yes | No
 PUrgecycle   ==> 00 , 30           0-59
 SIgnid       ==>
 STANdbymode  ==> Reconnect         Reconnect | Connect | Noconnect
 STATsqueue   ==> CDB2
 TCblimit     ==> 0012              4-2000
 THREADError  ==> N906D             N906D | N906 | Abend
POOL THREAD ATTRIBUTES
 ACcountrec   ==> None              None | TXid | TAsk | Uow
 AUTHId       ==>
 AUTHType     ==>                   Userid | Opid | Group | Sign | TErm
                                    | TX
 DRollback    ==> Yes               Yes | No
 PLAN         ==>
 PLANExitname ==>
 PRiority     ==> High              High | Equal | Low
 THREADLimit  ==> 0003              3-2000
 THREADWait   ==> Yes               Yes | No
COMMAND THREAD ATTRIBUTES
 COMAUTHId    ==>
 COMAUTHType  ==>                   Userid | Opid | Group | Sign | TErm
                                    | TX
 COMThreadlim ==> 0001              0-2000
```

Figure 19. The DEFINE panel for DB2CONN

Options

DB2CONN(*name*)
The name to identify a DB2CONN. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ and #.

Connection attributes

CONNECTERROR({SQLCODE|ABEND})
Specifies the way that the information, that CICS is not connected to DB2

because the attachment facility is in 'standby mode', is reported back to an application that has issued an SQL request.

ABEND
The application abends with abend code AEY9.

SQLCODE
The application receives a -923 sqlcode. SQLCODE cannot be specified if STANDBYMODE is set to NOCONNECT.

DB2ID(*name*)
Specifies the name of the DB2 subsystem to which the CICS DB2 attachment facility is to connect. The default is blanks. This DB2id can be overridden by a DB2 subsystem id specified on a DSNC STRT command, by a DB2ID specified on a SET DB2CONN command, or by a subsystem id specified using INITPARM. If no override is used, the default of blanks is replaced by DSN when connection is attempted. Hence, the hierarchy for determining the DB2 subsystem is as follows: #

1. Use the subsystem ID if specified in a DSNC STRT command. #
2. Use the DB2ID in the installed DB2CONN if not blank. #
3. Use the subsystem ID if specified on the INITPARM when the DB2ID in the last isntalled DB2CONN is blank (or has subsequently been set to blanks). On any startup, INITPARM is always used if the last installed DB2CONN contained a blank DB2ID, even if the DB2ID has been consequently changed using a SET command. #
4. Use a default subsystem ID of DSN. #

MSGQUEUE1({CDB2*name*}
Specifies the first transient data destination to which unsolicited messages from the CICS DB2 attachment facility are sent. This first destination cannot be blank

MSGQUEUE2(*name*)
Specifies a second transient data destination to which unsolicited messages from the CICS DB2 attachment facility are sent.

MSGQUEUE3(*name*)
Specifies a third transient data destination to which unsolicited messages from the CICS DB2 attachment facility are sent

NONTERMREL({YES | NO})
Specifies whether or not a non-terminal transaction releases threads for reuse at intermediate syncpoints.

NO Non-terminal transactions do not release threads for reuse at intermediate syncpoints.

YES Non-terminal transactions release threads for reuse at intermediate syncpoints.

PURGECYCLE({0 | value 1},{30,value 2})
Specifies the duration, in minutes and seconds, of the purge cycle for protected threads. The default is 0, 30; that is, 30 seconds. A protected thread is not terminated immediately when it is released. It is terminated only after two completed purge cycles, if it has not been reused in the meantime. Therefore, if the purge cycle is set to 30 seconds, a protected thread is purged 30 - 60 seconds after it is released. The first purge cycle after the attachment facility starts is always 5 minutes. After that the purgecycle values are applied. An unprotected thread is terminated when it is released (at syncpoint or end of task) if there are no other transactions waiting for a thread on that DB2ENTRY.

Only threads belonging to a DB2ENTRY can be protected. Pool threads and command threads cannot be protected. See page 182.

SIGNID(*name*)
Specifies the authorization ID to be used by the CICS DB2 attachment facility when signing on to DB2 for pool and DB2ENTRY threads that specify AUTHTYPE(SIGN). The default is blanks which are replaced by the applid of the CICS system when the DB2CONN is installed.

STANDBYMODE({RECONNECT | CONNECT | NOCONNECT})
Specifies the action to be taken by the CICS DB2 attachment facility if DB2 is not active when an attempt is made to connect CICS to DB2.

CONNECT
Specifies that the CICS DB2 attachment facility is to wait in 'standbymode' for DB2 to become active. If the connection is made, and DB2 subsequently fails, the CICS DB2 attachment facility terminates.

NOCONNECT
Specifies that the CICS DB2 attachment facility is to terminate.

RECONNECT
Specifies that the CICS DB2 attachment facility is to go into 'standby mode' and wait for DB2. If DB2 subsequently fails after the connection is made, the CICS DB2 attachment facility reverts to 'standby mode', and CICS subsequently reconnects to DB2 when DB2 recovers.

STATSQUEUE({CDB2 | name})
Specifies the transient data destination for CICS DB2 attachment facility statistics produced when the CICS DB2 attachment facility is shut down.

THREADERROR({N906D | N906 | ABEND})
Specifies the processing that is to occur following a create thread error.

ABEND
| For a second or subsequent SQL error the transaction abends with abend code AD2S, AD2T or AD2U, depending on the type of error that occurred. The transaction must be terminated and reinitialized before it is allowed to issue another SQL request.

N906D
A transaction dump is to be taken and the DSNCSQL RMI associated with the transaction is *not* to be disabled. The transaction receives a -906 SQLCODE if another SQL is issued, unless the transaction issues SYNCPOINT ROLLBACK. SYNCPOINT without the ROLLBACK option results in an ASP3 or ASP7 abend. The transaction dump records an abend of AD2S, AD2T or AD2U.

N906 The DSNCSQL RMI associated with the transaction is *not* to be disabled. The transaction receives a -906 SQLCODE if another SQL request is issued, unless the transaction issues a SYNCPOINT ROLLBACK. SYNCPOINT without the ROLLBACK option results in an ASP3 or ASP7 abend.

TCBLIMIT({12 | value})
Specifies the maximum number of subtasks (TCBs) that can be attached by the CICS DB2 attachment facility to process DB2 requests. The default is 12. The minimum number is 4 and the maximum is 2000. The value controls the total number of threads for the CICS region. For this reason, the recommended value for TCBLIMIT is the sum of all the thread limit values (that is, the sum

of all THREADLIMIT parameters on the DB2CONN and DB2ENTRY resource definitions, plus the COMTHREADLIMIT value on the DB2CONN) up to the limit of 2000.

Note: If you specify THREADLIMIT values which, in total, exceed the number of TCBs (TCBLIMIT) specified, a task could acquire a thread and then find that there is no available TCB. In this case, the task is suspended, with the INQUIRE TASK command showing HTYPE(CDB2TCB) with no corresponding HVALUE. This contrasts with the case where a task is suspended waiting for a thread, which shows HTYPE(CDB2RDYQ) and HVALUE(*POOL), HVALUE(*COMD), or HVALUE(*entry_name*), as appropriate.

When determining the number for TCBLIMIT, you must consider the amount you specified for the MAX USERS parameter on DB2 installation panel DSNTIPE.

Pool thread attributes

ACCOUNTREC({NONE | TASK | TXID | UOW})

Specifies the minimum amount of DB2 accounting required for transactions using pool threads. The specified minimum may be exceeded as described in the following options.

NONE

No accounting records are required for transactions using pool threads.

DB2 produces at least one accounting record for each thread when the thread is terminated. Authorization changes additionally cause accounting records to be produced.

TASK The CICS DB2 attachment facility causes a minimum of one accounting record for each CICS task to be produced.

A transaction containing multiple UOWs (assuming the thread is released at syncpoint) may use a different thread for each of its UOWs. The result may be an accounting record produced for each UOW.

TXID The CICS DB2 attachment facility causes an accounting record to be produced when the transid using the thread changes.

Because pool threads are typically used by a number of different transaction IDs, there is an increased chance that a transaction containing multiple units of work (UOWs) will use a different thread for each UOW (assuming the thread is released at syncpoint). In this case an accounting record may be produced per UOW.

UOW The CICS DB2 attachment facility causes an accounting record to be produced for each UOW, assuming that the thread is released at the end of the UOW.

AUTHID(*name*)

Specifies the ID that should be used for security checking when using pool threads. If AUTHId is specified, AUTHTYPE may not be specified.

AUTHTYPE({USERID | OPID | GROUP | SIGN | TERM | TX})

Specifies the type of ID that can be used for threads on this DB2ENTRY. If AUTHTYPE is specified, AUTHID may not be specified.

USERID

The 1 to 8-character USERID associated with the CICS transaction is used as the authorization ID. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

When the DB2 sample sign-on exit DSN3@SGN is used with AUTHTYPE(USERID), the exit sends the user ID to DB2 as the primary authorization ID and the RACF group ID to DB2 as the secondary ID. When the sample sign-on exit is used, there is no difference between AUTHTYPE(USERID) and AUTHTYPE(GROUP).

OPID The operator identification that is associated with the userid that is associated with the CICS transaction is used as the authorization ID (three characters padded to eight).

GROUP
Specifies the 1 to 8-character USERID and the connected group name as the authorization ID. The following table shows how these two values are interpreted by DB2.

	IDs passed to DB2	**How DB2 interprets values**
	CICS sign-on user ID (USERID)	Represents the primary DB2 authorization ID.
	RACF connected group name	If the RACF list of group options is not active, DB2 uses the connected group name supplied by the CICS attachment facility as the secondary DB2 authorization ID. If the RACF list of group options is active, DB2 ignores the connected group name supplied by the CICS attachment facility, but the value appears in the DB2 list of secondary DB2 authorization IDs.

To use the GROUP option, the CICS system must have SEC=YES specified in the CICS system initialization table (SIT).

If no RACF group ID is available for this USERID, an 8-character field of blanks is passed to DB2 as the group ID.

SIGN Specifies that the SIGNID parameter of the DB2CONN is used as the resource authorization ID.

TERM Specifies the terminal identification (four characters padded to eight) as an authorization ID. An authorization ID cannot be obtained in this manner if a terminal is not connected with the transaction.

If a transaction is started (using a CICS command) and has no terminal associated with it, AUTHTYPE(TERM) should not be used.

TX Specifies the transaction identification (four characters padded to eight) as the authorization ID.

DROLLBACK({YES|NO})
Specifies whether or not the CICS DB2 attachment facility should initiate a SYNCPOINT ROLLBACK if a transaction is selected as the victim of a deadlock resolution.

YES The attachment facility issues a syncpoint rollback before returning control to the application. An SQL return code of -911 is returned to the program.

NO The attachment facility does not initiate a rollback for a transaction. An SQL return code of -913 is returned to the application.

PLAN(*name*)
Specifies the name of the plan to be used for all pool threads. If PLAN is specified, PLANEXITNAME may not be specified.

PLANEXITName({DSNCUEXT|name})
Specifies the name of the dynamic plan exit to be used for pool threads. If you change the PLAN and PLANEXITNAME while there are active transactions for the pool, the next time the transaction releases the thread the plan/exit will be determined using the new rules. If PLANEXITNAME is specified, PLAN may not be specified.

PRIORITY({HIGH|EQUAL|LOW})
Specifies the priority of the pool thread subtasks relative to the CICS main task (QR TCB).

HIGH Subtasks attain a higher priority than the CICS main task from which the subtask was generated.

EQUAL
Subtasks have equal priority with the CICS main task.

LOW Subtasks have a lower priority than the CICS main task.

THREADLIMIT({3|value})
Specifies the current maximum number of pool threads that the CICS DB2 attachment facility allows to be active before requests are made to wait or are rejected (subject to the THREADWAIT parameter). The default threadlimit (3)
# is also the minimum you can specify. The maximum value must not be greater
# than the value specified for TCBLIMIT.

THREADWAIT({YES|NO})
Specifies whether or not transactions should wait for a pool thread, or be abended if the number of active pool threads reaches the thread limit.

The CICS DB2 attachment issues a unique abend code AD3T, message DFHDB2011, when THREADWAIT=NO is coded and the number of pool threads is exceeded.

YES If all threads are busy, a transaction must wait until one becomes available. A transaction can wait as long as CICS allows it to wait, generally until a thread becomes available.

NO If all threads are busy, the transaction is terminated with abend code AD2T or AD3T.

Command thread attributes

COMAUTHID(*name*)
Specifies what id the CICS DB2 attachment facility should use for security checking when using command threads. If COMAUTHId is specified, COMAUTHType may not be specified.

COMAUTHTYPE({USERID|OPID|GROUP|SIGN|TERM|TX})
Specifies the type of id that can be used for security checking when using command threads. If COMAUTHType is specified, COMAUTHid may not be specified.

USERID
The 1 to 8-character userid associated with the CICS transaction is used as the authorization ID. The name can be up to eight characters in

length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

When the DB2 sample sign-on exit DSN3@SGN is used with AUTHTYPE(USERID), the exit sends the USERID to DB2 as the primary authorization ID and the RACF group ID to DB2 as the secondary ID. When the sample sign-on exit is used, there is no difference between COMAUTHTYPE(USERID) and COMAUTHTYPE(GROUP).

OPID The operator identification associated with the userid that is associated with the CICS transaction sign-on facility is used as the authorization ID (three characters padded to eight).

GROUP
Specifies the 1 to 8-character USERID and the connected group name as the authorization ID. The following table shows how these two values are interpreted by DB2.

	IDs passed to DB2	**How DB2 interprets values**
	CICS sign-on user ID (USERID)	Represents the primary DB2 authorization ID.
	RACF connected group name	If the RACF list of group options is not active, DB2 uses the connected group name supplied by the CICS attachment facility as the secondary DB2 authorization ID. If the RACF list of group options is active, DB2 ignores the connected group name supplied by the CICS attachment facility, but the value appears in the DB2 list of secondary DB2 authorization IDs.

To use the CGROUP option the CICS system must have SEC=YES specified in the CICS system initialization table (SIT).

If no RACF group ID is available for this USERID, an 8-character field of blanks is passed to DB2 as the group ID.

SIGN Specifies that the SIGNID parameter of the DB2CONN is used as the resource authorization ID.

TERM Specifies the terminal identification (four characters padded to eight) as an authorization ID. An authorization ID cannot be obtained in this manner if a terminal is not connected with the transaction.

If a transaction is started (using a CICS command) and has no terminal associated with it, the COMAUTHTYPE(TERM) should not be used.

TX Specifies the transaction identification (four characters padded to eight) as the authorization ID.

COMTHREADLIMIT({$\underline{1}$|value})
The number specifies the current maximum number of command threads the CICS DB2 attachment facility allows active before requests overflow to the pool.

DB2ENTRY

DB2ENTRY defines resources to be used by a specific CICS transaction, or a group of transactions, when using the CICS DB2 interface. A DB2ENTRY can specify one specific transaction or a group of transactions may be represented by the use of one or more wildcard characters (see "Using wildcard characters for transaction IDs" on page 183). Also, further transactions can be associated with a DB2entry by defining a DB2Tran.

Note: DB2ENTRY definitions can be installed using a non-CSD definition with the EXEC CICS CREATE command, and EXEC CICS DISCARD enables you to discard DB2ENTRY definitions from a running system.

Installing and discarding a DB2ENTRY

This section describes the guidelines for installing and discarding DB2ENTRY definitions and the implications of interruptions in partial activity.

- You can install a DB2ENTRY only if you have previously installed a DB2CONN.
- You can install a new DB2ENTRY at any time, even when the CICS DB2 adapter is connected to DB2.
- You can reinstall (by replacing an existing DB2ENTRY) only when the DB2ENTRY is disabled and no transaction is using it. Use the SET DB2ENTRY DISABLED command to quiesce activity and disable the entry. New transactions trying to use the DB2ENTRY are routed to the pool, abended, or returned an SQLCODE, dependent on the setting of the DISABLEDACT keyword when the DB2ENTRY is disabled.
- You can discard a DB2ENTRY only if it is disabled. Use the SET DB2ENTRY DISABLED command to quiesce activity and disable the entry. New transactions trying to use the DB2ENTRY are routed to the pool, abended, or returned an SQLCODE, dependent on the setting of the DISABLEDACT keyword when the DB2ENTRY is disabled.

  If you discard a DB2ENTRY you could make the corresponding DB2TRAN an 'orphan'. If you then run a transaction, a message is sent to the CDB2 transient data destination, and the request is rerouted to the pool.

Defining a DB2ENTRY

```
 DB2Entry     ==>
 Group        ==>
 DEscription  ==>
THREAD SELECTION ATTRIBUTES
 TRansid      ==>
THREAD OPERATION ATTRIBUTES
 ACcountrec   ==> None              None | TXid | TAsk | Uow
 AUTHId       ==>
 AUTHType     ==>                   Userid | Opid | Group | Sign | TErm
                                    | TX
 DRollback    ==> Yes               Yes | No
 PLAN         ==>
 PLANExitname ==>
 PRIority     ==> High              High | Equal | Low
 PROtectnum   ==> 0000              0-2000
 THREADLimit  ==> 0000              0-2000
 THREADWait   ==> Pool              Pool | Yes | No
```

Figure 20. The DEFINE panel for DB2ENTRY

Options

DB2ENTRY(*name*)
One to eight character name to identify a DB2ENTRY definition. Acceptable characters are those allowed for a transaction id.

Thread selection attributes

TRANSID(*name*)
Specifies the transaction id associated with the entry. Only one transaction can be specified here. However, the use of one or more wildcard characters in the TRANSID (see "Using wildcard characters for transaction IDs" on page 183) allows a group of transactions to be represented. Additional transactions can be defined for this entry by defining a DB2TRAN that refers to this DB2ENTRY. Transid is optional on a DB2ENTRY. All transactions can be associated with a DB2ENTRY by means of DB2TRANs instead. However, if only one transaction is associated with a DB2ENTRY it is easier to specify it on the DB2ENTRY.

Thread operation attributes

ACCOUNTREC({NONE | TASK | TXID | UOW})
Specifies the minimum amount of DB2 accounting required for transactions using this DB2ENTRY. The specified minimum may be exceeded, as described in the following options.

NONE
No accounting records are required for transactions using threads from this DB2ENTRY

However, DB2 produces at least one accounting record for each thread after the thread is terminated. Authorization changes additionally cause records to be produced.

TASK The CICS DB2 attachment facility causes a minimum of one accounting record for each CICS task to be produced.

A transaction containing multiple UOWs (assuming the thread is released at syncpoint) may use a different thread for each UOW. The result may be that an accounting record is produced for each UOW.

TXID The CICS DB2 attachment facility causes an accounting record to be produced when the transid using the thread changes.

This option applies to DB2ENTRYs that are used by more than one transaction ID. As threads are typically released at syncpoint, a transaction containing multiple UOWs may use a different thread for each UOW. The result may be that an accounting record is produced per UOW.

UOW The CICS DB2 attachment facility causes an accounting to be produced for each UOW, assuming that the thread is released at the end of the UOW.

AUTHID(*name*)
Specifies the id to be used for security checking when using this DB2ENTRY. If AUTHId is specified, AUTHTYpe may not be specified.

AUTHTYPE({USERID | OPID | GROUP | SIGN | TERM | TX})
Specifies the type of id that can be used for security checking when using this DB2ENTRY. If AUTHTYpe is specified, AUTHid may not be specified.

USERID
The 1 to 8-character USERID associated with the CICS transaction is

used as the authorization ID. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

When the DB2 sample sign-on exit DSN3@SGN is used with AUTHTYPE(USERID), the exit sends the user ID to DB2 as the primary authorization ID and the RACF group ID to DB2 as the secondary ID. When the sample sign-on exit is used, there is no difference between AUTHTYPE(USERID) and AUTHTYPE(GROUP).

OPID The operator identification that is associated with the userid that is associated with the CICS transaction sign-on facility, is used as the authorization ID (three characters padded to eight).

GROUP
Specifies the 1 to 8-character USERID and the connected group name as the authorization ID. The following table shows how these two values are interpreted by DB2.

	IDs passed to DB2	How DB2 interprets values
	CICS sign-on user ID (USERID)	Represents the primary DB2 authorization ID.
	RACF connected group name	If the RACF list of group options is not active, DB2 uses the connected group name supplied by the CICS attachment facility as the secondary DB2 authorization ID. If the RACF list of group options is active, DB2 ignores the connected group name supplied by the CICS attachment facility, but the value appears in the DB2 list of secondary DB2 authorization IDs.

To use the GROUP option the CICS system must have RACF external security SEC=YES specified in the CICS system initialization table (SIT).

If no RACF group ID is available for this USERID, an 8-character field of blanks is passed to DB2 as the group ID.

SIGN Specifies that the SIGNID parameter of the DB2CONN is used as the resource authorization ID.

TERM Specifies the terminal identification (four characters padded to eight) as an authorization ID. An authorization ID cannot be obtained in this manner if a terminal is not connected with the transaction.

If a transaction is started (using a CICS command) and has no terminal associated with it, AUTHTYPE(TERM) should not be used.

TX Specifies the transaction identification (four characters padded to eight) as the authorization ID.

DROLLBACK({YES|NO})
Specifies whether or not the CICS DB2 attachment should initiate a SYNCPOINT rollback in the event of a transaction being selected as victim of a deadlock resolution.

YES The attachment facility issues a syncpoint rollback before returning control to the application. An SQL return code of -911 is returned to the program.

NO The attachment facility does not to initiate a rollback for this transaction. An SQL return code of -913 is returned to the application.

PLAN(*name*)
Specifies the name of the plan to be used for this entry. If PLAN is specified, PLANEXITNAME cannot be specified.

PLANEXITName(DSNCUEXT)
Specifies the name of the dynamic plan exit to be used for this DB2ENTRY. If you change the PLAN and PLANEXITNAME while there are active transactions for the DB2ENTRY, the next time the transaction releases the thread, the plan/exit will be determined using the new rules. If PLANEXITNAME is specified, PLAN cannot be specified.

PRIORITY({HIGH|MEDIUM|LOW})
Specifies the priority of the thread subtasks for the DB2ENTRY relative to the CICS main task (QR TCB).

EQUAL
Subtasks have equal priority with the CICS main subtask.

HIGH Subtasks attain a higher priority than the CICS main task from which the subtask was generated.

LOW Subtasks have a lower priority than the CICS main task.

PROTECTNUM({0|value})
Specifies the maximum number of protected threads allowed for this DB2ENTRY. A thread, when it is released by a transaction and there is no other work queued, can be protected, meaning that it is not terminated immediately. A protected thread is terminated after only two complete purge cycles if it has not been reused in the meantime. Hence, if the purge cycle is set to 30 seconds, a protected thread is terminated 30 - 60 seconds after it is released, assuming it is not reused in the meantime. The first purge cycle after the CICS DB2 attachment facility has been started is 5 minutes, after which the PURGECYCLE value is applied. Threads are only protected while they are inactive. If a transaction reuses a protected thread, the thread becomes active, and the current number of protected threads is decremented.

THREADLIMIT({0|value})
Specifies the maximum number of threads for this DB2ENTRY that the CICS DB2 attachment allows active before requests are made to wait, are abended, or diverted to the pool.

THREADWAIT({POOL|YES|NO})
Specifies whether or not transactions should wait for a DB2ENTRY thread, be abended, or overflow to the pool should the number of active DB2ENTRY threads reach the THREADLimit number.

POOL If all threads are busy, the transaction is diverted to use the pool of threads. If the pool is also busy, and NO has been specified for the THREADWAIT parameter on the DB2CONN, the transaction is terminated with abend code AD3T.

NO If all threads are busy, a transaction is terminated with an abend code AD2P.

YES If all threads are busy, a transaction waits until one becomes available.

DB2TRAN

A DB2TRAN resource definition defines additional transactions to be associated with a particular DB2Entry. DB2Tran allows a DB2Entry to have an unrestricted number of transactions associated with it, including names using wildcard characters (see "Using wildcard characters for transaction IDs"). You can define any number of DB2Trans to be associated with a single DB2Entry.

Note: DB2TRAN definitions can be installed using a non-CSD definition with the EXEC CICS CREATE command, and EXEC CICS DISCARD enables you to discard DB2TRAN definitions from a running system.

Installing and discarding a DB2TRAN

This section describes the guidelines for installing and discarding DB2TRAN definitions and the implications of interruptions in partial activity.

- A DB2TRAN that refers to a non-existent DB2ENTRY cannot be installed. The DB2ENTRY must be installed first.
- Note that when DB2ENTRY and DB2TRAN definitions are defined in the same group, CICS installs the DB2ENTRY first at install time.
- You can install a new DB2TRAN at any time, even when the CICS DB2 adapter is connected to DB2.
- A DB2TRAN can be re-installed at any time, and can be discarded at any time.

Defining a DB2TRAN

```
DB2Tran      ==>
Group        ==>
Description  ==>
Entry        ==>
Transid      ==>
```

Figure 21. The DEFINE panel for DB2TRAN

Options

DB2TRAN(*name*)
The one to eight character name to identify this DB2Tran. Acceptable characters are all those allowed for a transaction ID.

ENTRY(*name*)
Specifies the name of the DB2Entry to which this DB2Tran refers. It is the DB2ENTRY with which this additional transaction should be associated.

TRANSID(*name*)
Specifies the transaction id to be associated with the entry. If the TRANSID is not specified it defaults to the first four characters of the DB2TRAN name. The transaction id can include wildcard characters (see "Using wildcard characters for transaction IDs").

Using wildcard characters for transaction IDs

Transaction IDs may be coded in generic form using asterisk and plus signs in the following ways:

- An asterisk (*) can be added to a transaction name, or used alone, to produce the effect on any value of making it 'wild'. The transaction name specified with an asterisk at the end of the name represents 0-3 unspecified characters in the transaction ID. For example, a TRansid of “T*” would represent transaction “T”, “TA”, “TAB”, “TABE”.

- A plus sign (+) is allowed in any position to represent any single character.
- An asterisk alone represents any transaction and can act as an alternative pool definition. It differs from a pool by having the additional attribute of the DB2ENTRY of overflowing to the pool when the thread allocation is exhausted.

The rules of matching are that the most specific match is taken. For example, transaction FRED will use DB2ENTRY(1) specifying a generic transaction id of “FRE*”, rather than DB2ENTRY(2) specifying a generic transaction id of “F*”. Also a ′+′ is more specific than a ′*′, eg “FRE+” is more specific than “FRE*”.

If AUTHTYPE(TX) is specified, the actual TXID is passed to DB2 as the primary authorization ID, and not the name that used wildcard characters.

Grouping a set of transaction identifiers together using wildcard characters can reduce the number of DB2ENTRY and DB2TRAN definitions required. However, this removes the ability to collect statistics by transaction. CICS–DB2 resource statistics are collected for each DB2ENTRY. If you want separate CICS–DB2 statistics for a specific transaction, you need to give that transaction its own DB2ENTRY definition.

DOCTEMPLATE

Use this resource definition to define document templates to CICS. Document templates allow you to perform variable substitution on documents in a manner similar to that done by BMS for 3270 screens.

The template can reside in any one of the following places:
- MVS partitioned data set
- CICS temporary storage
- CICS transient data
- CICS load module
- exit program

Defining a DOCTEMPLATE

```
 DOctemplate  ==>
 Group        ==>
 DEscription  ==>
FULL TEMPLATE NAME
 TEmplatename ==>
ASSOCIATED CICS RESOURCE
 File         ==>
 TSqueue      ==>
 TDqueue      ==>
 Program      ==>
 Exitpgm      ==>
PARTITIONED DATA SET
 DDname       ==>
 Membername   ==>
TEMPLATE PROPERTIES
 Appendcrlf   ==> Yes                Yes | No
 TYpe         ==>                    BINARY | EBCDIC
```

Figure 22. The DEFINE panel for DOCTEMPLATE

Options

APPENDCRLF
: specifies whether CICS is to delete trailing blanks from and append carriage-return line-feed to each logical record of the template.

DDNAME
: specifies the DD name of the PDS containing the template. The DDName applies only to a template of type PDS. If a membername is supplied without a value for DDNAME, the default value DFHHTML is used.

DESCRIPTION(*text*)
: You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

DOCTEMPLATE
: specifies the name of this DOCTEMPLATE definition. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ # . / - _ % & ¢ ? ! : | " = ¬ , ; < and >.

EXITPGM
specifies the exit program to be invoked when a request is made for this template. CICS passes a commarea to the exit program which is mapped by the following copybooks:

- DFHDHTXD (Assembler)
- DFHDHTXH (C)
- DFHDHTXL (PL/I)
- DFHDHTXO (COBOL)

FILE(*filename***)**
specifies, for a template of type FILE, the 8-character name of the CICS file definition for the data set containing the template.

GROUP(*groupname***)**
specifies the group name, which can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and $. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see “How the CSD file is organized—groups and lists” on page 8.

MEMBER
specifies the name of the member in the PDS containing the template. It applies only to a template of type PDS.

PROGRAM
specifies the program in which the template data is stored. CICS loads the program and takes all data after the entrypoint to be the template.

TDQUEUE
specifies the name of the TD queue on which the template is stored.

TEMPLATENAME
specifies the extended Template-name by which the doctemplate is to be known outside the resource definition function.

TSQUEUE
specifies the name of the TS queue on which the template is stored.

TYPE(BINARY|EBCDIC)
specifies the format of the contents of the template.

BINARY
When the template is loaded from the template library, no parsing of the template’s contents is done.

EBCDIC
When the template is loaded from the template library, the contents are parsed as EBCDIC text.

Note: The fields EXITPGM, FILE, MEMBER, PROGRAM, TDQUEUE, and TSQUEUE define alternative sources of the template data. Only one of them may be specified on each DOCTEMPLATE definition.

ENQMODEL

Use of ENQMODEL definitions provides an enqueue mechanism that serializes access to a named resource across a specified set of CICS regions operating with a sysplex. This applies equally to a CICSplex within a single MVS image and to a CICSplex that resides in more than one MVS.

Combined with an ENQMODEL resource definition, CICS uses MVS global resource serialization to provide sysplex-wide protection of application resources.

Local enqueues within a single CICS region are managed within the CICS address space. Sysplex-wide enqueues that affect more than one CICS region are managed by GRS.

The ENQSCOPE attribute of an ENQMODEL resource definition, defines the set of regions that share the same enqueue scope. If the ENQSCOPE attribute is left blank (the default value), CICS treats any matching ENQ |DEQ as local to the issuing CICS region. If the ENQSCOPE is non-blank, CICS treats the ENQ |DEQ as sysplex-wide and passes a queue name and the resource name to GRS to manage the enqueue.

The CICS regions that need to use sysplex-wide enqueue/dequeue function must all have the required ENQMODELs defined and installed. The recommended way to ensure this is for the CICS regions to share a CSD, and for the initialization group lists to include the same ENQMODEL groups.

Existing applications can use sysplex enqueues simply by defining appropriate ENQMODELs, without any change to the application programs. Those existing applications where the resource name is determined dynamically and not known in advance can only be so converted by use of the global user exit XNQEREQ and the parameter UEPSCOPE (see the *CICS Customization Guide* for details).

In a multi-tasking, multi-processing environment, resource serialization is the technique used to coordinate access to resources that are used by more than one program. MVS global resource serialization (GRS) provides a way to control access to such resources.

GRS combines systems into a global resource serialization complex, consisting of one or more systems that are:
- Connected to each other in a ring configuration
- Connected to a coupling facility lock structure in a star configuration

Recommendation:: For reasons of performance, a ring configuration is not recommended for production regions in a multisystem environment. You should use a star configuration only in a multisystem sysplex. Note that a coupling facility is required for a star configuration.

You can display sysplex ENQUEUEs using GRS facilities with the Display GRS command. This command has many optional keywords, but for this purpose you can use:

```
D  GRS,RES=(DFHEqname|*,[  rname|,*])
```

where:

qname
specifies a 4-character ENQSCOPE defined by an ENQMODEL

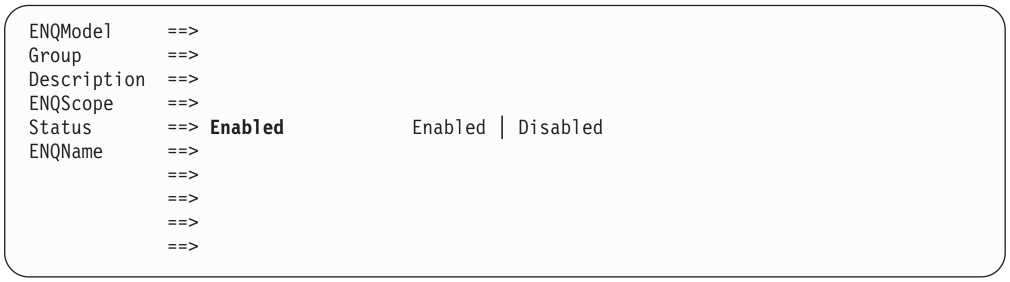
rname
specifies an ENQNAME defined by an ENQMODEL

For information about GRS, see OS/390 MVS Planning: Global Resource Serialization, GC28-1759.

Installing ENQMODEL definitions

As ENQMODELs usually have the default status of *ENABLED*, the order of installing ENQMODELs must follow the rules for ENABLING ENQMODELs; that is, ENQMODELs forming nested generic ENQnames must be enabled in order from the most to the least specific. For example, ABCD* then ABC* then AB*.

Defining an ENQMODEL



```
ENQModel     ==>
Group        ==>
Description  ==>
ENQScope     ==>
Status       ==> Enabled            Enabled | Disabled
ENQName      ==>
             ==>
             ==>
             ==>
             ==>
```

Figure 23. The DEFINE panel for ENQMODEL

Options

DESCRIPTION(*text*)
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

ENQMODEL(*name*)
specifies the name of this ENQMODEL definition. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ # . / - _ % & ¢ ? ! : | " = ¬ , ; < and >.

This name is used to identify the ENQMODEL definition on the CSD file. It is not used within the active CICS system.

ENQNAME(*text*)
specifies the 1 to 255-character resource name. The acceptable characters are: A-Z 0-9 $ @ # * . / - _ % & ¢ ? ! : | " = ¬ , ; < and >. The * (asterisk) is treated as a wildcard when it appears as the last character of the resource name, such names are treated as generic.

ENQSCOPE(*scope-name*)
specifies the optional 4-character enqmodel scope name. If omitted or specified as blanks, matching ENQs will have LOCAL scope.

GROUP(*groupname*)
specifies the group name, which can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and $. Lowercase characters are treated as

uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see "How the CSD file is organized—groups and lists" on page 8.

STATUS({ENABLED | DISABLED})
specifies whether the ENQMODEL is to be installed in ENABLED or DISABLED status. ENABLED is the default.

ENABLED
Matching ENQ requests are processed in the normal way.

DISABLED
Matching ENQ requests are rejected, and the issuing task is abended. Matching INSTALL CREATE and DISCARD requests are processed.

FILE

The following resources associated with CICS files can be managed using RDO:
- VSAM files (this includes files that refer to CICS-maintained, user-maintained, and coupling facility data tables as well as files that refer to VSAM data sets)
- Remote VSAM files
- Remote BDAM files
- VSAM local shared resource (LSR) pools

You use the FILE definition to describe to CICS file control the physical and operational characteristics of a file. The definition includes attributes that provide information about record characteristics, the types of operations allowed on the file, recovery attributes, and the operations that are to be journaled. This information is used to generate control information used by CICS. CICS files usually correspond to physical data sets that must have been defined to VSAM before they are used. Using CICS files, your applications can:
- Access records in the data set directly
- Access records in a data table that has been loaded from the data set
- Access records in a coupling facility data table where there is no data set involved (because LOAD(NO) is specified on the CFDT file definition).

For the file to be used by an active CICS system, its definition must have been installed on to the system. CICS file control uses the installed definition to find the file when it needs to access it, to keep count of the number of tasks using the file, to capture processing statistics, and maintain other file control information.

Remote files

When multiple CICS systems are connected, they can share each other's files; all such files must be defined to CICS, including those belonging to another system. Files on other systems are defined as 'remote'. (This does not apply to files accessed in RLS mode, or coupling facility data tables, which are always defined as local files. Remote files are accessed through CICS function shipping of file control requests to the remote region that owns the file.)

The resource attributes needed by CICS for remote files are not specific to the access method used. You can therefore define both remote BDAM files and remote VSAM files using RDO.

If you name a REMOTESYSTEM, you may also supply a REMOTENAME, which is the name of the file used in the remote system.

If you specify a REMOTESYSTEM name that corresponds to the SYSIDNT of the CICS region in which the file definition is installed, CICS installs the definition as a local file. Otherwise, CICS installs the definition as a remote file.

Coupling facility data tables

Coupling facility data table support provides a method of file data sharing, using CICS file control, without the need for a file-owning region, and without the need for VSAM RLS support. Data is held in a coupling facility list structure, in a table that is similar in many ways to a shared user-maintained data table.

Unlike user-maintained data tables, coupling facility data tables do not have to be pre-loaded from a source data set. Loading a coupling facility data table is controlled by the DSNAME and LOAD attributes of the file resource definition, which allows CFDTs to be populated entirely by the application programs that use them by specifying LOAD(NO).

The way you use LOAD(YES) and the DSNAME attributes allows you to control loading of a CFDT in various ways, such as:

1. Any CICS region can load the coupling facility data table. The first file open for the CFDT loads it, regardless of which CICS region issues the open. The data set is opened read-only by the loading CICS. All file definitions for the table specify LOAD(YES) and the DSNAME of a source data set. If you use this approach, ensure that the same data set is named on each file definition, otherwise the data set named on the first to be opened is the one that is loaded into the CFDT. CICS does not verify that the DSNAMEs are the same for all files that refer to the same CFDT.
2. One CICS region can be made responsible for loading the coupling facility data table. The loading region contains a file definition for the CFDT that specifies LOAD(YES) and the DSNAME for the data set, which is opened read-only by the loading CICS. Other CICS regions do not need access to the source data set, but they cannot open the CFDT until the loading region has opened it. The file definitions for the CFDT in non-loading regions must also specify LOAD(YES) but omit the DSNAME.

   You can restrict access to a coupling facility data table until after it has been loaded by using two (or more) file names that refer to the same coupling facility data table. To control access in this way:

   - Define only one file name as being capable of loading the data table, by specifying LOAD(YES) and DSNAME(*dataset_name*) Do not refer to this file name from application programs.
   - Define another file (or files) for application program use, ensuring that this file definition cannot initiate table loading. Specify LOAD(YES), but ensure the data set name is not specified on the DSNAME parameter, and that there is no DD statement for this file (or files).
   - Ensure that the load-capable file is opened before any application programs are likely to require access to the data table. For example, define the table-loading file name with OPENTIME(STARTUP) to ensure that the file is opened automatically toward the end of CICS initialization.
   - Ensure that application programs access the data table by referring to a filename that does not load the data.
3. Some hybrid of the above two approaches can be used, where some CICS regions can load the table, and others require it to be loaded on their behalf.

Installing FILE definitions

The following procedure uses the CICS CEMT and CEDA transactions to install a FILE definition:

1. If the file already exists, ensure that it is closed and disabled:
   ```
   CEMT SET FILE(filename) CLOSED DISABLED
   ```
2. Install the file definition:
   ```
   CEDA INSTALL GROUP(groupname) FILE(filename)
   ```
3. When you have successfully installed the group containing the file, use CEMT to open and enable the file (if it is not already defined as enabled, and if you want to open the file explicitly):
   ```
   CEMT SET FILE(filename) OPEN ENABLED
   ```

As an alternative to CEMT, you can use the EXEC CICS SET FILE command in a user-written transaction to disable and enable the file.

Defining a FILE

The CEDA DEFINE panel for defining a file resource definition is shown in Figure 24.

```
 File        ==>
  Group        ==>
  DEscription  ==>

VSAM PARAMETERS
 DSNAme       ==>
 Password     ==>                     PASSWORD NOT SPECIFIED
 RLSACCESS    ==> NO                  YES|NO
 Lsrpoolid    ==> 1                   1-8 | None
 READINTEG    ==> UNCOMMITTED         UNCOMMITTED|CONSISTENT|REPEATABLE
 DSNSharing   ==> Allreqs             Allreqs | Modifyreqs
 STRings      ==> 001                 1 - 255
 Nsrgroup     ==>

REMOTE ATTRIBUTES
 REMOTESystem ==>
 REMOTEName   ==>
REMOTE AND CFDATATABLE PARAMETERS
 RECORDSize   ==>                     1 - 32767
 Keylength    ==>                     1 - 255 (1-16 For CF Datatable)
INITIAL STATUS
 STAtus       ==> Enabled             Enabled | Disabled | Unenabled
 Opentime     ==> Firstref            Firstref | Startup
 DIsposition  ==> Share               Share | Old

BUFFERS
 DAtabuffers  ==> 00002               2 - 32767
 Indexbuffers ==> 00001               1 - 32767

DATATABLE PARAMETERS
 Table        ==> No                  No | Cics | User | CF
 Maxnumrecs   ==> Nolimit             Nolimit | 1-99999999

CFDATATABLE PARAMETERS
 Cfdtpool     ==>
 TABLEName    ==>
 UPDATEModel  ==> Locking             Contention | Locking
 LOad         ==> No                  No | Yes
DATA FORMAT
 RECORDFormat ==> V                   V | F

OPERATIONS
 Add          ==> No                  No | Yes
 BRowse       ==> No                  No | Yes
 DELete       ==> No                  No | Yes
 REAd         ==> Yes                 Yes | No
 Update       ==> No                  No | Yes

AUTO JOURNALING
 JOurnal      ==> No                  No | 1 - 99
 JNLRead      ==> None                None | Updateonly | Readonly | All
 JNLSYNCRead  ==> No                  No | Yes
 JNLUpdate    ==> No                  No | Yes
 JNLAdd       ==> None                None | Before | AFter |ALl
 JNLSYNCWrite ==> Yes                 Yes | No

RECOVERY PARAMETERS
 RECOVery     ==> None                None | Backoutonly | All
 Fwdrecovlog  ==> No                  No | 1-99
 BAckuptype   ==> STAtic              STAtic | DYNamic

SECURITY
 RESsecnum      : 00                  0-24 | Public
```

Figure 24. The DEFINE panel for FILE

Options

ADD(NO | YES)
specifies whether records can be added to the file.

BACKUPTYPE(STATIC | DYNAMIC)
CICS VSAM files can be defined as eligible for backup while open for update. This attribute is not used for files defined with RLSACCESS(YES), or if the recovery options are defined in the ICF catalog. For files that are accessed in RLS mode, you must specify the backup type on the data set definition in the ICF catalog.

This attribute is ignored for coupling facility data tables and, if there are any recovery attributes defined in the ICF catalog for a source data set associated with the table, these also are ignored. A CFDT is not eligible for backup-while-open (BWO).

Possible values are:

DYNAMIC
Specify this along with the RECOVERY attribute of ALL to make the file eligible for backup while open for update.

STATIC
The file is not eligible for backup while open for update.

BROWSE(NO | YES)
specifies whether records can be retrieved sequentially from the file.

CFDTPOOL(*name*)
specifies the name of the coupling facility data table pool containing the table defined by this file definition.

This attribute is required if you specify TABLE(CF). The name can be up to 8 characters in length, from the allowable characters of: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase when the definition is written to the CSD.

Coupling facility data tables can be separated (for purposes of accounting, security, administration, and so on) into groups (pools) of CFDTs. The names of all coupling facility data table pools must be unique within the sysplex, but can be the same as the names of other types of pools, such as TS data sharing pools.

Opening a file that references a coupling facility data table requires that a coupling facility data table server for the named pool is running in the MVS in which the open request is issued. If the required server has not been started, the file open request fails.

Note: The CFDTPOOL attribute is meaningful only for CFDTs. You can specify a pool name for a file that is not defined as TABLE(CF), but CICS ignores it. If you subsequently alter the file definition to reference a coupling facility data table, the CFDTPOOL name comes into effect.

DATABUFFERS(2 | *value*)
specifies the number of buffers to be used for data. Use a value in the range 2 through 32767. The minimum value you may specify is one more than the number of strings defined in the STRINGS attribute.

DELETE(NO | YES)
specifies whether records can be deleted from the file.

DESCRIPTION(*text*)
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

DISPOSITION(SHARE | OLD)
specifies the disposition of this file.

OLD Equivalent to the DISP=OLD parameter in JCL.

SHARE
Equivalent to the DISP=SHR parameter in JCL.

| **DSNAME(*name*)**
| specifies the data set name (as known to the operating system) to be used for
| this file. DSNAME can be 1 through 44 characters, conforming to the rules for
| MVS data set names (see the DSNAME parameter in the OS/390 MVS JCL
Reference). The characters allowed are A-Z 0-9 @ # $ and -. Lowercase
| characters are treated as uppercase characters.

| At file open time, if no JCL statement exists for this file, the open is preceded
| by a dynamic allocation of the file using this DSNAME. If the file definition
| refers to a data table (CICS, USER, or CF) the DSNAME must be that of a
| VSAM base KSDS.

| The DSNAME specified on a DD statement for this file in the CICS start-up
| JCL takes precedence over the DSNAME specified in this file definition.

| **Coupling facility data tables**
| If the file definition specifies LOAD(YES) and it is not already opened,
| DSNAME specifies the name of the source data set from which the
| table is to be loaded. Alternatively, you can specify the source data set
| on a DD statement in the CICS startup JCL. The data set specified
| must be a VSAM base KSDS.

| If you want table loading to be initiated by the opening of another file
| specified by a different file definition, omit this attribute. In this case,
| also ensure that the file name is not specified on a DD statement in the
| CICS JCL. Attempts to open the file fail until CFDT loading has been
| initiated. For more information about loading a coupling facility data
| table from a data set, see “Coupling facility data tables” on page 190.

| If LOAD(NO) is specified, this attribute is not required and is ignored.

| **DSNSHARING(ALLREQS | MODIFYREQS)**
specifies whether VSAM data set name sharing is used for the VSAM file. The possible values are:

ALLREQS
Data set name sharing is set in the ACB when the file is opened and is therefore used for all file requests.

MODIFYREQS
Data set name sharing is set in the ACB when the file is opened only if an operation of DELETE, ADD, or UPDATE is set for the file.

FILE(*name***)**
specifies the name of the file. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

FWDRECOVLOG(NO | *number***)**
specifies the journal name that corresponds to the MVS system logger log stream that is to be used for forward recovery.

| This attribute is ignored for coupling facility data tables and, if there are any
| recovery attributes defined in the ICF catalog for a source data set associated
| with the table, these also are ignored. A CFDT is not forward recoverable.

NO Forward recovery logging is not required for this file.

number
The number that identifies the journal that CICS is to use for the forward recovery log. CICS journal names are of the form DFHJ*nn* where *nn* is in the range 1 through 99. The after images for forward recovery are written to the MVS log stream that corresponds to journal name DFHJ*nn*.

Note: Unlike in releases prior to CICS Transaction Server for OS/390 Release 1, DFHJ01 is **not** the system log.

This attribute is used by CICS only if:
- RECOVERY(ALL) is specified
- RLSACCESS(NO) is specified
- No recovery attributes are defined in the ICF catalog

If you define the recovery attributes for a file in the ICF catalog entry for the corresponding data set, CICS always uses the ICF catalog recovery attributes and ignores those in the file definition. You can alter the recovery attributes defined in the ICF catalog by using the IDCAMS ALTER command. This is not prevented while there are ACBs open for a data set. However, if you change the recovery attributes, be aware of the possible effect on data integrity.

CICS takes a copy of the recovery attributes for a data set from the ICF catalog on the first open-for-update in a sequence of open requests for a data set. This means that a single CICS region is not affected by an update to recovery attributes. However, if a data set is opened in RLS mode and the attributes on the ICF catalog are modified, a second CICS region could open the same data set for update and copy a different set of attributes, with a risk to data integrity.

If you need to alter recovery attributes defined in the ICF catalog (for example, to change the forward recovery log stream name), quiesce the data set before making any changes. This ensures that the data set cannot be used in RLS mode until you have made the change and unquiesced the data set.

GROUP(*groupname***)**
specifies the group name, which can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and $. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see "How the CSD file is organized—groups and lists" on page 8.

INDEXBUFFERS(1|*value*)
specifies the number of buffers to be used for the index. Use a value in the range 1 through 32767. The minimum value you may specify is the number of strings defined in the STRINGS attribute.

JNLADD(NONE|BEFORE|AFTER|ALL)
specifies the add operations you want recorded on the journal nominated by the JOURNAL attribute. Possible values are:

AFTER
Journal the file control write operation after the VSAM I/O operation.

ALL
Journal the file control write operation both before and after the VSAM I/O operation has completed.

BEFORE
Journal the file control write operation before the VSAM I/O operation.

NONE
Do not journal add operations.

JNLREAD(NONE|UPDATEONLY|READONLY|ALL)
specifies the read operations you want recorded on the journal nominated by the JOURNAL attribute. Possible values are:

ALL
Journal all read operations.

NONE
Do not journal read operations.

READONLY
Journal only READ ONLY operations (not READ UPDATE operations).

UPDATEONLY
Journal only READ UPDATE operations (not READ ONLY operations).

JNLSYNCREAD(NO|YES)
specifies whether you want the automatic journaling records, written for READ operations to the journal specified by JOURNAL, to be written synchronously or asynchronously.

JNLSYNCWRITE(YES|NO)
specifies whether you want the automatic journaling records, written for WRITE operations to the journal specified by JOURNAL, to be written synchronously or asynchronously.

JNLUPDATE(NO|YES)
specifies whether you want REWRITE and DELETE operations recorded on the journal nominated by the JOURNAL attribute.

JOURNAL(NO|*number*)
specifies whether you want automatic journaling for this file. The journaled data is in the format of the VSAM record and is used for user controlled journaling.

The data to be journaled is identified by the JNLADD, JNLREAD, JNLSYNCREAD, JNLSYNCWRITE, and JNLUPDATE attributes.

For a CICS-maintained data table, journaling is performed only for requests that result in VSAM I/O requests.

For a user-maintained data table or a coupling facility data table journaling is not performed for any file control operations. However, although automatic journaling for these tables is not supported, if you specify a journal number, CICS tries to open the log stream for the specified journal when opening the file.

Note: Automatic journaling is independent of logging to the system and forward recovery logs, as specified by the RECOVERY and FWDRECOVLOG attributes.

Possible values are:

NO No automatic journaling is to take place for this file.

number
The number that identifies the journal that CICS is to use for the autojournal. CICS journal names are of the form DFHJ*nn*, where *nn* is in the range 1 through 99.

Note: Unlike in releases prior to CICS Transaction Server for OS/390 Release 1, DFHJ01 is **not** the system log.

KEYLENGTH(*value***)**
specifies the length in bytes of the logical key of records in remote files, and in coupling facility data tables that are specified with LOAD(NO).

If not defined here, the length option must be specified on file control API commands in the application programs that refer to this file.

Remote files
The range for key length is 1 through 255.

Coupling facility data tables
The range for key length is 1 through 16. Key length is required only if LOAD(NO) is specified.

You can, optionally, specify a key length for coupling facility data tables specified with LOAD(YES), in which case you should be aware of the following:

- The key length is obtained from the ICF catalog entry for the data set from which the table is loaded. If you specify a key length, the key length must match that specified for the source data set, otherwise attempts to open the file fail with an error message.
- If, when opening the file, CICS finds that the CFDT has already been created, and the key length is different from that used when loading the data set, the open fails.

If you specify a key length for a file that is not a remote file, or does not refer to a CFDT, it has no effect unless the file is redefined, either as a remote file or to reference a CFDT. Note, however, that if you specify a key length, the value returned by an INQUIRE FILE command is as follows:

- If the file is open, CICS returns the value obtained from VSAM, which can be different from that specified on the file definition.
- If the file is closed, CICS returns the value specified on the file definition.

The value for this attribute must be the same throughout the sysplex in all file definitions that reference the same coupling facility data table.

LOAD(NO | YES)
specifies whether the coupling facility data table is to be loaded from a source data set when first opened.

NO Means the coupling facility data table does not require loading from a source data set; it is fully usable by application programs as soon as it is open. The table is loaded by the application programs that use it, which is the default method for a coupling facility data table.

YES Means the coupling facility data table has to be loaded from a source data set before it is fully usable; the application programs that use this coupling facility data table rely on it containing the records from a source data set. Loading does not have to be completed before data can accessed.

This parameter is meaningful only for files defined with the TABLE(CF) attribute. You can specify the LOAD attribute for a file that is not defined as TABLE(CF), but CICS ignores it. (CICS-maintained and user-maintained tables are loaded automatically always from a source data set.) If you subsequently alter the file definition to reference a coupling facility data table, the LOAD attribute comes into effect.

Ensure that the value for this attribute is the same throughout the sysplex in all file definitions that reference the same coupling facility data table.

For more information about using this parameter, see "Coupling facility data tables" on page 190.

LSRPOOLID(1–8 | NONE)
specifies the identity of the local shared resource pool. The default value for LSRPOOLID is 1, unless a value has been specified for the NSRGROUP attribute, in which case the default value for LSRPOOLID is NONE.

NONE
Specifies that the data set associated with this file uses VSAM nonshared resources (NSR).

Note: You cannot specify NONE for a CICS shared data table (CICS- or user-maintained): both these types of data table must use an LSR pool. However, this restriction does not apply to a coupling facility data table, for which you can specify NONE.

1–8 The value, in the range 1 through 8, identifies the number of the VSAM shared resource pool that is used by the VSAM data set associated with this file. The data set is defined as using VSAM local shared resources (LSR). You are recommended to define the buffers, strings, and other resources explicitly in an LSRPOOL resource definition that corresponds to this LSRPOOLID.

By default, if the file definition specifies RLSACCESS(YES), the LSRPOOLID is ignored when CICS opens the file. However, if you change a file definition that specifies an LSR pool from RLSACCESS(NO) to RLSACCESS(YES), you are recommended to keep the LSRPOOLID. This ensures that, if the file is switched at any time from RLS to LSR mode, the file correctly references an LSR pool. For more information, see the description of the RLSTOLSR system initialization parameter in the *CICS System Definition Guide*.

MAXNUMRECS(<u>NOLIMIT</u> | *number*)
specifies the maximum number of records (entries) to be accommodated in the data table. You can use this parameter to prevent a runaway transaction from using:

- All the storage in the server’s pool if the table is a coupling facility data table
- All the storage in the MVS data space if the table is a CICS- or user-maintained table.

This parameter is meaningful only for files defined with ‘CICS’, ‘USER’, or ‘CF’ for the TABLE attribute. You can specify MAXNUMRECS for a file that is defined with TABLE(NO), but it has no effect. If you subsequently alter the file definition to reference a data table, the MAXNUMRECS value comes into effect.

<u>NOLIMIT</u>
There is no user-specified limit placed on the number of records that can be stored in the table. CICS imposes a limit of 2 147 483 647, the maximum fullword value.

number
Specifies the maximum number of records allowed in the table, in the range 1 through 99 999 999.

If you are setting a limit for a recoverable coupling facility data table, specify a value that is about 5 to 10% greater than the maximum number of records the table is expected to contain. This is to allow for additional records that may be created internally for processing recoverable requests. The margin you allow for this internal processing depends on the level of use of the coupling facility data table, and the nature of that use. An effect of this internal processing is that the NOSPACE condition with a RESP2 of 102 can be raised on a WRITE or REWRITE request to a recoverable CF data table that apparently has fewer records than the MAXNUMRECS limit that has been specified for it.

NSRGROUP(*value*)
specifies a symbolic name (up to eight characters) to group together file definitions that refer to the same VSAM base data set. The value is purely symbolic and need not refer to any particular file definition. It is merely necessary that all file definitions that need to be grouped together have the same name. You do not have to specify this attribute to ensure correct processing, but if you do not provide it, performance of your system may be degraded.

The NSRGROUP attribute takes effect only for files referencing data sets that use VSAM nonshared resources. The NSRGROUP parameter must not be coded for a data table. It is associated with the VSAM concept of data set name sharing which causes VSAM to create a single control block structure for the strings and buffers required by all the files that relate to the same base data set.

When the first member of such a group of files is opened, the total number of strings to be allocated for all file entries in the group must be specified to VSAM (by means of the BSTRNO value in the Access Control Block). The VSAM control block structure is built this time regardless of whether the first file to be opened is associated with a path or base data set. The value of

BSTRNO is calculated at this time by adding together the STRINGS values in all the file definitions with the same NSRGROUP attribute. After the first file in the group is opened, any new files added to the group do not affect the VSAM control block structure already built. This would change only if all the files open against the base were closed and then re-opened.

Data set name sharing is forced by CICS as the default for all VSAM files. Data set name sharing is not in effect if a file is opened for read-only processing with DSNSHARING=MODIFYREQS. A file with DSNSHARING=MODIFYREQS still, however, contributes to the BSTRNO calculation.

If a file is using VSAM nonshared resources, and you do not provide an NSRGROUP attribute, the VSAM control block structure may be built with insufficient strings for later processing. When this happens, VSAM invokes the dynamic string addition feature to provide the extra control blocks for the strings as they are required. This mechanism is, however, inefficient and the extra storage is not released until the end of the CICS run.

For files specifying that VSAM local shared resources are to be used (LSRPOOLID=n, where n is in the range 1 to 8), NSRGROUP has no effect.

Figure 25 shows an example of how to specify the required file control definition for a VSAM base data set and alternate index path.

```
CEDA DEFINE FILE(VSAM10B)  GROUP(xxxxxx)
            DSNAME(DTGCAT.VSAM10B)
            DISPOSITION(SHARE)  ADD(YES)
            BROWSE(YES)  DELETE(YES)  READ(YES)
            UPDATE(NO)  RECORDFORMAT(F)
            STRINGS(8)  LSRPOOLID(NONE)
            RECOVERY(NONE)  NSRGROUP(GROUP1)
            INDEXBUFFERS(8)  DATABUFFERS(9)
CEDA DEFINE FILE(VSAM10P)  GROUP(xxxxxx)
            DSNAME(DTGCAT.VSAM10P)
            LSRPOOLID(NONE)  DISPOSITION(SHARE)
            STRINGS(5)  NSRGROUP(GROUP1)
            BROWSE(YES)  DELETE(NO)  READ(YES)
            ADD(NO)  UPDATE(NO)  RECORDFORMAT(F)
            RECOVERY(NONE)  INDEXBUFFERS(5)
            DATABUFFERS(6)
```

Figure 25. VSAM base data set and alternate index path definition.

OPENTIME(<u>FIRSTREF</u>|STARTUP)
specifies when the file is opened. Possible values are:

<u>FIRSTREF</u>
The file remains closed until a request is made to open it by:
- A master terminal command
- An EXEC CICS SET FILE OPEN command in an application program
- An implicit open

STARTUP
The file is opened immediately after CICS initialization by an automatically initiated CICS transaction (CSFU), unless the status of the file is UNENABLED when the file is left closed.

PASSWORD(*name*)
specifies the 1-to 8-character password that is used to verify user access to the file.

CICS masks the password you supply to avoid unauthorized access. You should therefore find a safe way of recording the password.

READ(YES|NO)
specifies whether records on this file can be read.

READINTEG(UNCOMMITTED|CONSISTENT|REPEATABLE)
specifies the level of read integrity required for files defined with RLSACCESS(YES). Read integrity does not apply to non-RLS access mode files, CICS shared data tables, or coupling facility data tables.

You can use READINTEG to set a default level of read integrity for a file. The default level of read integrity is used by programs that do not specify one of the API read integrity options UNCOMMITTED, CONSISTENT, or REPEATABLE on the READ, READNEXT, or READPREV commands. However, if an application program uses one of these explicitly to specify read integrity, the API option overrides any value specified on this READINTEG attribute.

Note: You can specify read integrity options only on CICS file control API commands or in CICS file resource definitions. You cannot use the equivalent parameter on the DD statement for files opened by CICS.

You can specify CONSISTENT or REPEATABLE in a file resource definition, to make read integrity available to programs written before these options were available on the API, and without having to modify those programs. However, if you do this, be aware that enforcing consistent or repeatable reads can introduce unexpected deadlocks. Programs may also encounter the LOCKED condition.

CONSISTENT
The record is read with consistent read integrity. If the record is being modified by another transaction, the READ request waits until the update is complete, the timing of which depends on whether the data set is recoverable or nonrecoverable:

- For a recoverable data set, the READ request completes when the updating transaction completes its next syncpoint or rollback.
- For a nonrecoverable data set, the READ completes as soon as the VSAM request performing the update completes.

CONSISTENT is valid only if you also specify RLSACCESS(YES)—the resource definition is rejected with an error if you specify CONSISTENT for a non-RLS file.

REPEATABLE
The record is read with repeatable read integrity. If the record is being modified by another transaction, the READ request waits until the update is complete, the timing of which depends on whether the data set is recoverable or nonrecoverable:

- For a recoverable data set, the READ request completes when the updating transaction completes its next syncpoint or rollback.
- For a nonrecoverable data set, the READ completes as soon as the VSAM request performing the update completes.

After the read completes, a shared lock remains held until syncpoint. This guarantees that any record read within a unit-of-work cannot be modified while the task makes further read requests. Error responses such as NOTFND may not be repeatable.

REPEATABLE is valid only if you also specify RLSACCESS(YES)—the resource definition is rejected with an error if you specify REPEATABLE for a non-RLS file.

UNCOMMITTED
The record is read without read integrity. CICS obtains the current value of the record as known to VSAM. No attempt is made to serialize this read request with any concurrent update activity for the same record. The record returned may be a version updated by another transaction, but not yet committed, and this record may change if the update is subsequently backed out.

Notes:

1. UNCOMMITTED is the same level of integrity that is provided by those releases of CICS that do not support the READINTEG attribute.
2. Specify UNCOMMITTED for any kind of data table. Any value other than UNCOMMITTED is allowed if RLSACCESS(YES) but is ignored if TABLE(CF), TABLE(CICS), or TABLE(USER) is also specified for the file.

RECORDFORMAT(V|F)
specifies the format of the records on the file.

F The records are fixed length. For VSAM files, specify this only if the VSAM access method services definition specifies fixed size records (that is, the average size is equal to the maximum size), and all the records in the file are of that size.

F is invalid for user-maintained data tables and coupling facility data tables

V The records are variable length. All user-maintained data tables and coupling facility data tables must be specified as variable length. Otherwise, CICS returns an error message stating that RECORDFORMAT(F) conflicts with TABLE(CF) or TABLE(USER) options and is ignored.

RECORDSIZE(*number*)
specifies the maximum length in bytes of records in a remote file or a coupling facility data table. The size specified can be in the range 1 through 32767.

For coupling facility data tables only
This value is required if the file definition for the table specifies LOAD(NO).

You can also specify this attribute if LOAD(YES) is specified (for example, to make it easier for switching the file definition between LOAD(NO) and LOAD(YES)). However, if you specify LOAD(YES), the record size value must match that for the source data set, otherwise CICS fails to open the table. There are three conditions in which CICS can detect an error because of an incorrect record size with LOAD(YES):

1. Before opening the table, CICS verifies that the VSAM-defined record size for the data set from which the coupling facility data

table is to be loaded is the same as the size, if any, in the file definition. If the record size is different, CICS returns error message DFHFC7081.

2. The record size (if specified) on the file definition is the same as that defined to VSAM for the data set, but on opening the table, CICS finds the table is already loaded with data of a different record size. This is probably because the data was loaded from a different data set from the one specified by this file definition. In this case CICS returns error message DFHFC7082.
3. The file definition for the table being opened specifies a record size, but not a data set name because the table is to be loaded by the opening of a different file. If the table has already been created, the open of a file specifying a different record size fails with message DFHFC7083.

To avoid the above errors, ensure the value for this attribute is the same throughout the sysplex in all file definitions that reference the same coupling facility data table, or omit it altogether for files that specify LOAD(YES).

If you specify a record size for a file that is not a remote file, or does not refer to a CFDT, it has no effect unless the file is redefined, either as a remote file or to reference a coupling facility data table. Note, however, that if you specify a record size, the value returned by an INQUIRE FILE command is as follows:

- If the file is open, CICS returns the value obtained from VSAM, which can be different from that specified on the file definition.
- If the file is closed, CICS returns the value specified on the file definition.

Note: For coupling facility data tables,if you can keep the record size to 63 bytes or less, there is a signficant gain in performance as a result of the way records are held in the coupling facility.

RECOVERY(NONE|BACKOUTONLY|ALL)
specifies the type of recovery required for the file.

This attribute is not used for files defined with RLSACCESS(YES), or if the recovery options are defined in the ICF catalog. If LOG is defined in the ICF catalog, CICS ignores the RECOVERY option and takes the LOG value from the ICF catalog, even for files defined with RLSACCESS(NO). If LOG(ALL) is specified in the ICF catalog, CICS also takes the LOGSTREAMID and BWO values from the ICF catalog.

For files that are accessed in RLS mode, you must specify the recovery parameters with the data set definition in the ICF catalog. See the *CICS Recovery and Restart Guide* for more information.)

For coupling facility data tables, the recovery attribute must be the same throughout the sysplex in all file definitions that reference the same coupling facility data table.

ALL Log before images to the system log, and after images to the journal specified in the FWDRECOVLOG attribute. (**Coupling facility data table restriction**:

Records written to the FWDRECOVLOG are independent of any automatic journaling options that may be set.

RECOVERY=ALL together with FWDRECOVLOG provide a means of separating the needs of a forward recovery utility from those of automatic journaling. Additional information, not available via automatic journaling, is recorded on the FWDRECOVLOG. RECOVERY=ALL plus FWDRECOVLOG is the recommended way to provide forward recovery support.

Existing forward recovery utilities that used the JREQ=(WU,WN) and JID=FCT macro settings can still be used with these settings. The RDO equivalents of these automatic journaling settings are JNLADD=BEFORE, JNLUPDATE=YES, and the JOURNAL attribute.

For CICS-maintained data tables, the data table and its source data set are logged, journaled, and recovered together.

For user-maintained tables, specifying ALL has the same effect as specifying BACKOUTONLY: only dynamic backout is provided. There is no forward recovery support for user-maintained tables.

For coupling facility data tables you cannot specify ALL.

Note: When ALL is specified for VSAM ESDS files, CICS is unable to perform backout of ADDs. To cope with this situation, code user exit XFCLDEL to avoid the file being closed because of the error.

BACKOUTONLY
Log before images to the system log.

For CICS-maintained data tables, BACKOUTONLY specifies that the data table and its source data set are recoverable. They are both updated in step and, if required, recovered in step.

For user-maintained tables, this specifies only dynamic backout. No log records are written and, therefore, there is no recovery at emergency restart.

For coupling facility data tables, BACKOUTONLY is permitted only if the coupling facility data table is defined with UPDATEMODEL(LOCKING). You cannot specify this attribute for UPDATEMODEL(CONTENTION). Specifying BACKOUTONLY implies that a coupling facility data table is UOW-recoverable. This means that updates made to the CFDT within a unit of work are backed out if the unit of work fails, or if CICS or the CFDT server fails while the unit of work is in-flight, or if MVS fails. If the table is defined with a source data set, any recovery attributes in the ICF catalog are ignored. For a data table, the recovery attributes are a property of the file not the associated data set.

Note: When BACKOUTONLY is specified for VSAM ESDS files, CICS is unable to perform backout of ADDs. To cope with this situation, code user exit XDBFERR to avoid the file being closed because of the error.

NONE
No recovery logging for this file.

REMOTENAME(*name*)
specifies the name by which this file is known in the system or region in which

it is resident. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command. If REMOTENAME is not specified, the name given in the FILE attribute is used.

REMOTESYSTEM(*name*)
If you are operating in an ISC or MRO environment, and the file is held by a remote system, this specifies the name of the system or region in which the file is resident. The name can be up to four characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command. If you specify REMOTESYSTEM, you may also supply a REMOTENAME, to specify the name of the file in the remote system.

Note: If you modify a resource definition from RLSACCESS(NO) to RLSACCESS(YES), you must remove the remote system name, otherwise CICS will continue to function ship file requests.

RESSECNUM({0|*value*|PUBLIC})
The RESSECNUM attribute is not valid in CICS Transaction Server for OS/390. See “Appendix A. Obsolete attributes retained for compatibility” on page 497 for information.

RLSACCESS(NO|YES)
specifies whether CICS is to open the file in RLS mode.

NO The file is not to be opened in RLS mode. If you specify RLSACCESS(NO) or allow it to default, CICS opens the file in LSR or NSR access mode, depending on the LSRPOOLID attribute. If you also specify LSRPOOLID(NONE), the access mode is NSR; if LSRPOOLID(*number*), the access mode is LSR.

YES The file is to be opened in RLS mode. If you specify RLSACCESS(YES), it takes precedence over the LSRPOOLID attribute, which is ignored when the FILE is opened.

Specifying RLSACCESS(YES) alters the effect of some other attributes defined in the FILE definition, as shown in Table 13.

Table 13. Effects of RLSACCESS(YES) on other FILE attributes

Attribute	Effect of RLSACCESS(YES)
PASSWORD	Ignored.
LSRPOOLID	Ignored.
DSNSHARING	Ignored.
STRINGS	Ignored; RLS access-mode files always have 1024 strings.
REMOTESYSTEM REMOTENAME RECORDSIZE KEYLENGTH	The meanings of these attributes are unchanged. However, dual FILE definitions of local and remote is of less value for RLS access-mode files. With RLS, there may be many file-owning regions (FORs) instead of one, and a local CICS region may have the choice of several FORs.
DATABUFFERS INDEXBUFFERS	Ignored.
TABLE	TABLE(CICS) is not allowed; an error message is issued if specified. TABLE(USER) or TABLE(CF) is allowed. The source data set (if there is one) is accessed in RLS mode to load the data table, after which requests access the data table directly using data table services.
RECOVERY	Ignored. The recovery attribute is obtained from the ICF catalog for an RLS file.

Table 13. Effects of RLSACCESS(YES) on other FILE attributes (continued)

Attribute	Effect of RLSACCESS(YES)
FWDRCOVLOG	Ignored. The forward recovery log stream name is obtained from the ICF catalog for an RLS file.
BACKUPTYPE	Ignored. The type of backup is determined by the DFSMSdss™ backup utility for RLS.

Notes:

1. As long as a file is opened in RLS mode, any values specified for PASSWORD, LSRPOOLID, DSNSHARING, STRINGS, DATABUFFERS, and INDEXBUFFERS are ignored, as described in Table 13 on page 205. However, if you use a CEMT, or EXEC CICS, SET FILE command to change the RLSACCESS parameter from YES to NO, these values are no longer ignored, and CICS uses them when the file is closed and re-opened in non-RLS mode.
2. CICS always takes the RLS access mode from the file resource definition and you cannot override this using the RLS=NRI or RLS=CR parameter on a DD statement.

STATUS({ENABLED | DISABLED | UNENABLED})
specifies the initial status of the file following a CICS initialization with START=COLD or START=INITIAL. You can change the status of a closed file with the master terminal transaction CEMT. The status of a file (enabled, disabled, or unenabled) following a CICS restart is recovered to its status at the previous shutdown.

DISABLED
Any request against this file from a command-level application program causes the DISABLED condition to be passed to the program.

ENABLED
Normal processing is allowed against this file.

UNENABLED
This prevents the file being opened by an implicit open from an application program. Any such attempt to access the file raises the NOTOPEN condition. By contrast, an explicit request to open the file (for example, a CEMT or EXEC CICS SET FILE OPEN command) changes the status to ENABLED before attempting to open the file.

STRINGS(1 | *value*)
specifies the number, in the range 1 through 255, of concurrent requests that can be processed against the file. When the number of requests reaches this value, CICS queues any additional requests until one of the active requests terminates. This applies both to files using shared resources, and to those not using shared resources. Note that if the file definition specifies RLSACCESS(YES), the strings is ignored; you always get 1024 strings with RLS mode access.

For files using local shared resources, this number is not used by VSAM. It is used by CICS, not only as described above, but also to calculate the default value in the buffer pool definition.

Notes:

1. When choosing a STRINGS value, be aware that a proportion (20%) of the specified number of strings is reserved by CICS for use in read-only requests

2. When choosing a STRINGS value for an ESDS, consider the following:
   - If an ESDS is used as an 'add-only' file (that is, it is used only in write mode), a string number of 1 is strongly recommended. Any string number greater than 1 can significantly affect performance, because of exclusive control conflicts that occur when more than one task attempts to write to the ESDS at the same time.
   - If an ESDS is used for both writing and reading, with writing being 80% of the activity, it is better to define two file definitions—using one file for writing and the other for reading.
3. For user-maintained data tables and coupling facility data tables, the STRINGS value does *not* limit the number of concurrent requests against the table. However, the value does limit the number of concurrent requests during the loading of a user-maintained table.

TABLE(NO | CICS | USER | CF)
specifies the type of data table that you require.

CF A coupling facility data table (CFDT). This remains independent of its source data set, and changes to the table are not reflected in the corresponding source data set, if there is one. A source data set is optional for a CFDT, and is specified by LOAD(YES) on the file definition.

If you specify CF, also specify:
- CFDTPOOL, to give the name of the coupling facility pool in which the table resides
- LOAD, to specify whether or not the table is to be loaded from a source data set (or let this default to NO)
- UPDATEMODEL to specify whether the table is to use the CONTENTION or the LOCKING update model (or let this default to LOCKING)
- RECORDFORMAT as V (or let this default to V)
- MAXNUMRECS with the value you require.

A coupling facility data table requires a coupling facility data table server. For information on how to start a coupling facility data table server, see *CICS System Definition Guide*.

CICS A CICS-maintained data table. This automatically reflects all modifications made to the table in its source data set. If you specify CICS, also specify:
- LSRPOOLID with a value of 1 through 8
- MAXNUMRECS with the value you require.

NO Data table not required.

USER A user-maintained table. This remains independent of its source data set, and changes to the user-maintained table are not reflected in corresponding source data set. If you specify USER, also specify:
- LSRPOOLID with a value of 1 through 8
- RECORDFORMAT as V (or let this default to V)
- MAXNUMRECS with the value you require.

TABLENAME(*name*)
specifies the name of the coupling facility data table that is accessed through this file definition. The name can be up to 8 characters in length, from the

allowable characters of: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase when the definition is written to the CSD.

If you omit this attribute when TABLE(CF) is specified, it defaults to the name specified for the FILE. To enable CICS regions to share a coupling facility data table, the file definitions installed in each region must specify the same CFDTPOOL name and TABLENAME (or FILE name when TABLENAME is not used). The TABLENAME need only be unique within its pool.

Note that the table name is not only an identifier for the table, but is also used as the resource name in security checks.

This parameter is meaningful only for files defined with the TABLE(CF) attribute. You can specify a table name for a file that is not defined as TABLE(CF), but CICS ignores it. If you subsequently alter the file definition to reference a coupling facility data table, the TABLENAME attribute comes into effect.

UPDATE(NO|YES)
specifies whether records on this file can be updated.

UPDATEMODEL(LOCKING|CONTENTION)
specifies the type of update model to be used for a coupling facility data table.

LOCKING
means that the CFDT is updated using the locking model, in which records are locked when they are read for update, so that they cannot be changed by any other units of work until the update request has been completed (at syncpoint, or by a REWRITE, DELETE, or UNLOCK command for non-recoverable tables) LOCKING is the default for a file that specifies TABLE(CF). With the LOCKING model, the CFDT can be defined as:

- **Non-recoverable**, meaning that CFDT updates are not backed out if a unit of work fails, and the locks are only held for the duration of a request. You specify that a CFDT is not recoverable by specifying RECOVERY(NONE).
- **Recoverable**, or UOW-recoverable, meaning that updates made to the CFDT within a unit of work are backed out if the unit of work fails, or if CICS or the CFDT server fails while the unit of work is in-flight, or if MVS fails. You specify that a CFDT is recoverable by specifying RECOVERY(BACKOUTONLY).

A recoverable CFDT that uses the locking model is very similar to a recoverable file or data set, except that it does not survive a failure of the coupling facility in which it resides. There is no forward recovery for a coupling facility data table.

CONTENTION
means that the CFDT is updated using the contention model, in which records are *not* locked when they are read for update. An error is returned on a subsequent REWRITE or DELETE if the record was changed or deleted by another task after it was read for update. The CFDT must be non-recoverable (RECOVERY(NONE)), meaning that updates are not backed out if a unit of work fails.

The value for this attribute must be the same throughout the sysplex in all file definitions that reference the same coupling facility data table.

This parameter is meaningful only for files defined with the TABLE(CF) attribute. You can specify the update model for a file that is not defined as TABLE(CF), but CICS ignores it. If you subsequently alter the file definition to reference a coupling facility data table, the UPDATEMODEL attribute comes into effect.

JOURNALMODEL

A JOURNALMODEL resource definition provides the connection between a CICS journal name (or identifier) and the associated physical log streams managed by the MVS system logger, or between the journal name and the SMF log.

Although they are intended mainly for user journals, you can also define journal models for the system log and forward recovery logs (non-RLS only). However, for forward recovery logs, you are recommended to define all log stream names for forward recovery in the VSAM catalog. This is mandatory for VSAM files processed in RLS mode, but optional for non-RLS mode files.

Unlike the journal control table, you do not need to define a journal model for every journal that CICS uses. Instead, define some generic model definitions that describe the mapping to log stream names for the majority of your CICS journals. You may find that you can use the default models supplied by CICS and need not define any of your own. In addition to generic models, you can define the necessary specific models where special handling is required (for example, SMF logging, or merging with other log streams).

You can change JOURNALMODEL definitions at any time, but any journal entries that CICS has already created using model definition cannot reflect the change unless you first delete the existing entry using a DISCARD JOURNALNAME() command.

Compatibility note: For API compatibility with releases earlier than CICS Transaction Server for OS/390, CICS continues to support numeric journal identifiers in the range 01 through 99, for the following purposes:

- For file control autojournaling, as specified in FILE resource definitions (or on DFHFCT macro entries)
- For terminal control autojournaling, as specified in PROFILE resource definitions
- For forward recovery logging, as specified in FILE resource definitions
- For user journaling using API journal commands, such as the EXEC CICS WRITE JOURNALNUM command.

Defining a JOURNALMODEL

```
Journalmodel ==> ........
Group        ==> ........
Description  ==> ............................................
Journalname  ==> ........          name | JOURNALMODEL name
Type         ==> MVS               MVS | SMF | DUMMY
Streamname   ==>                   stream_name_template|&USERID..&APPLID..&JNAME.
```

Figure 26. The DEFINE panel for JOURNALMODEL

Options

DESCRIPTION(*text*)
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use

parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a group name. The resource definition becomes a member of the group, and is installed in the CICS system when the group is installed.

The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

JOURNALMODEL(*name*)
specifies the name of this JOURNALMODEL definition.

The journal model name is used to refer to a specific JOURNALMODEL definition in the CSD file—it does not have to correspond to a CICS journal name. However, the JOURNALMODEL name is also used as the JOURNALNAME if you omit the JOURNALNAME attribute.

The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

JOURNALNAME(*name*)
specifies the journal names to which this definition applies. If you omit the JOURNALNAME attribute, the name you specify on the JOURNALMODEL attribute is used as the journal name. Name can be either the specific name of a journal or a generic name, although using a generic name for system log and log-of-logs models does not serve much purpose.

| The name can be up to eight characters in length. The acceptable characters
| are: A-Z 0-9 $ @ % + and #. Lowercase characters are converted to uppercase
| except when using the CREATE command.

The forms of the names you can define are as follows:

For system logs
To define a JOURNALMODEL for system logs, specify the name as DFHLOG for the primary system log stream, and as DFHSHUNT for the secondary log stream. Install one journal model only for each of these log streams in a CICS region.

CICS-supplied definitions for DFHLOG, DFHSHUNT, and DFHLGLOG are contained in group DFHLGMOD in DFHLIST.

For log-of-logs
To define a JOURNALMODEL for the log-of-logs, specify the name as DFHLGLOG. See the *CICS System Definition Guide* for more information about the purpose of the log of logs.

For autojournals
For autojournals (file control and terminal control), the name must be of the form DFHJ*nn* where *nn* is a number in the range 1 through 99. The name can be either the specific name of a journal or a generic name.

For user journals
For user journals, the name can be up to 8 characters, and can be either the specific name of a journal or a generic name. If compatibility with releases earlier than CICS Transaction Server for OS/390, is required, the name must be of the form DFHJ*nn* where *nn* is a number in the range 1 through 99.

For forward recovery logs (non-RLS)
For non-RLS forward recovery logs, the name must be of the form DFHJ*nn* where *nn* is a number in the range 1 through 99. The name can be either the specific name of a journal or a generic name.

Note: You cannot define a journal model for use with VSAM RLS forward recovery logs. CICS obtains the fully-qualified LSN directly from the VSAM catalog, and therefore does not need a journal model to obtain the LSN.

You define generic names, using the special symbols %, +, and *, as follows:

- You can use the % or + symbols to represent any single character within a journal name.
- You can use the * symbol at end of a name to represent any number of characters. A default name of a single * is used to match any journal names that do match on a more specific name.

If there are several installed JOURNALMODEL definitions that match a journal name, CICS chooses the best match as follows:

1. If there is a JOURNALMODEL with a specific JOURNALNAME that exactly matches, CICS uses this model.
2. If there is no exact match, the journal name is compared with the matching generic entries and the most specific entry is used.

   In comparing names to see which one is more specific, the names are compared character by character. Where they first differ:
   - If one has a discrete character (not %, +, or *) and the other has a generic character (%, +, or *) the one with the discrete character is used.
   - If one has a % or a + and the other has a *, the one with % or + is used.
3. If there are duplicate JOURNALMODEL definitions (that is, definitions with the same JOURNALNAME), CICS uses the last one processed.

Attention: Take care when defining a completely generic journal name using only the single asterisk (*). This is particularly important if you have not defined a specific journal model for the system log (using journal name DFHLOG), and the log stream name is a fully-qualified literal name. If you define a journal model with JOURNALNAME(*) and do not define a journal model for the system log, CICS uses the log stream name defined on the generic model definition. This causes problems if other journals and forward recovery logs are assigned to the same log stream by means of the generic journal model.

STREAMNAME({<u>&USERID..&APPLID..&JNAME.</u> | stream_name_template})
specifies either an explicit MVS system logger log stream name, or a template used to construct the log stream name. STREAMNAME is applicable only to journal models defined as TYPE(MVS).

The 4 symbolic names, from which you can use a maximum of 3, are as follows:

&USERID.
The symbolic name for the CICS region userid, which can be up to eight characters. If the region does not have a userid, the string 'CICS' will be used.

&APPLID.
The symbolic name for the CICS region APPLID as specified on the system initialization parameter, and which can be up to eight characters.

Note: If you are using XRF and you specify the APPLID system initialization parameter as APPLID=(generic_applid,specific_applid), it is the generic applid that CICS uses when resolving &APPLID..

&JNAME.
The symbolic name for a journal name that references, either by a specific or generic match, this journal model definition. &JNAME. can be up to eight characters in length.

&SYSID.
The symbolic name for the CICS region SYSID as specified on the SYSIDNT system initialization parameter. If SYSIDNT is not specified, the string 'CICS' will be used.

The default set of symbolic names is: &USERID..&APPLID..&JNAME.

For Example: &USERID..&APPLID..&JNAME. =

CICSHA##.CICSHAA1.DFHJ02

where:

CICSHA##
is the CICS region userid used by all the AORs.

CICSHAA1
is the applid of one AOR.

DFHJ02
is the journal name of an auto journal.

An alternative set of symbolic names could be:

&SYSID..&APPLID..&JNAME. =
SYSA.CICSHAA1.DFHJ02

where:

SYSA is the character string as specified by the SYSIDNT system initialization parameter.

CICSHAA1
is the applid of one AOR.

DFHJ02
is the journal name of an auto journal.

CICS installs the JOURNALMODEL resource as defined, including the symbolic names.

stream_name_template

A log stream name can be either an unqualified name or a qualified name, as defined for MVS data set names:

Unqualified name: 1 through 8 alphanumeric or national characters ($ # @), or a hyphen. The first character of the name must be alphabetic or national (A-Z $ # @).

Qualified name: Multiple names joined by periods, up to a maximum of 26 characters. Each name in a qualified name must follow the rules for an unqualified name, with each qualified name (except the last) followed by a period. For example,

```
name_1.name_2...name_n
```

where the number of names is restricted by the 26-character limit.

For more information about the rules for qualified and unqualified data set names, see the *OS/390 MVS JCL Reference* manual.

You can construct log stream names consisting of a mixture of specific characters (from within the allowed set), and symbolic names for substitution. After substitution, the name must meet the rules for qualified and unqualified log stream names, and must not exceed 26 characters, including periods. Thus, if each name in a qualified name uses the maximum of eight characters, you are restricted to three names only, with the first and second names, and the second and third names separated by a period. For example:

```
CICSDA##.CICSDAA1.FWDRECOV
```

for a forward recovery log stream. The log stream name is determined by symbolic substitution when a journal name is first resolved to a JOURNALMODEL definition.

By specifying the same log stream name for multiple CICS general logs, you can merge the log streams from different CICS regions. However, you cannot merge general log streams with the CICS system log, nor can you merge system logs from different CICS regions.

When merging log streams from different CICS systems, the log data blocks are written to their log streams in strict MVS system logger time-stamp sequence. However, the individual records from different CICS regions may not be in strict time-stamp sequence across different blocks

CICS log streams should not be merged with log streams generated by other products unless any programs that read the log stream are prepared to handle the formats.

Security note: When you have defined a log stream name to CICS and the MVS system logger, you must ensure that the required security authorizations are defined to RACF (or an equivalent external security manager). This security authorization is necessary before you attempt to bring up a CICS region that references a new log stream. RACF supports the LOGSTRM general resource class for this purpose.

TYPE({DUMMY|MVS|SMF})

specifies where the journal records are to be written. It can be up to five characters, and can have the following values:

DUMMY
No log records are to be written. For example, you can use this to suppress unwanted log records without changing an application, or without changing file or profile resource definitions.

If you do not want a system log or a log-of-logs, specify DUMMY on the JOURNALMODEL definitions for the DFHLOG, DFHSHUNT, and DFHLGLOG, as required.

MVS Records are to be written to an MVS system logger log stream. The name of the log stream is specified in the STREAMNAME attribute.

SMF Journal records are to be written in SMF format to the MVS SMF log instead of to an MVS system logger log stream.

Note: SMF is not allowed for the CICS system log or for forward recovery logs.

Examples

Given the following set of definitions for a CICS AOR with applid CICSHAA3 and region userid CICSHA##:

```
1.  DEFINE JOURNALMODEL(USERJNL8) GROUP(LOGS)
           JOURNALNAME(DFHJ08)
           TYPE(SMF)
```

Records written to autojournal 08 or user journal 08 using an EXEC CICS WRITE JOURNALNAME(DFHJ08)... or EXEC CICS WRITE JOURNALNUM(08)... command are written in SMF format to the MVS SMF data set.

```
2.  DEFINE JOURNALMODEL(USERJNL9) GROUP(LOGS)
           JOURNALNAME(DFHJ09)
           TYPE(DUMMY)
```

Records written to autojournal 09 or user journal 09 using an EXEC CICS WRITE JOURNALNAME(DFHJ09)... or EXEC CICS WRITE JOURNALNUM(09)... command are not written to any log stream, though the application program receives a normal response.

```
3.  DEFINE JOURNALMODEL(UJ10TO19) GROUP(LOGS)
           JOURNALNAME(DFHJ1%)
           STREAMNAME(&USERID..MERGED.USRJRNLS)
           TYPE(MVS)
```

Records written to user journals 10–19 (DFHJ10–DFHJ19) are merged together on log stream CICSHA##.MERGED.USRJRNLS, together with records from any other CICS regions running under the CICSHA## userid and with the same JOURNALMODELs installed.

```
4.  DEFINE JOURNALMODEL(LOGOFLOG) GROUP(LOGS)
           JOURNALNAME(DFHLGLOG)
           STREAMNAME(CICSVR.SHARED.DFHLGLOG)
           TYPE(MVS)
```

File tie-up records and other records written by file control and the CICS log manager to journal DFHLGLOG for use by forward recovery products, such as CICSVR, are written to a shared log stream CICSVR.SHARED.DFHLGLOG. This log stream is shared by all the CICS regions in the sysplex in which this JOURNALMODEL resource definition is installed.

```
5. DEFINE JOURNALMODEL(JNLMODL1) GROUP(LOGS)
          JOURNALNAME(USERJNL*)
          STREAMNAME(&USERID..ANYCORP.&JNAME..UK)
          TYPE(MVS)
```

Records written to any user journals or autojournals that begin with the letters USERJNL are merged together on a log stream with a name that is obtained by substituting the CICS region userid for &USERID. and the journal name for &JNAME..

With only the above examples installed, other forms of journaling, such as terminal control automatic message journaling defined with PROFILE ... JOURNAL(25), use the default entry (shown below as JOURNALMODEL(OTHERS) with records written to log stream CICSHA##.CICSHAA3.DFHJ25.

CICS default JOURNALMODEL: If CICS cannot find an installed JOURNALMODEL definition, it assumes the attributes of the following "built-in" default definition:

```
DEFINE JOURNALMODEL(OTHERS) GROUP(LOGS)
         JOURNALNAME(*)
         STREAMNAME(&USERID..&APPLID..&JNAME.)
         TYPE(MVS)
```

JOURNALNAME(*) is the default journal model that CICS uses if there is no matching JOURNALMODEL entry for a journal name.

LSRPOOL

The local shared resources (LSR) pool is a reserve of data buffers, strings, and Hiperspace [3] buffers that VSAM uses when processing access requests for certain files. You use the LSRPOOL definition to define the size and characteristics of the pool. Up to eight LSR pools can be defined concurrently in the system, each identified by their LSRPOOLID. This LSRPOOLID is used to associate a FILE with an LSR pool if that file is to use shared resources.

When the LSRPOOL definition is installed in the active system, its information is stored and used when the pool with the specified ID is next built. A pool is built when the first file using a particular LSR pool is opened, and is dynamically deallocated only when no files are currently open against that pool. This means that when an LSRPOOL definition is installed into the system it may not take effect immediately.

CICS sets default attributes if an LSRPOOL is not defined, but you are advised to define the LSRPOOL anyway, for reasons of performance. In a production system, for example, delay may be incurred while pool requirements are being calculated by CICS. Another possible problem is that if files in the FCT are not allocated at the time the pool is built, the data set names will not be known to CICS. In this case, the pool is built based on the information available, but the subsequent performance of the system may suffer or files may fail to open.

You can associate the CSD file with a particular LSRPOOL by specifying the CSDLSRNO system initialization parameter. The default is pool 1; ensure that sufficient buffers of an appropriate size are provided to permit the CSD file to be used by CICS. See the *CICS System Definition Guide* for further information about CSDLSRNO and about calculating the buffer requirements for the CSD file.

3. A Hiperspace buffer is a high-performance storage area in the MVS image. This area is used for reading and writing 4KB pages. The type of Hiperspace used by VSAM resides entirely in expanded storage, which is additional processor storage used only for paging to and from real storage.

Defining an LSRPOOL

```
 Lsrpool      ==> ....
 Group        ==> ....
 DEscription  ==>

 Lsrpoolid    ==> 1                  1 - 8
 Maxkeylength ==>                    0 - 255
 SHarelimit   ==>                    1 - 100
 STrings      ==>                    1 - 255

DATA BUFFERS
 DATA512      ==>                    3 - 32767
 DATA1K       ==>                    3 - 32767
 DATA2K       ==>                    3 - 32767
 DATA4k       ==>                    3 - 32767
 DATA8k       ==>                    3 - 32767
 DATA12k      ==>                    3 - 32767
 DATA16k      ==>                    3 - 32767
 DATA20k      ==>                    3 - 32767
 DATA24k      ==>                    3 - 32767
 DATA28k      ==>                    3 - 32767
 DATA32k      ==>                    3 - 32767

INDEX BUFFERS
 INDEX512     ==>                    3 - 32767
 INDEX1K      ==>                    3 - 32767
 INDEX2K      ==>                    3 - 32767
 INDEX4k      ==>                    3 - 32767
 INDEX8k      ==>                    3 - 32767
 INDEX12k     ==>                    3 - 32767
 INDEX16k     ==>                    3 - 32767
 INDEX20k     ==>                    3 - 32767
 INDEX24k     ==>                    3 - 32767
 INDEX28k     ==>                    3 - 32767
 INDEX32k     ==>                    3 - 32767

HIPERSPACE DATA BUFFERS
 HSDATA4K     ==>                    0 - 16777215
 HSDATA8K     ==>                    0 - 16777215
 HSDATA12K    ==>                    0 - 16777215
 HSDATA16K    ==>                    0 - 16777215
 HSDATA20K    ==>                    0 - 16777215
 HSDATA24K    ==>                    0 - 16777215
 HSDATA28K    ==>                    0 - 16777215
 HSDATA32K    ==>                    0 - 16777215

HIPERSPACE INDEX BUFFERS
 HSINDEX4K    ==>                    0 - 16777215
 HSINDEX8K    ==>                    0 - 16777215
 HSINDEX12K   ==>                    0 - 16777215
 HSINDEX16K   ==>                    0 - 16777215
 HSINDEX20K   ==>                    0 - 16777215
 HSINDEX24K   ==>                    0 - 16777215
 HSINDEX28K   ==>                    0 - 16777215
 HSINDEX32K ==>                      0 - 16777215
```

Figure 27. The DEFINE panel for LSRPOOL

Options

DATA512(*value*)
: specifies the number, in the range 3 through 32767, of 512-byte data buffers you require.

DATA1K(*value*)
: specifies the number, in the range 3 through 32767, of 1KB data buffers you require.

DATA2K(*value*)
specifies the number, in the range 3 through 32767, of 2KB data buffers you require.

DATA4K(*value*)
specifies the number, in the range 3 through 32767, of 4KB data buffers you require.

DATA8K(*value*)
specifies the number, in the range 3 through 32767, of 8KB data buffers you require.

DATA12K(*value*)
specifies the number, in the range 3 through 32767, of 12KB data buffers you require.

DATA16K(*value*)
specifies the number, in the range 3 through 32767, of 16KB data buffers you require.

DATA20K(*value*)
specifies the number, in the range 3 through 32767, of 20KB data buffers you require.

DATA24K(*value*)
specifies the number, in the range 3 through 32767, of 24KB data buffers you require.

DATA28K(*value*)
specifies the number, in the range 3 through 32767, of 28KB data buffers you require.

DATA32K(*value*)
specifies the number, in the range 3 through 32767, of 32KB data buffers you require.

DESCRIPTION(*text*)
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see "How the CSD file is organized—groups and lists" on page 8.

The GROUP name can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and $. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

HSDATA4K(*value*)
specifies the number, in the range 0 through 16777215, of 4KB Hiperspace data buffers you require.

HSDATA8K(*value*)
specifies the number, in the range 0 through 16777215, of 8KB Hiperspace data buffers you require.

HSDATA12K(*value*)
specifies the number, in the range 0 through 16777215, of 12KB Hiperspace data buffers you require.

HSDATA16K(*value*)
specifies the number, in the range 0 through 16777215, of 16KB Hiperspace data buffers you require.

HSDATA20K(*value*)
specifies the number, in the range 0 through 16777215, of 20KB Hiperspace data buffers you require.

HSDATA24K(*value*)
specifies the number, in the range 0 through 16777215, of 24KB Hiperspace data buffers you require.

HSDATA28K(*value*)
specifies the number, in the range 0 through 16777215, of 28KB Hiperspace data buffers you require.

HSDATA32K(*value*)
specifies the number, in the range 0 through 16777215, of 32KB Hiperspace data buffers you require.

HSINDEX4K(*value*)
specifies the number, in the range 0 through 16777215, of 4KB Hiperspace index buffers you require.

HSINDEX8K(*value*)
specifies the number, in the range 0 through 16777215, of 8KB Hiperspace index buffers you require.

HSINDEX12K(*value*)
specifies the number, in the range 0 through 16777215, of 12KB Hiperspace index buffers you require.

HSINDEX16K(*value*)
specifies the number, in the range 0 through 16777215, of 16KB Hiperspace index buffers you require.

HSINDEX20K(*value*)
specifies the number, in the range 0 through 16777215, of 20KB Hiperspace index buffers you require.

HSINDEX24K(*value*)
specifies the number, in the range 0 through 16777215, of 24KB Hiperspace index buffers you require.

HSINDEX28K(*value*)
specifies the number, in the range 0 through 16777215, of 28KB Hiperspace index buffers you require.

HSINDEX32K(*value*)
specifies the number, in the range 0 through 16777215, of 32KB Hiperspace index buffers you require.

INDEX512(*value*)
specifies the number, in the range 3 through 32767, of 512-byte index buffers you require.

INDEX1K(*value*)
specifies the number, in the range 3 through 32767, of 1KB index buffers you require.

INDEX2K(*value***)**
specifies the number, in the range 3 through 32767, of 2KB index buffers you require.

INDEX4K(*value***)**
specifies the number, in the range 3 through 32767, of 4KB index buffers you require.

INDEX8K(*value***)**
specifies the number, in the range 3 through 32767, of 8KB index buffers you require.

INDEX12K(*value***)**
specifies the number, in the range 3 through 32767, of 12KB index buffers you require.

INDEX16K(*value***)**
specifies the number, in the range 3 through 32767, of 16KB index buffers you require.

INDEX20K(*value***)**
specifies the number, in the range 3 through 32767, of 20KB index buffers you require.

INDEX24K(*value***)**
specifies the number, in the range 3 through 32767, of 24KB index buffers you require.

INDEX28K(*value***)**
specifies the number, in the range 3 through 32767, of 28KB index buffers you require.

INDEX32K(*value***)**
specifies the number, in the range 3 through 32767, of 32KB index buffers you require.

LSRPOOL(*name***)**
specifies the name of the local shared resource pool being defined. The name may be up to eight characters in length.

If only DATA BUFFERS is specified, one set of buffers is built for the pool to be used for both the index and the data components of a VSAM KSDS data set.

If no data buffers are specified, CICS calculates the buffers required for both data and index components, both components sharing the same set of buffers.

If INDEX BUFFERS is specified, two parts of the pool are built, one for data and the other for index buffers. If you specify INDEX BUFFERS, you must also specify DATA BUFFERS.

LSRPOOLID({1 | *value***})**
specifies the identifier of the local shared resource pool being defined. The value must be in the range 1 through 8.

MAXKEYLENGTH(*value***)**
specifies the maximum key length of any of the files that are to share resources. The value must be in the range 0 through 255. This value overrides part of the CICS resource calculation. If you do not specify it, CICS determines the maximum key length.

SHARELIMIT(*value*)
specifies, as an integer, the percentage of the maximum amount of VSAM resources to be allocated. The number can be any value from 1 through 100.

Specify this if CICS is to calculate the maximum amount of resources required by the VSAM files that are to share resources. Because these resources are to be shared, some percentage of this maximum amount of resources must be allocated. If this attribute is omitted, 50 percent of the maximum amount of resources is allocated.

If both the STRINGS and SIZE attributes are specified, SHARELIMIT has no effect.

STRINGS(*value*)
specifies the limit, in the range 1 through 255, of all the strings of the files in the pool.

MAPSET

Each interactive application using a display device can use specific screen layouts, or maps. These are not specified in the program itself. Instead, you use basic mapping support (BMS). This gives greater flexibility and allows the maps to be used by multiple invocations of the same program, or by several different programs. You specify maps, and the fields on them, using the DFHMSD, DFHMDI, and DFHMDF macros. For further guidance on this, see the *CICS Application Programming Guide*.

Instead of using the BMS map definition macros, you can define maps interactively with the Screen Definition Facility II (SDF II) program product, program numbers 5665-366 (for MVS) and 5664-307 (for VM). SDF II allows you to paint a screen interactively. You can then generate the screen to get the equivalent of a CICS/BMS map set. The test facilities of SDF II also enable you to see your map in its run-time appearance. For background information, see the *Screen Definition Facility II Primer for CICS/BMS Programs* and the *Screen Definition Facility II General Information* manual.

An application can use a series of related maps at different times during the interaction with the user. It can also use several related maps at the same time and on the same display, to build up a complete screen.

These related maps belong to a map set, which you specify in a MAPSET definition. Even if your program has only one map, this must still belong to a map set. You can define your MAPSETs either by using CEDA or DFHCSDUP, or by setting the appropriate system initialization parameters to enable them to be autoinstalled. See “Chapter 11. Autoinstall for other resources” on page 145 for further information about autoinstall.

There is no link through resource definitions between a program and its map sets. Instead, you specify the MAPSET name in the BMS SEND MAP and RECEIVE MAP commands in your program.

Defining a MAPSET

```
Mapset       ==>
 Group        ==>
 Description  ==>
REsident     ==> No               No | Yes
USAge        ==> Normal           Normal | Transient
USElpacopy   ==> No               No | Yes
Status       ==> Enabled          Enabled | Disabled
RSl            : 00               0-24 | Public
```

Figure 28. The DEFINE panel for MAPSET

Options

DESCRIPTION(*text***)**
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

GROUP(*groupname***)**
Every resource definition must have a GROUP name. The resource definition

becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see "How the CSD file is organized—groups and lists" on page 8.

The GROUP name can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and $. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

MAPSET(*name*)
specifies the name of this MAPSET definition. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command. Do not use map set names beginning with DFH, because these characters are reserved for use by CICS.

For a BMS device-dependent map set, the map set name must be derived by adding the map set suffix to the original (1-to 7-character) map set name. The suffix depends on the parameter specified in the TERM operand of the DFHMSD macroinstruction that defined the map set.

To use device-dependent suffixes, you need to specify BMS=(,,,DDS) as a system initialization parameter. For programming information on map set suffixes, see the *CICS Application Programming Reference* manual.

RESIDENT({NO|YES})
specifies the residence status of the map set.

NO The map set is not to be permanently resident.

YES The map set is to be loaded on first reference and is then to be permanently resident in virtual storage, but is to be pageable by the operating system.

RSL
The RSL attribute is not valid in CICS Transaction Server for OS/390. See "Appendix A. Obsolete attributes retained for compatibility" on page 497 for information.

STATUS({ENABLED|DISABLED})
specifies the map set status.

DISABLED
The map set may not be used.

ENABLED
The map set may be used.

USAGE({NORMAL|TRANSIENT})
specifies when the storage for this map set will be released.

NORMAL
When the use count of the map set reaches zero, it will become eligible for removal from storage as part of the normal dynamic storage compression process.

TRANSIENT
When the use count for this map set becomes zero, the storage for this map set is released. This value should be specified for map sets that are referenced infrequently.

USELPACOPY({NO|YES})
specifies whether the map set is to be used from the link pack area (LPA).

NO The map set is not to be used from the LPA. It is loaded into the CICS partition.

YES The map set can be used from the LPA if LPA=YES is specified as a system initialization parameter. The use of the map set from the LPA requires that it has been installed there and that the map set is not named by the PRVMOD start-up option. For further guidance on this, see the *CICS Transaction Server for OS/390 Installation Guide*.

PARTITIONSET

A partition set is a table that describes to CICS how to partition a display screen. Partition sets are created by coding and assembling a series of commands.

The screen areas of some display devices (the 8775 Display Terminal and the IBM 3290 Information Panel, for example), can be divided into **partitions**, which can be treated as several different small displays. Different programs or program steps in a transaction can write to or receive input from different partitions.

You specify the partition set with DFHPSD and DFHPDI macros, described for programming purposes in the *CICS Application Programming Reference* manual.

You specify each different partition configuration as a PARTITIONSET. PARTITIONSET definitions are created using CEDA or DFHCSDUP, or they can be autoinstalled if the appropriate system initialization parameters have been set. See “Chapter 11. Autoinstall for other resources” on page 145 for information.

You can name the PARTITIONSET that you want the transaction to use in the TRANSACTION definition. When the transaction starts, the information is loaded into the internal buffer of the display device.

Alternatively, if you do not specify a PARTITIONSET, CICS sets the display device to its base state before the transaction is initiated.

The transaction may require CICS to load a PARTITIONSET, or change to a new one, by issuing the BMS SEND PARTNSET command. This loads the partition set dynamically, if its definition has been installed in the active CICS system.

Instead of using the BMS partition definition macros, you can define partitions interactively with the Screen Definition Facility II (SDF II) program product, program numbers 5665-366 (for MVS) and 5664-307 (for VM). SDF II allows you to paint a screen interactively. You can then generate the screen to get the equivalent of a CICS/BMS partitionset. The test facilities of SDF II also enable you to see your partition in its run-time appearance. For general information, see the *Screen Definition Facility II Primer for CICS/BMS Programs* and the *Screen Definition Facility II General Information* manual.

Defining a PARTITIONSET

```
PARTItionset ==>
 Group        ==>
 Description  ==>
REsident     ==> No                  No | Yes
USAge        ==> Normal              Normal | Transient
USElpacopy   ==> No                  No | Yes
Status       ==> Enabled             Enabled | Disabled
Rsl            : 00                  0-24 | Public
```

Figure 29. The DEFINE panel for PARTITIONSET

Options

DESCRIPTION(*text*)
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use

parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see “How the CSD file is organized—groups and lists” on page 8.

The GROUP name can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and $. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

PARTITIONSET(*name*)
specifies the name of this PARTITIONSET definition. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command. Do not use partition set names beginning with DFH, because these characters are reserved for use by CICS.

For a device-dependent partition set, the partition set name must be derived by adding the partition set suffix to the original (1- to 6-character) partition set name. The suffix depends on the parameter specified in the SUFFIX operand of the DFHPSD macro instruction that defined the partition set.

To use device-dependent suffixes, you need to specify BMS=(,,,DDS) as a system initialization parameter.

For programming information on partition set suffixes, see the *CICS Application Programming Reference* manual.

RESIDENT({NO|YES})
specifies the residence status of the partition set.

NO The partition set is not to be permanently resident.

YES The partition set is to be loaded on first reference and is then to be permanently resident in virtual storage, but is to be pageable by the operating system.

RSL
The RSL attribute is not valid in CICS Transaction Server for OS/390. See “Appendix A. Obsolete attributes retained for compatibility” on page 497 for information.

STATUS({ENABLED|DISABLED})
specifies the partition set status.

DISABLED
The partition set may not be used.

ENABLED
The partition set may be used.

USAGE({NORMAL|TRANSIENT})
specifies when the storage for this partition set is released.

NORMAL
When the use count for this partition set reaches zero, it becomes eligible for removal from storage as part of the normal dynamic program compression process.

TRANSIENT
When the use count for this partition set becomes zero, the storage for this partition set is released. This value should be specified for partition sets that are referenced infrequently.

USELPACOPY({NO|YES})
specifies whether the partition set is to be used from the link pack area (LPA).

NO The partition set is not to be used from the LPA. It is loaded into the CICS partition.

YES The partition set can be used from the LPA if LPA=YES is specified as a system initialization parameter. The use of the partition set from the LPA requires that it has been installed there and that the partition set is not named by the PRVMOD start-up option. For more details on this, see the *CICS Transaction Server for OS/390 Installation Guide*.

PARTNER

You use the PARTNER definition to enable CICS application programs to communicate via APPC protocols with a partner application program running on a remote logical unit. This interaction between a CICS application program and a partner application program is called a **conversation**.

The PARTNER definition also facilitates the use of the call to the interface with the communications element of the System Application Architecture (SAA). For more information on the SAA communications interface, see the *Common Programming Interface Communications Reference* manual.

To allow the SAA communications interface to be used, you must specify the following resources:
- A PROFILE definition (see "PROFILE" on page 235)
- A CONNECTION definition (see "CONNECTION" on page 157)
- A SESSIONS definition (see "SESSIONS" on page 253)

You can define your CICS partner information in one of two ways:
- Create a PARTNER definition
- In an application program, by setting SYMDESTNAME to a null value, and issuing the appropriate CPI SET calls. See the *CPI-C Communications Reference* manual for further details.

PARTNER uses the following attributes:

NETNAME
: is the name of the logical unit on which the partner application program is running.

PROFILE
: is the name of the communication profile that specifies the interaction between transactions and the logical units involved.

What happens to PARTNER definitions

When you install a PARTNER definition, CICS attempts to resolve references to CONNECTION and PROFILE definitions. CICS then creates an entry for the PARTNER definition in the **partner resource table (PRT)**. See the *CICS Intercommunication Guide* for more information.

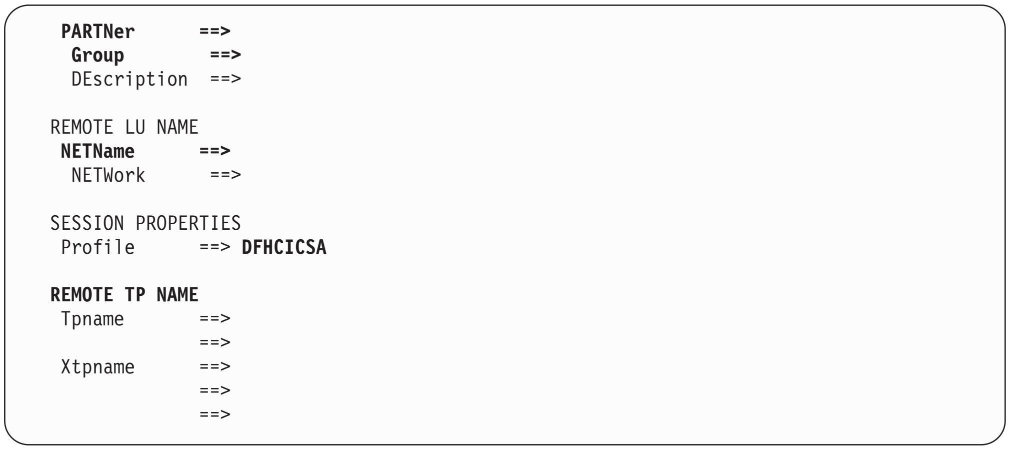
Because it is possible for programs to use the SAA communications interface SET calls to change the PARTNER name and the TPNAME name, CICS does not check that the CONNECTION definition is present when the PARTNER resource is being installed.

If you leave out the PROFILE attribute, CICS uses the default profile DFHCICSA. Because it is not possible for an SAA communications interface program to set a different profile, the PROFILE must be installed before the PARTNER resource. However, the PARTNER install does not fail if the PROFILE is missing; instead, a run-time error occurs when the partner conversation is attempted.

The MODENAME specified in the PROFILE need not be specified in a corresponding SESSIONS definition, as it is possible for the SAA communications interface program to set a different value for MODENAME.

CICS programs, and SAA communications interface programs that do not use the SET calls, require that all the relevant definitions be installed. You can use CEDA CHECK to help find out if all required definitions are in a group.

Defining a PARTNER



```
 PARTNer      ==>
  Group        ==>
  DEscription  ==>

REMOTE LU NAME
 NETName      ==>
  NETWork      ==>

SESSION PROPERTIES
 Profile      ==> DFHCICSA

REMOTE TP NAME
 Tpname       ==>
              ==>
 Xtpname      ==>
              ==>
              ==>
```

Figure 30. The DEFINE panel for PARTNER

Options

DESCRIPTION(*text*)
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see “How the CSD file is organized—groups and lists” on page 8.

The GROUP name can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and $. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

NETNAME(*name*)
is the NETNAME attribute specified in the CONNECTION definition. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command. For further information on the CONNECTION definition and its NETNAME attribute, see “CONNECTION” on page 157.

NETWORK(*name*)
You can use this optional attribute to specify the name of the network on which the partner LU is located. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

PARTNER(*name*)
specifies the name of this PARTNER definition. The name can be up to eight characters in length. The characters allowed are A-Z a-z and 0-9. Lowercase characters are treated as uppercase characters. Do not use partner names beginning with DFH, because these characters are reserved for use by CICS.

A partner definition specifies the SAA communications interface information required to establish a conversation with a partner program. For further guidance on this, see the *Common Programming Interface Communications Reference* manual.

PROFILE(*name*)
specifies which communication profile is to be used for the session and conversation. The default PROFILE is DFHCICSA. The name can be up to 8 characters in length. The acceptable characters are: A-Z a-z 0-9 $ @ # . / - _ % & ¢ ? ! : | " = ¬ , ; < and >. For information on how PROFILE resources are defined, see "PROFILE" on page 235.

TPNAME(*name*)
specifies the name of the remote transaction program that will be running on the partner LU. The definition of a remote TP name is mandatory; you must specify either TPNAME or its alternative, XTPNAME.

This name can be up to 64 characters in length. The acceptable characters are: A-Z a-z 0-9 $ @ # . / - _ % & ¢ ? ! : | " = ¬ , ; < and >. If this range of characters is not sufficient for a name that you wish to specify, you may use the XTPNAME attribute instead of TPNAME.

XTPNAME(*value*)
This attribute may be used as an alternative to TPNAME; you **must** specify one of the two, because the definition of a remote TP name is mandatory.

Enter a hexadecimal string up to 128 characters in length, representing the name of the remote transaction program that runs on the partner LU. All hexadecimal combinations are acceptable **except X'40'**.

To specify an XTPNAME more than 72 characters long to DFHCSDUP, put an asterisk in column 72. This causes the following line to be concatenated.

PROCESSTYPE

Using the CICS business transaction services (BTS) API, you can define and execute complex business applications called *processes*. A process is represented in memory as a block of storage containing information relevant to its execution. It also has associated with it at least one additional block of information called an activity instance. When not executing under the control of the CICS business transaction services domain, a process and its activity instances are written to a data set called a *repository*.

You can categorize your BTS processes by assigning them to different process-types. This is useful, for example, for browsing purposes. The activities that constitute a process are of the same process-type as the process itself.

A PROCESSTYPE definition defines a BTS process-type. It names the CICS file which relates to the physical VSAM data set (repository) on which details of all processes of this type (and their activity instances) are to be stored.

Note: Records for multiple process-types can be written to the same repository data set.

Figure 31 on page 233 shows the relationship between PROCESSTYPE definitions, FILE definitions, and BTS repository data sets. Notice that multiple PROCESSTYPE definitions can reference the same FILE definition; and that multiple FILE definitions can reference the same BTS data set.

You may want to record the progress of BTS processes and activities for audit purposes, and to help diagnose errors in BTS applications. If so, you can name the CICS journal to which audit records are to be written, and the level of auditing that is required, for processes of the specified type.

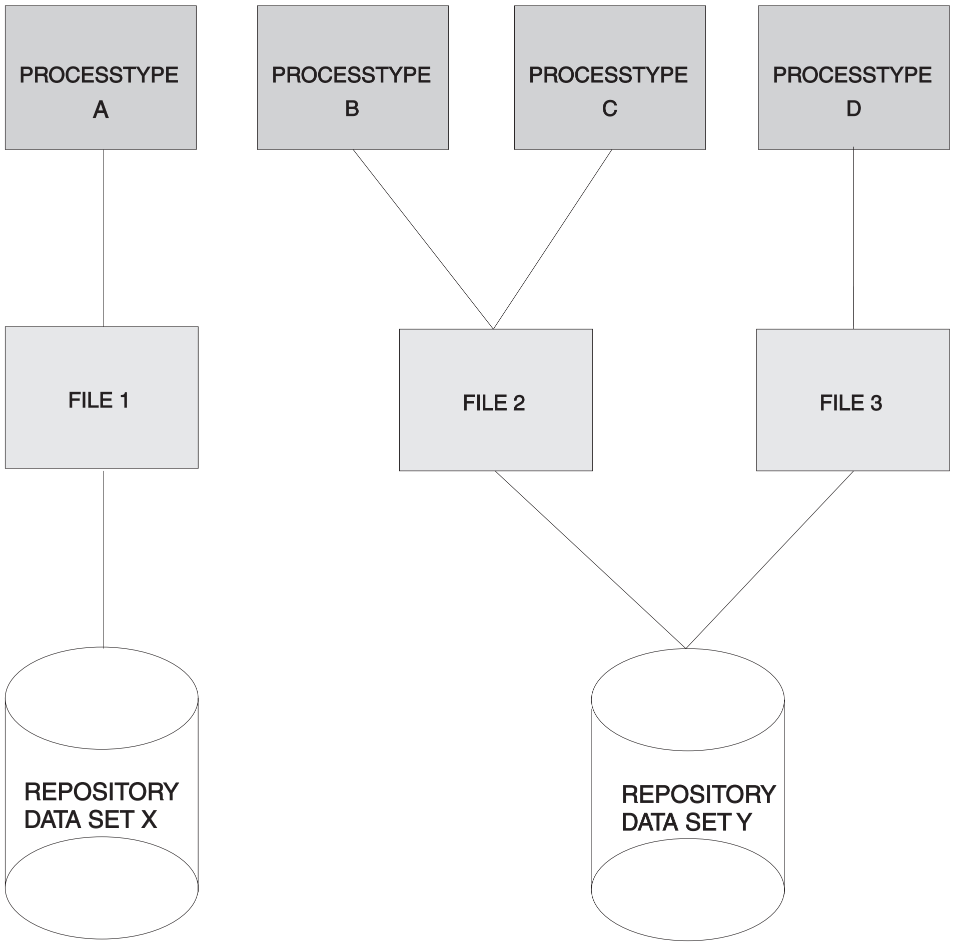
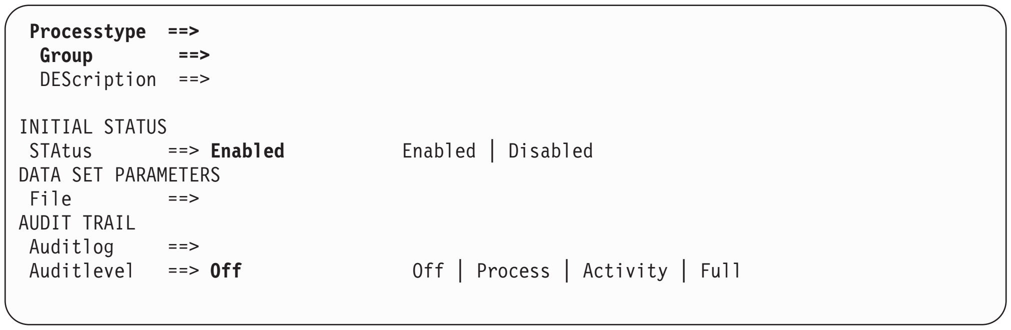


Figure 31. PROCESSTYPE definitions, FILE definitions, and BTS repository data sets

|



```
 Processtype  ==>
  Group        ==>
  DEScription  ==>

INITIAL STATUS
 STAtus       ==> Enabled           Enabled | Disabled
DATA SET PARAMETERS
 File         ==>
AUDIT TRAIL
 Auditlog     ==>
 Auditlevel   ==> Off                Off | Process | Activity | Full
```

Figure 32. The DEFINE panel for PROCESSTYPE

| ## Options

| **AUDITLEVEL({OFF|*PROCESS*|*ACTIVITY*|*FULL*})**
| specifies the initial level of audit logging for processes of this type. If you
| specify any value other than OFF, you must also specify the AUDITLOG
| option.

| **ACTIVITY**
| Activity-level auditing. Audit records will be written from:

1. The process audit points
2. The activity primary audit points.

FULL Full auditing. Audit records will be written from:
1. The process audit points
2. The activity primary *and* secondary audit points.

OFF No audit trail records will be written.

PROCESS
Process-level auditing. Audit records will be written from the process audit points only.

For details of the records that are written from the process, activity primary, and activity secondary audit points, see the *CICS Business Transaction Services* manual.

AUDITLOG(*name***)**
specifies the name of a CICS journal to which audit trail records will be written, for processes of this type and their constituent activities. The name can be up to eight characters long. If you do not specify an audit log, no audit records will be kept for processes of this type.

DESCRIPTION(*text***)**
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

FILE(*name***)**
specifies the name of the CICS file definition that will be used to write the process and activity records of this process-type to its associated repository data set. The name can be up to eight characters long. The acceptable characters are A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase.

PROCESSTYPE(*name***)**
specifies the name of this PROCESSTYPE definition. The name can be up to eight characters in length. The acceptable characters are: A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬, ; < >. Leading and embedded blank characters are not permitted. If the name supplied is less than eight characters, it is padded with trailing blanks up to eight characters.

STATUS({ENABLED|*DISABLED***})**
specifies the initial status of the process-type following a CICS initialization with START=COLD or START=INITIAL. After initialization, you can use the CEMT SET PROCESSTYPE command to change the status of the process-type. The status of the process-type following a restart is recovered to its status at the previous shutdown.

DISABLED
Processes of this type cannot be created. An EXEC CICS DEFINE PROCESS request that tries to create a process of this type results in the INVREQ condition being returned to the application program.

ENABLED
Processes of this type can be created.

PROFILE

You specify options that control the interactions between transactions and terminals or logical units, as a PROFILE. The PROFILE is a means of standardizing the use of such options as screen size and printer compatibility, and the use of such functions as message journaling and the node error program.

MODENAME
A profile is associated with the communication between a transaction and an LUTYPE6.1 or APPC session to another system. For APPC sessions, you refer on the PROFILE definition to the MODENAME that is also named on the SESSIONS definitions. This MODENAME is the name of the mode set to which the sessions belong. See “APPC links and parallel sessions” on page 158 and “APPC (LUTYPE6.2) single session terminal” on page 158.

When installed in CICS, the information from the PROFILE definition creates an entry in the profile table. This entry is later used by each transaction that references that PROFILE.

There are CICS-supplied PROFILE definitions suitable for most purposes. Each TRANSACTION definition names the PROFILE to be used. If you do not specify a PROFILE, the transaction uses the PROFILE supplied for using a terminal in a standard way.

With CICS intercommunication facilities (for example, function shipping), a PROFILE is needed for the communication between the transaction and the session. The attributes of the CICS-supplied profiles are shown in “PROFILE definitions in group DFHISC” on page 526. The *CICS Intercommunication Guide* gives further information about the CICS-supplied PROFILEs, and tells you about defining your own profiles.

Defining a PROFILE

```
  PROFile      ==>
   Group        ==>
   DEscription  ==>
  Scrnsize     ==> Default           Default | Alternate
  Uctran       ==> No                No | Yes
  MOdename     ==>
  Facilitylike ==>
  PRIntercomp  ==> No                No | Yes
 JOURNALLING
  Journal      ==> No                No | 1-99
  MSGJrnl      ==> No                No | INPut | Output | INOut
 PROTECTION
  MSGInteg     ==> No                No | Yes
  Onewte       ==> No                No | Yes
  PROtect        : No                No | Yes
  Chaincontrol ==> No                No | Yes
 PROTOCOLS
  DVsuprt      ==> All               All | Nonvtam | Vtam
  Inbfmh       ==> No                No | All | Dip | Eods
  RAq          ==> No                No | Yes
  Logrec       ==> No                No | Yes
 RECOVERY
  Nepclass     ==> 000               0-255
  RTimout      ==> No                No | 1-7000
```

Figure 33. The DEFINE panel for PROFILE

Options

CHAINCONTROL({NO|YES})
specifies whether the application program can control the outbound chaining of request units. If you specify CHAINCONTROL(YES), ONEWTE(YES) means one chain and not one terminal control output request.

DESCRIPTION(*text*)
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

DVSUPRT({ALL|NONVTAM|VTAM})
specifies the devices (terminals or logical units) that are to be supported. The access method used by a particular terminal or logical unit is specified in its associated TCTTE.

ALL The profile can be used with any terminal or logical unit.

NONVTAM
The profile can be used only with non-VTAM terminals.

VTAM
The profile can be used only with logical units.

FACILITYLIKE(*name*)
This is an optional parameter that specifies the name of an existing (four character) terminal resource definition to be used as a template for the bridge facility. It can be overridden by specifying FACILITYLIKE in the bridge exit.

There is no default value for this parameter.

If you are running in a CICS system started with the VTAM=NO system initialization (SIT) parameter, the resource definition specified by FACILITYLIKE must be defined as a remote terminal.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see "How the CSD file is organized—groups and lists" on page 8.

The GROUP name can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and $. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

INBFMH({NO|ALL|DIP|EODS}) (SNA LUs only)
specifies, for profiles used with logical units, whether a function management header (FMH) received from a logical unit is to be passed to the application program.

ALL All FMHs (except APPC FMHs and LU6.1 ATTACH and SYNCPOINT FMHs that are processed by CICS) are passed to the application program. This value is required for function shipping transactions such as CSMI, transactions which use distributed transaction processing, and for distributed program link requests.

DIP The batch data interchange program (DFHDIP) is to process inbound

FMHs. BMS issues a batch data interchange receive request if a BMS receive request has been issued, and a batch data interchange receive request is issued instead of a terminal control receive request.

EODS An FMH is passed to the application program only if it indicates end of data set (EODS).

NO The FMHs are discarded.

JOURNAL({NO|*number*)
specifies that you want automatic journaling of messages to take place, by giving the identifier of the journal.

NO No automatic journaling of messages is to take place.

number
The journal identification to be used for automatic journaling. This can be any number in the range 01 through 99. This number is appended to the letters DFHJ to give a journal identification of the form DFHJ*nn* and this maps to an MVS system logger general log stream.

Note: Unlike in releases prior to CICS Transaction Server for OS/390 Release 1, DFHJ01 is **not** the system log.

LOGREC({NO|YES})
specifies whether the design of the application requires that each EXEC CICS RECEIVE request is to be satisfied by a logical record. This option allows existing 2770-and 2780-based application programs to be attached to a batch logical unit (for example, 3790 or 8100) without modification to the program.

MODENAME(*name*)
specifies the name that identifies a group of sessions for use on an APPC connection. The name can be up to eight characters in length, and must be the name of a VTAM LOGMODE entry defined to VTAM. It must not be the reserved name SNASVCMG. If you omit the modename, it defaults to blanks. See the *CICS Intercommunication Guide* for more information about VTAM modenames.

If a transaction that specifies this profile has been started using an EXEC CICS START command, the MODENAME is used for allocation of the principal facility. If a transaction performs an EXEC CICS ALLOCATE command specifying this profile, the MODENAME is used for allocation of the alternate facility.

If you do not specify a MODENAME, CICS selects a session from any one of the mode sets that have been defined.

The CICS-supplied profile DFHCICSA is used, if PROFILE is not specified on an EXEC CICS ALLOCATE command. For function shipping, the profile DFHCICSF is always used. MODENAME is not specified on the definition for either of these profiles, but you can add a MODENAME if you make your own copy. You must then ensure that the mode sets using your MODENAME have been defined in the TERMINAL or SESSIONS definition for all the systems with which communication takes place using APPC.

If a MODENAME is specified and you wish to remove it, delete completely the value previously specified by pressing the ERASE EOF key.

MSGINTEG({NO|YES}) (SNA LUs only)
specifies whether a definite response is to be requested with an output request to a logical unit. You cannot specify YES for a pipeline transaction.

MSGJRNL({NO|INPUT|OUTPUT|INOUT})
specifies which messages are to be automatically journaled. If you specify a value other than NO, you must also supply a value for the JOURNAL attribute.

NO No message journaling is required.

INPUT
Journaling is required for input messages.

OUTPUT
Journaling is to be performed for output messages.

INOUT
Journaling is to be performed for input and output messages.

NEPCLASS({0|*value*}) (VTAM only)
specifies the node error program transaction class. This value overrides the value specified on the TYPETERM and SESSION definitions.

0 This results in a link to the default node error program module for VTAM devices, or is the default value for non-VTAM devices.

value The transaction class for the (nondefault) node error program module. The value can be in the range 1 through 255. For programming information on the node error program, see the *CICS Customization Guide*.

ONEWTE({NO|YES})
specifies whether the transaction is permitted only one write operation or EXEC CICS SEND during its execution. YES has the effect of forcing the LAST option on the first write of the transaction. Any additional write requests are treated as errors, and the task is made ready for abnormal termination.

You must specify YES for a PIPELINE transaction.

PRINTERCOMP({NO|YES})
specifies the level of compatibility required for the generation of data streams to support the printer compatibility option for the BMS SEND TEXT command.

NO Each line of output starts with a blank character, so that the format is equivalent to that on a 3270 display where an attribute byte precedes each line.

YES No blank character is inserted, so that forms-feed characters included as the first character of your data are honored and the full width of the printer is available for your data.

If you use the BMS forms feed option, specify YES.

PROFILE(*name*)
specifies the name of this PROFILE definition. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ # . / - _ % & ¢ ? ! : | " = ¬ , ; < and >. Do not use profile names beginning with DFH, because
| these characters are reserved for use by CICS.

| **Note:** If you use a comma (,) in a name, you will be unable to use those
| commands such as
| `CEMT INQUIRE PROFILE(`*`value1`*`,`*`value2`*`)`

where the comma serves as a list delimiter. See the *CICS Supplied Transactions* manual for information about using lists of resource identifiers.

A profile specifies the options that control the interaction between CICS and a terminal or logical unit. A profile name is specified on the transaction definition to indicate the set of options that control the communication between the transaction and its principal terminal. You can also specify a profile name on an EXEC CICS ALLOCATE command to indicate the options that control communication between the transaction and the allocated session.

CICS supplies a number of profile definitions that are suitable for most purposes. For guidance on the names of the definitions, see the *CICS Operations and Utilities Guide*. Further guidance is also given in the *CICS Intercommunication Guide*.

PROTECT({NO|YES}) (SNA LUs only)
This attribute is obsolete from CICS Transaction Server for OS/390, but is retained for compatibility with earlier releases. If you already use PROTECT, you can still access it by using compatibility mode (see “Compatibility mode (CSD file sharing)” on page 12 for information). See “Appendix A. Obsolete attributes retained for compatibility” on page 497 for a description of PROTECT.

RAQ({NO|YES}) (SNA terminals only)
specifies whether the ‘read ahead queuing’ option is required.

NO The transaction obeys SNA protocols and only SEND and RECEIVE when in the correct mode. If it does not follow the protocol, it may be abended with code ATCV.

YES The transaction may not obey SNA protocols, and CICS queues incoming data on temporary storage until the data is specifically requested by the transaction. RAQ(YES) is provided only for compatibility with transactions that support both bisynchronous devices and logical units, and its use is not recommended.

RTIMOUT({NO|*value*})
specifies the time-out value for the read time-out feature. The task that is timed out receives an AKCT or AZCT abend. (Note that if a value is specified and you wish to let it default to NO, you must completely delete the value previously specified.)

RTIMOUT has no effect for MRO or basic (unmapped) APPC connections.

NO The read time-out feature is not required.

value This is an interval (MMSS for minutes and seconds) after which the task is terminated if no input has been received from the terminal. The maximum value that can be specified is 70 minutes. The value specified in this option is rounded up to units of 16.78 seconds. Thus, the minimum value (after rounding-up) is 16.78 seconds.

SCRNSIZE({DEFAULT|ALTERNATE})
specifies whether the DEFAULT or ALTERNATE buffer size for a 3270 display or printer is to be used. For further information on the choice of screen sizes and buffer sizes, refer to the ALTSCREEN and DEFSCREEN attributes on the TYPETERM definition.

The SCRNSIZE value is ignored if the TYPETERM definition has ALTSCREEN(0,0) and DEFSCREEN(0,0). That is, the screen size is assumed from the related TERMMODEL attribute in the TYPETERM definition; the page size is taken from PAGESIZE, and the ALTPAGE value is ignored. The 3270 erase write (EW) command is inserted for output requests with the ERASE option.

ALTERNATE
If the TYPETERM definition has nonzero ALTSCREEN, the alternate screen size mode is applied, using the erase write alternate (EWA) command. That is, whenever a terminal output request with the ERASE option is issued, the 3270 EWA command is inserted in the data stream. The ALTSCREEN value is assumed as the screen size, and BMS uses the value in ALTPAGE as the page size.

SCRNSIZE(ALTERNATE) may be used for all CICS service transactions (for example, CSMT).

DEFAULT
If the TYPETERM definition has nonzero ALTSCREEN or nonzero DEFSCREEN, the default screen size mode is applied, using the erase write (EW) command. That is, whenever the terminal issues a terminal output request with the ERASE option, the 3270 EW command is inserted in the data stream. The screen size specified in the DEFSCREEN attribute is assumed, and BMS uses the value specified in the PAGESIZE attribute as the page size.

Note: Both DEFAULT and ALTERNATE can be overridden by the DEFAULT and ALTERNATE options on the SEND MAP, SEND TEXT, and SEND CONTROL commands. See the *CICS Application Programming Reference* manual for programming information about these commands.

UCTRAN({NO)|YES}) (VTAM only)
specifies whether terminal input is to be translated to uppercase before passing to programs for the transaction using this profile.

You can also request translation to uppercase at the terminal level on the associated TYPETERM definition (see page 363) but be aware of the following points:

- A TYPETERM UCTRAN(YES) definition overrides a PROFILE UCTRAN(NO) definition. So, if you specify TYPETERM UCTRAN(YES), a PROFILE UCTRAN(NO) has no effect.
- A PROFILE UCTRAN(YES) definition overrides a TYPETERM UCTRAN(NO) definition.
- Specifying TYPETERM UCTRAN(TRANID) causes the tranid to be translated to uppercase so that CICS can locate the transaction definition. All other input received by the application is translated according to what is specified for PROFILE UCTRAN.
- UCTRAN(YES) on a profile definition does not cause translation of the input data until an EXEC CICS RECEIVE or CONVERSE is executed. This means that if the transaction is routed through a dynamic routing program, for example DFHDYP, the copy of the input data passed to the routing program is unaffected by the UCTRAN option of the PROFILE definition.

Note: In a transaction routing environment where your VTAM terminals have a remote definition on the AOR, and the AOR has a different UCTRAN value from the TOR, the TOR value of UCTRANST (as specified in an EXEC CICS SET TERMINAL command) overrides that on the AOR.

Table 14 shows which portion of the terminal input is translated (transaction id and/or data) according to the setting of the UCTRAN on the PROFILE and TYPETERM resource definitions.

Table 14. The effect of UCTRAN attributes on tranid and data translation

Profile (PROFILE)	**Terminal (TYPETERM)**		
	UCTRAN (YES)	**UCTRAN (NO)**	**UCTRAN (TRANID)**
UCTRAN (YES)	Tranid: Yes Data: yes	Tranid: No Data: Yes	Tranid: Yes Data: Yes
UCTRAN (NO)	Tranid: Yes Data: Yes	Tranid: No Data: No	Tranid: Yes Data: No

PROGRAM

You use the PROGRAM definition to describe the control information for a program that is stored in the program library and used to process a transaction, or part of a transaction. For example, this is where you would tell CICS whether the program can handle data located above the 16MB line. You can create PROGRAM definitions either by using CEDA or DFHCSDUP, or by setting the appropriate system initialization parameters and allowing programs to be autoinstalled. See "Chapter 11. Autoinstall for other resources" on page 145 for information on autoinstall for programs.

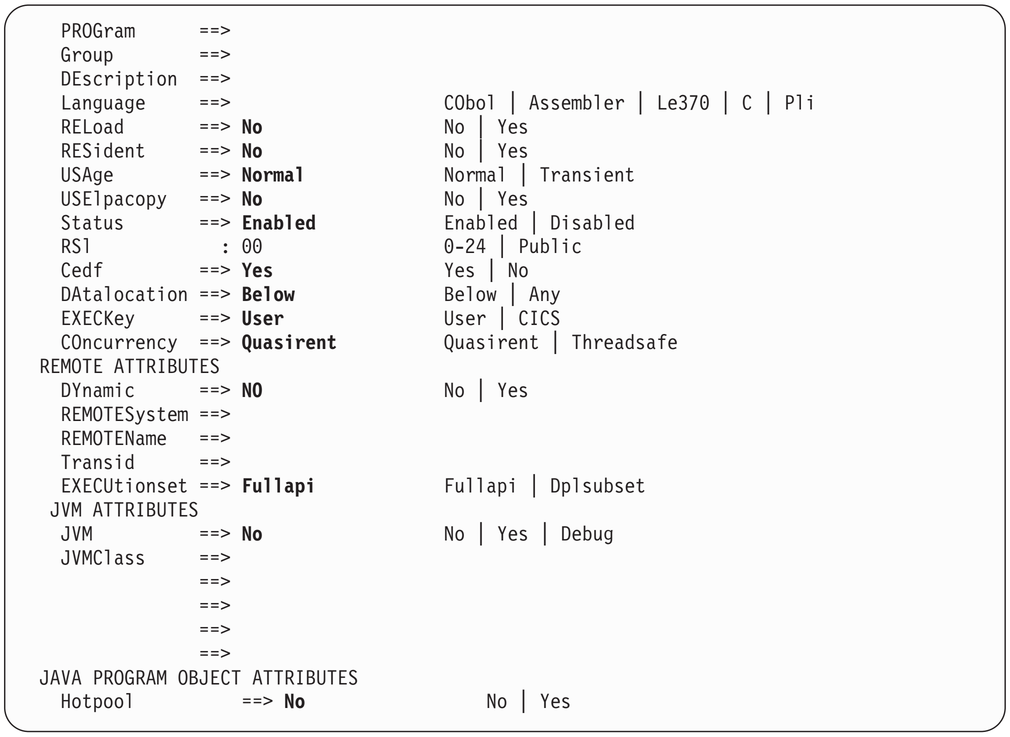
Installing PROGRAM definitions

Use the INSTALL command to install a new PROGRAM definition in your CICS system. If the program is a new version of an existing program, you must install it again using the INSTALL command, then use CEMT to make the new version available:

```
CEMT SET PROGRAM(pgmid) NEWCOPY
```

See the *CICS Supplied Transactions* manual for further information about CEMT.

Defining a PROGRAM



```
  PROGram      ==>
  Group        ==>
  DEscription  ==>
  Language     ==>                    CObol | Assembler | Le370 | C | Pli
  RELoad       ==> No                 No | Yes
  RESident     ==> No                 No | Yes
  USAge        ==> Normal             Normal | Transient
  USElpacopy   ==> No                 No | Yes
  Status       ==> Enabled            Enabled | Disabled
  RSl            : 00                 0-24 | Public
  Cedf         ==> Yes                Yes | No
  DAtalocation ==> Below              Below | Any
  EXECKey      ==> User               User | CICS
  COncurrency  ==> Quasirent          Quasirent | Threadsafe
REMOTE ATTRIBUTES
  DYnamic      ==> NO                 No | Yes
  REMOTESystem ==>
  REMOTEName   ==>
  Transid      ==>
  EXECUtionset ==> Fullapi            Fullapi | Dplsubset
 JVM ATTRIBUTES
  JVM          ==> No                 No | Yes | Debug
  JVMClass     ==>
               ==>
               ==>
               ==>
               ==>
JAVA PROGRAM OBJECT ATTRIBUTES
  Hotpool          ==> No                 No | Yes
```

Figure 34. The DEFINE panel for PROGRAM

Options

CEDF(YES|NO)
specifies the action of the execution diagnostic facility (EDF) when the program is running under EDF control.

NO The EDF diagnostic screens are not displayed.

YES The EDF diagnostic screens are displayed. If the program is translated with the NOEDF option, only the program initiation and termination

EDF screens are displayed. See Table 15.

Table 15. The effect on programs of CEDF(NO) and NOEDF

CEDF option on PROGRAM	EDF option on translator	
	EDF	**NOEDF**
YES	ALL EDF screens	Program initiation and termination screens only
NO	NO EDF screens	NO EDF screens
Note: The table shows how the CEDF option on the program resource definition interacts with the EDF option specified for the translator.		

CONCURRENCY({QUASIRENT | THREADSAFE})
specifies whether the program is written to threadsafe standards, or is only quasi-reentrant. You can specify the concurrency option for all CICS executable program objects:
- User application programs
- PLT programs
- User replaceable modules (URMs)
- Global user exit programs
- Task-related user exit programs

However, there are special considerations for URMs, task-related user exit programs, and global user exit programs. See the *CICS Customization Guide* for information about these categories of threadsafe program.

QUASIRENT
The program is quasi-reentrant only, and relies on the serialization provided by CICS when accessing shared resources.

The program is restricted to the CICS permitted programming interfaces, and must comply with the CICS quasi-reentrancy rules. For details of these, see the multithreading topic in the *CICS Application Programming Guide*.

This option is supported for all executable programs.

CICS ensures that the program always executes under the QR TCB, even when control is returned after it has invoked a JVM or an open API task-related user exit, or when it interacts with threadsafe programs.

THREADSAFE
The program is written to threadsafe standards, and when it accesses shared resources it takes into account the possibility that other programs may be executing concurrently and attempting to modify the same resources. The progam, therefore, uses appropriate serialization techniques when accessing any shared resources. JVM programs must be defined as threadsafe.

For information about writing threadsafe application programs, see the *CICS Application Programming Guide*.

This option is supported for all executable programs. Threadsafe programs must be Language Environment-conforming, or be assembler programs.

You can also specify the program CONCURRENCY option:
- Using a program autoinstall exit, if program autoinstall is active.

- Using the LE run-time options (for Language Environment-conforming programs only). See the *CICS Application Programming Guide* for information about using LE run-time options.

DATALOCATION({<u>BELOW</u>|ANY})
Commands using the SET option can return a data address to an application program; this operand specifies the location of the data. For example, in the command EXEC CICS RECEIVE SET(ptr-ref), ptr-ref is less than 16MB if DATALOCATION(BELOW) is specified or allowed to default, but may be greater than 16MB if DATALOCATION(ANY) is specified. Note that DATALOCATION does not affect the operation of the GETMAIN command. See the *CICS Application Programming Reference* manual for programming information about where CICS obtains storage in response to a GETMAIN command.

ANY The program can handle 31-bit addresses. The address of the data can be above or below the 16MB line. The DATALOCATION options are independent from the addressing mode of the link-edited program. Programs link-edited with addressing mode AMODE=24 cannot access data above 16MB; you should therefore ensure that the DATALOCATION option you specify is compatible with the addressing mode of the link-edited application program. For example:

- You are recommended to specify ANY for all 31-bit programs, unless they pass CICS data addresses on to other 24-bit programs.
- Specify DATALOCATION(BELOW) for an AMODE=24 program, unless storage addresses are being passed to a program that can access storage above 16MB, or the program explicitly switches addressing mode.

<u>BELOW</u>
The program can handle only 24-bit addresses and must therefore only be given data located below the 16MB line. If necessary, data is copied below the 16MB line before passing its address to the application program.

DESCRIPTION(*text*)
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

DYNAMIC({<u>NO</u>|YES})
specifies whether, if the program is the subject of a program-link request, the request can be dynamically routed.

<u>NO</u> If the program is the subject of a program-link request, the dynamic routing program is not invoked.

For a distributed program link (DPL) request, the server region on which the program is to execute must be specified explicitly on the REMOTESYSTEM option of the PROGRAM definition or on the SYSID option of the EXEC CICS LINK command; otherwise it defaults to the local region.

YES If the program is the subject of a program-link request, the CICS dynamic routing program is invoked. Providing that a remote server region is not named explicitly on the SYSID option of the EXEC CICS

LINK command, the routing program can route the request to the region on which the program is to execute.

The DYNAMIC option takes precedence over the REMOTESYSTEM option—see the description of the REMOTESYSTEM option.

For guidance information about the dynamic routing of DPL requests, see the *CICS Intercommunication Guide*.

EXECKEY(USER | CICS)
specifies the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. For all except reentrant programs (that is, programs link-edited with the RENT attribute), EXECKEY also defines, in conjunction with the residency mode, into which of the DSAs CICS loads the program (see note 2).

CICS This specifies that CICS is to give control to the program in CICS key when it is invoked. CICS loads the program into one of the CICS-key DSAs—either the CDSA or the ECDSA, depending on the residency mode specified for the program (see note 2).

In a CICS region with storage protection active, a CICS-key program has read and write access to CICS-key and user-key storage of its own task and all other tasks, whether or not transaction isolation is active.

USER This specifies that CICS is to give control to the program in user key when it is invoked. CICS loads the program into one of the user-key shared DSAs—either the SDSA or the ESDSA, depending on the residency mode specified for the program (see note 2 below).

In a CICS region with storage protection only active, a user-key program has read and write access to all user-key storage, but read-only access to CICS-key storage.

In a storage protection **and** transaction isolation environment, a user-key program has read and write access to the user-key task-lifetime storage of its own task only, and to any shared DSA storage, if the transaction is defined with ISOLATE(YES).

However, if a transaction is defined with ISOLATE(NO) in a transaction isolation environment, its user-key programs also have read and write access to the user-key task-lifetime storage of other transactions that are defined with ISOLATE(NO).

User-key programs always have read-only access to CICS-key storage.

Notes:

1. First-level global user exit programs, task-related user exit programs, user-replaceable programs, and PLT programs always execute in CICS key, regardless of the EXECKEY definition.
2. If the program is link-edited with the RENT attribute, CICS loads the program into one of the read-only DSAs—either the RDSA or the ERDSA, depending on the residency mode specified for the program, regardless of the EXECKEY option. The read-only DSAs are allocated from read-only storage only if RENTPGM=PROTECT is specified as a system initialization parameter.
3. This option is ignored for JVM programs, which always run in CICS-key.

EXECUTIONSET(FULLAPI | DPLSUBSET)
specifies whether you want CICS to link to and run a program as if it were running in a remote CICS region.

DPLSUBSET
Specify DPLSUBSET if you want CICS to link to the program and run it with the API restrictions of a remote DPL program. See the *CICS Application Programming Guide* for details of the API restrictions for a DPL program.

FULLAPI
Specify FULLAPI if you want CICS to link to the program and run it without the API restrictions of a DPL program. The program can use the full CICS API.

The EXECUTIONSET attribute applies only:

- to programs which are being linked to, and not to those which are the first to be given control by a transaction.
- when the REMOTESYSTEM name is the same name as the local CICS region. Its purpose is to test programs in a local CICS environment as if they were running as DPL programs.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see "How the CSD file is organized—groups and lists" on page 8.

The GROUP name can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and $. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

HOTPOOL(NO | YES)
specifies whether or not the Java program object is to be run in a preinitialized Language Environment enclave reused by multiple invocations of the program, under control of an H8 TCB. Programs defined with Hotpool (YES) must also be defined with CONCURRENCY (THREADSAFE).

Note: A Java program object is a Java program that has been processed by the VisualAge for Java Enterprise Toolkit for OS/390 bytecode binder.

NO The Java program object is not to be run in a preinitialized Language Environment enclave.

YES The Java program object is to be run in a preinitialized Language Environment enclave.

JVM(NO | YES | DEBUG)
specifies whether or not the program is to operate under the control of a Java Virtual Machine (JVM), and whether or not the JVM should operate in debugging mode. The debugging mode can be overridden at runtime using the user-replaceable module DFHJVMAT (see *CICS Customization Guide*).

NO The program is not to operate under a JVM.

YES The program is to operate under a JVM. You must specify a class name in the JVMCLASS option.

DEBUG
The program is to operate under a JVM in debugging mode. You must specify a class name in the JVMCLASS option.

JVMCLASS
specifies the main class in the CICS Java program to be executed by a JVM. This class name can be overridden using the user-replaceable module DFHJVMAT (see *CICS Customization Guide*). You can use DFHJVMAT to specify a class name larger than 255 characters.

Note that this option applies only to Java applications running under the control of a JVM, that is, the JVM field contains either DEBUG or YES. If you specify JVM(No), any value specified in the JVMClass field is ignored by CICS at program execution time.

LANGUAGE(COBOL | ASSEMBLER | LE370 | C | PLI)
specifies the program language. There is no default language, and if you omit the language, or define it incorrectly, the CICS program manager deduces the correct language and ignores your incorrect definition. This option is irrelevant for JVM programs; CICS deduces that the program is a Java program to run under the control of a JVM by the existence of either JVM(Yes) or JVM(Debug).

Note: The language of an assembler program can only be deduced if it has the assembler stub DFHEAI. If no language is defined and CICS cannot deduce the language, it will abend the transaction with an ALIG abend.

If you intend to share the CSD file with a level of CICS prior to CICS/ESA 4.1, do not leave this field blank, because it defaults to COBOL in the earlier release, which may not be correct.

ASSEMBLER
This is an assembler language program.

C This is a C/370™ program not compiled by a Language Environment®-conforming compiler.

COBOL
This is an OS/VS COBOL or a VS COBOL II program.

LE370 LANGUAGE(LE370) must be specified if the program exploits multi-language support, or if the program has been compiled by a Language Environment-conforming compiler.

PLI This is a PL/I program.

PROGRAM(*name*)
specifies the name of this PROGRAM definition. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command. Do not use program names beginning with DFH, because these characters are reserved for use by CICS.

To use the program in an active CICS system, it must have been link-edited into one of the libraries specified as part of the DFHRPL DD statement or, if it is reentrant, it may have been placed in the link pack area (LPA). For more information about installing application programs, see the *CICS System Definition Guide*.

RELOAD({NO | YES})
specifies whether a program control link, load, or XCTL request is to bring in a fresh copy of a program. This option does not apply to JVM programs.

NO Any valid copy of the program currently in storage is reused for the request.

YES A fresh copy of the program is brought into storage for every request. Furthermore, each of these program copies must be removed from storage explicitly, using a storage control FREEMAIN request, when it is no longer required and before the transaction terminates. If the relevant FREEMAINs are not issued, areas of the DSA/EDSA become tied up with inaccessible program copies, potentially causing storage shortage or fragmentation.

RELOAD(YES) can be used to load tables or control blocks that are modified by execution of any associated programs. It should not be specified for the first program loaded for a task. This is because the task would have no way of issuing a FREEMAIN for the program.

You must specify RELOAD(YES) for nonreentrant programs. For programming information about the RELOAD(YES) option, see the *CICS Application Programming Reference* manual.

REMOTENAME(*name*)
specifies the name by which the program is known in the remote CICS region. If you specify REMOTESYSTEM and omit REMOTENAME, the REMOTENAME attribute defaults to the same name as the local name (that is, the PROGRAM name on this resource definition).

REMOTESYSTEM(*name*)
specifies the name of a remote CICS region, if you want CICS to ship a distributed program link (DPL) request to another CICS region. The name you specify must be the name of the CONNECTION resource definition for the link to the remote CICS.

Note that, besides the REMOTESYSTEM option of the program definition, the DPL server region can also be specified by:

- The application program, using the SYSID option of the EXEC CICS LINK PROGRAM command
- The dynamic routing program.

The rules of precedence are as follows:

1. If an application program issues a DPL request, and the SYSID option on the EXEC CICS LINK command specifies a remote CICS region, CICS ships the request to the remote region.

   If the installed program definition specifies DYNAMIC(YES)—or there is no installed program definition—the dynamic routing program is invoked for notification only—it cannot re-route the request.
2. If an application program issues a DPL request, but the SYSID is the same name as the local CICS region or the SYSID option is not specified:
   a. If the installed program definition specifies DYNAMIC(YES)—or there is no installed program definition—the dynamic routing program is invoked, and can route the request.

      The REMOTESYSTEM option of the program definition, if specified, names the default server region passed to the dynamic routing program.

b. If the installed program definition specifies DYNAMIC(NO), CICS ships the request to the remote system named on the REMOTESYSTEM option. If REMOTESYSTEM is not specified, CICS runs the program locally.

The rules for specifying the remote system name are the same as for the CONNECTION attribute of the CONNECTION resource definition.

Note: You must not specify remote attributes for any user-written CICS programs, such as the dynamic transaction routing or autoinstall user programs.

RESIDENT({NO|YES})
specifies the residence status of the program. This option does not apply to JVM programs.

NO The program is not to be permanently resident. This value must be specified if RELOAD(YES) is specified.

YES The program is to be loaded on first reference and is then to be permanently resident in virtual storage, but is to be pageable by the operating system. When you specify RESIDENT(YES), CICS assumes a specification of USAGE(NORMAL).

Note: For further guidance information about the effects of the RESIDENT(YES) value, see the *CICS Performance Guide*.

RSL
The RSL attribute is not valid in CICS Transaction Server for OS/390. See "Appendix A. Obsolete attributes retained for compatibility" on page 497 for information.

STATUS({ENABLED|DISABLED})
specifies the program status.

DISABLED
The program may not be used.

ENABLED
The program may be used.

TRANSID(*name*)
If the program is dynamic, this is the default TRANSID used for the distributed program link (DPL) request. If the program is not dynamic, this specifies the name of the transaction you want the remote CICS to attach, and under which it is to run the remote program. If you do not specify a transaction name on the TRANSID attribute, the remote region executes the DPL program under one of the following CICS-supplied default mirror transactions:

CPMI This is the CICS mirror transaction for LU6.2 connections that specify data conversion.

CSMI This is the CICS ISC mirror transaction for MRO and LU6.2 connections with sync level 2.

CVMI This is the CICS/VM™ mirror transaction for LU6.2 connections with synclevel 1.

USAGE({NORMAL|TRANSIENT})
specifies when the storage for this program is released. This option does not apply to JVM programs.

NORMAL
When the use count for this program reaches zero, it becomes eligible for removal from storage as part of the normal dynamic program compression process.

This value must be specified if RELOAD(YES) is specified.

TRANSIENT
When the use count for this program becomes zero, the storage for this program is released. This value should be specified for programs that are referenced infrequently.

USELPACOPY({NO|YES})
specifies whether the program is to be used from the link pack area (LPA). This option does not apply to JVM programs.

NO The program is not to be used from the LPA. It is loaded into the CICS address space.

YES The program can be used from the LPA if LPA=YES is specified as a system initialization parameter. The use of the program from the LPA requires that it has been installed there and that the program is not named by the PRVMOD start-up option. For guidance on this, see the *CICS Transaction Server for OS/390 Installation Guide*.

REQUESTMODEL

A REQUESTMODEL resource definition provides the connection between an Internet Inter-ORB Protocol (IIOP) inbound request and the name of the CICS transaction that is to be initiated.

The inbound IIOP request is formatted according to the CORBA standards; the CICS transaction name requested is not explicitly specified. The REQUESTMODELs define templates that are compared with the inbound IIOP message to identify the type of request. The REQUESTMODEL selected supplies the name of a CICS TRANSID, which also associates the request with a set of execution characteristics such as security, priority, and monitor or accounting data.

Defining a REQUESTMODEL

```
REQuestmodel ==>
Group        ==>
DEscription  ==>
OMGModule    ==>
OMGInterface ==>
OMGOperation ==>
Transid      ==>
```

Figure 35. The DEFINE panel for the REQUESTMODEL resource definition

Options

DESCRIPTION(*text*)
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
specifies the group name, which can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see "How the CSD file is organized—groups and lists" on page 8.

REQUESTMODEL(*name*)
specifies the name of this REQUESTMODEL definition.

The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ and #.

OMGMODULE(*text*)
Defines a pattern that may match the qualified module name (coded in CORBA IDL), which defines the name scope of the interface and operation whose implementation is to be executed.

Each component of the module, and the interface and operation names, are CORBA identifiers made up from the following characters:

- Alphabetic, including accented characters.
- Numeric digits
- Underscore

The first character must be alphabetic. Case is not significant. However, mixed case is enabled for ease of use when referencing implementations in mixed case, such as Java. Each identifier (module component, interface and operation) can be arbitrarily long, but overall length is limited to 58 characters. Module components must be separated with a double colon (::).

OMGINTERFACE(*text*)
Defines a pattern that may match the IDL interface name. The maximum length of this field is 31 characters.

OMGOPERATION(*text*)
Defines a pattern matching the IDL operation name. The maximum length of this field is 31 characters.

TRANSID(*name*)
Defines the 4-character name of the CICS transaction to be executed when a request matching the specification of the REQUESTMODEL is received.

Generic pattern matching: OMGMODULE, OMGINTERFACE, and OMGOPERATION can be defined as generic patterns. The rules for pattern matching are based on those for RACF profile definitions, and are summarized as follows:

- Double colons are used as component separators.
- Wildcard characters **+** and * are used to match one (+) or more (*) characters (excluding colons).
- Wildcard ** matches any number of components of the module name, only one ** can be used in a pattern, but it can be used in any position (beginning, middle, or end).
- If used, the * wildcard character must be the last character in a double-colon separated component.

If a request is received that matches several generic patterns, the least generic is selected.

Examples:

```
DEFINE REQUESTMODEL(DFHGFACT) GROUP(DFHIIOP)
   DESCRIPTION(Supplied definitions)
   OMGMODULE(com::ibm::CosLifeCycle)
   OMGINTERFACE(GenericFactory)
   OMGOPERATION(*)
   TRANSID(CIOF)
```

You can define a generic definition that accepts any OMGMODULE, OMGINTERFACE, and OMGOPERATION, and applies no security restrictions.

```
DEFINE REQUESTMODEL(GENERIC) GROUP(TEST)
   DESCRIPTION(Generic definition for test purposes only)
   OMGMODULE(**)
   OMGINTERFACE(*)
   OMGOPERATION(*)
   TRANSID(IIOP)
```

SESSIONS

Before two systems can communicate using intersystem communication (ISC) or multiregion operation (MRO), they must be logically linked through one or more sessions. The nature of the logical link determines how they can communicate. You specify the logical link in the SESSIONS definition. CICS does **not** use the SESSIONS name when the definition has been installed in the active system. This name is used only to identify the definition in the CSD file.

You use the CONNECTION attribute of the SESSIONS resource definition to name the CONNECTION with which these SESSIONS are associated when they are installed in the active systems.

If you use the INSTALL command to install a new SESSIONS definition for MRO when a definition is already installed, you must close down all interregion communication (IRC) and open it again before you can use the definition.

1. Close IRC down:
   ```
   CEMT SET IRC CLOSED
   ```
2. Install the resource definitions:
   ```
   CEDA INSTALL GROUP(groupname)
   ```
3. When you have successfully installed the group containing the definitions, open IRC again:
   ```
   CEMT SET IRC OPEN
   ```

MRO links and sessions: When you install a SESSIONS definition for MRO, you are telling CICS about a set of parallel sessions between this CICS and another CICS. The number of sessions is determined by the SENDCOUNT and RECEIVECOUNT attributes. The SEND sessions are identified by names created from the SENDPFX and SENDCOUNT attributes. The RECEIVE sessions are identified by names created from the RECEIVEPFX and RECEIVECOUNT attributes.

APPC (LUTYPE6.2) links and parallel sessions: When you install the SESSIONS definition, the sessions are grouped (for the benefit of VTAM) into a modeset, which is identified by the MODENAME. The individual sessions are named by a counter; the first session created is named -999, the second -998, and so on. The value of this counter is retained over a warm or emergency start. The number of sessions created is controlled by the MAXIMUM attribute on the SESSIONS definition.

LUTYPE6.1 CICS-CICS ISC links and sessions: The way in which the sessions are identified by CICS depends on the way you defined them, using SENDPFX, SENDCOUNT, RECEIVEPFX, and RECEIVECOUNT like MRO sessions, or using SESSNAME as for CICS-IMS sessions.

Note: Use APPC for all new CICS-CICS ISC links.

LUTYPE6.1 CICS-IMS links and sessions: When you install the SESSIONS definitions in the active CICS system, CICS identifies each session by the SESSNAME attribute.

INDIRECT connections: Because the association between an INDIRECT link and the intermediate systems used for communicating with it is made at installation time, install the definition for the intermediate system before the definition for the INDIRECT link. If you install the INDIRECT link first, it remains dormant until the intermediate definition is installed, and until any other already installed connections that make reference to it are resolved. For example, System A is indirectly connected with system C through system B. In system A, install the following definitions in this order:

1. The intermediate system:
   ```
   CONNECTION(B) NETNAME(B) ACCESSMETHOD(IRC) ...
   ```
2. The INDIRECT link
   ```
   CONNECTION(C) NETNAME(C) ACCESSMETHOD(INDIRECT)
      INDSYS(B) ...
   ```

Defining a SESSION

```
 Sessions       :
 Group          :
 DEscription  ==>
SESSION IDENTIFIERS
 Connection   ==>
 SESSName     ==>
 NETnameq     ==>
 MOdename     ==>
SESSION PROPERTIES
 Protocol     ==> Appc              Appc | Lu61 | Exci
 MAximum      ==> 001 , 000         0-999
 RECEIVEPfx   ==>
 RECEIVECount ==>                   1-999
 SENDPfx      ==>
 SENDCount    ==>                   0-999
 SENDSize     ==> 04096             1-30720
 RECEIVESize  ==> 04096             1-30720
 SESSPriority ==> 000               0-255
 Transaction    :
OPERATOR DEFAULTS
 OPERId         :
 OPERPriority   : 000               0-255
 OPERRsl        : 0
 OPERSecurity   : 1
PRESET SECURITY
 USERId       ==>
OPERATIONAL PROPERTIES
 Autoconnect  ==> No                No | Yes | All
 INservice      :
 Buildchain   ==> Yes               Yes | No
 USERArealen  ==> 000               0-255
 IOarealen    ==> 00000 , 00000     0-32767
 RELreq       ==> No                No | Yes
 DIscreq      ==> No                No | Yes
 NEPclass     ==> 000               0-255
RECOVERY
 RECOVOption  ==> Sysdefault        Sysdefault | Clearconv | Releasesess
                                    | Uncondrel | None
 RECOVNotify    : None              None | Message | Transaction
```

Figure 36. The DEFINE panel for SESSIONS

Options

AUTOCONNECT({NO|YES|ALL})
specifies how connections are to be established. What you have to specify for LU6.1 and APPC sessions is discussed below:

APPC
For a VTAM-connected system that has AUTOCONNECT(YES) or (ALL) on the CONNECTION definition:

NO CICS does not attempt to bind any sessions when the connection is established. However, one or more user sessions may be allocated as part of any ACQUIRE CONNECTION processing which takes place.

YES or ALL
A contention-winner session is established (that is, BIND is performed) during CICS initialization, or when communication with VTAM is started using the CEMT SET VTAM OPEN command. If the connection cannot be made at this time because the remote system is unavailable, the link must be subsequently acquired using the CEMT SET CONNECTION(sysid) INSERVICE ACQUIRED command, unless the remote system becomes available in the meantime and itself initiates communications.

AUTOCONNECT(ALL) should not be specified for sessions to other CICS systems, because this can caused a bind race.

For a VTAM-connected system that has AUTOCONNECT(NO) on the CONNECTION definition:

ALL All sessions, not just contention winners, are established when the connection is acquired by issuing CEMT SET CONNECTION(name) ACQUIRED, or when the remote system itself initiates communication.

NO CICS does not attempt to bind any sessions when the connection is established. However, one or more user sessions may be allocated as part of any ACQUIRE CONNECTION processing that takes place.

YES Contention-winner sessions are established when the connection is acquired by issuing CEMT SET CONNECTION(sysid) ACQUIRED, or when the remote system itself initiates communication.

LU6.1
Specify AUTOCONNECT(YES) on the SESSIONS if you want the connection to be established at initialization or CEDA install.

Specify AUTOCONNECT(NO) on the SESSIONS if you do not want the connection to be established at initialization or CEDA installation.

BUILDCHAIN({YES|NO})
specifies whether CICS is to perform chain assembly before passing the input data to the application program.

NO Any TIOA received by an application program from this logical unit contains one request unit (RU).

YES Any terminal input/output area (TIOA) received by an application program from this logical unit contains a complete chain.

CONNECTION(*name*)
specifies the name of the CONNECTION definition that you want to use with this SESSIONS definition. The name can be up to four characters in length. The acceptable characters are A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

Note that the CONNECTION definition must be in the same GROUP as the SESSIONS definition.

DESCRIPTION(*text***)**
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

DISCREQ({NO | YES})
specifies whether disconnect requests are to be honored. DISCREQ applies to LUTYPE6.1 ISC sessions, but not to MRO sessions where CICS is not dealing with VTAM devices.

DISCREQ does not apply to APPC (LUTYPE6.2) sessions. When APPC is used, individual sessions are acquired as transactions need them, then are subsequently freed. Because it is possible to have multiple sessions between APPC logical units, there should never be a problem of one request holding up another. It is not possible to disconnect an individual APPC session; instead, you can issue a CEMT SET CONNECTION RELEASED command.

NO CICS is not to honor a disconnect request for a VTAM device.

YES CICS is to honor a disconnect request for a VTAM device, and issue a VTAM CLSDST macro instruction to terminate the VTAM session with that logical unit.

CESF LOGOFF or GOODNIGHT commands issued from the terminal also cause disconnection if you specify DISCREQ(YES).

GROUP(*groupname***)**
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see “How the CSD file is organized—groups and lists” on page 8.

The GROUP name can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and $. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

INSERVICE({YES | NO})
The INSERVICE attribute is not valid in CICS Transaction Server for OS/390. See “Appendix A. Obsolete attributes retained for compatibility” on page 497 for more information.

IOAREALEN({0 | *value1*},{0 | *value2*})
specifies the length, in bytes, of the terminal input/output area to be used for processing messages transmitted on the MRO link.

(*value1, value2***)**
Value1 specifies the minimum size of a terminal input/output area to be passed to an application program when a RECEIVE command is issued.

If value2 is not specified, or is less than value1, it defaults to the value of value1.

You can specify value2 as greater than or equal to value1. In this case, when the size of an input message exceeds value1, CICS uses a terminal input/output area value2 bytes long. If the input message size also exceeds value2, the node abnormal condition program sends an exception response to the terminal.

You can waste both real and virtual storage by specifying an IOAREALEN value that is too large for most messages transmitted on your MRO link. On the other hand, if you specify an IOAREALEN value that is either zero or smaller than most of your messages, excessive FREEMAIN and GETMAIN activity may occur. This results in additional processor requirements.

MAXIMUM({1|*value1*},{0|*value2*}) (APPC only)
specifies the maximum number of sessions that are to be supported for the modeset. Value1 must be greater than or equal to value2.

1|*value1*
The maximum number of sessions in the group. This value can be in the range 1 through 999. The default is 1.

0|*value2*
The maximum number of sessions that are to be supported as contention winners. This value can be in the range 0 to 999. The default is 0. Note that this operand has no meaning for a single session connection. (For further information on the effects of the MAXIMUM option, see the *CICS Intercommunication Guide*.)

SNA allows some resources (for example, switched lines) to be defined in the network as **limited resources**. At bind time, VTAM indicates to CICS whether the bind is over a limited resource. When a CICS task frees a session across a limited resource, CICS unbinds the session if no other task wants to use it.

If the sessions are to use limited resources, specify **MAXIMUM(***value1***,0)**. This causes any unbound session to be reset so that either side can then bind it as a winner when it is next required. For more information on limited resources, see the *CICS Intercommunication Guide*.

MODENAME(*name*) (APPC only)
specifies the name that identifies a group of sessions for use on an APPC connection. The name can be up to eight characters in length, and must be the name of a VTAM LOGMODE entry defined to VTAM. It must not be the reserved name SNASVCMG. If you omit the modename it defaults to blanks. See the *CICS Intercommunication Guide* for more information about VTAM modenames.

The MODENAME must be unique for each group of sessions defined for any one intersystem link. That is, the MODENAME must be unique among the SESSIONS definitions related to one CONNECTION definition. It is passed to VTAM as the LOGMODE name.

NEPCLASS({0|*value*})
specifies the transaction class for the node error program. This value acts as the default.

0 This results in a link to the default node error program module.

value The transaction class for the (nondefault) node error program module. The value can be in the range 1 through 255. For programming information about the node error program, see the *CICS Customization Guide*.

NETNAMEQ(*name*)
specifies the name by which the remote IMS system knows this particular session. This is used for CICS-IMS sessions. The name can be up to eight

characters in length. The acceptable characters are: A-Z 0-9 $ @ # . / - _ % & ¢ ? ! : | " = ¬ , ; < and >. Lowercase characters are converted to uppercase except when using the CREATE command.

OPERID(*code*)
The OPERID attribute is not valid in CICS Transaction Server for OS/390. See "Appendix A. Obsolete attributes retained for compatibility" on page 497 for more information.

OPERPRIORITY({0|*number*})
The OPERPRIORITY attribute is not valid in CICS Transaction Server for OS/390. See "Appendix A. Obsolete attributes retained for compatibility" on page 497 for more information.

OPERRSL({0|*number*[,...]})
The OPERRSL attribute is not valid in CICS Transaction Server for OS/390. See "Appendix A. Obsolete attributes retained for compatibility" on page 497 for more information.

OPERSECURITY({1|*number*[,...]})
The OPERSECURITY attribute is not valid in CICS Transaction Server for OS/390. See "Appendix A. Obsolete attributes retained for compatibility" on page 497 for more information.

PROTOCOL({APPC|LU61|EXCI})
specifies the type of protocol that is to be used for an intercommunication link (ISC or MRO).

APPC (LUTYPE6.2)
Advanced program-to-program communication (APPC) protocol. Specify this for CICS-CICS ISC.

EXCI The external CICS interface. Specify this to indicate that the sessions are for use by a non-CICS client program using the external CICS interface. If you specify EXCI, you must leave SENDCOUNT blank.

LU61 LUTYPE6.1 protocol. Specify this for CICS-CICS ISC, for CICS-IMS, or for MRO.

RECEIVECOUNT({blank|*number*})
specifies the number of MRO, LUTYPE6.1, or EXCI sessions that usually receive before sending.

For MRO, receive sessions can only receive before sending.

blank These sessions can send only; there are no receive sessions.

number
The number of receive sessions on connections that specify blank, LU61, or EXCI on the PROTOCOL attribute of the CONNECTION definition. CICS uses the number to generate the last two or three characters of the session names (see RECEIVEPFX for details).

If you are using the default receive prefix (<), or your own 1-character prefix, specify a number in the range 1 through 999.

If you specify a 2-character prefix, the number is restricted to the range 1 through 99.

Except for external CICS interface (EXCI) connections, the RECEIVECOUNT in this system should equal SENDCOUNT in the other system.

Note: CICS restricts the number of sessions for an EXCI address space to 100. When this limit is reached, IRP rejects further requests for a session with SYSTEM_ERROR reason code 608.

RECEIVEPFX(<|*prefix*)
specifies a 1-or 2-character prefix that CICS is to use as the first one or two characters of the receive session names (the names of the terminal control table terminal entries (TCTTEs) for the sessions).

Prefixes must not cause a conflict with an existing connection or terminal name.

< (MRO and EXCI sessions)
For MRO sessions, if you do not specify your own receive prefix, CICS enforces the default prefix—the less-than symbol (<), which is used in conjunction with the receive count to generate receive session names.

CICS creates the last three characters of the session names. The acceptable characters are A-Z 1-9. These 3-character identifiers begin with the letters AAA, and continue in ascending sequence until the number of session entries reaches the limit set by the RECEIVECOUNT value. Note that receive session names are generated **after** the send sessions, and they follow in the same sequence.

For example, if the last session name generated for the send sessions is <AAJ, using the default prefix (<) CICS generates the receive session names as <AAK, <AAL, <AAM, and so on. (This method of generation of session identifiers is the same as for APPC sessions, except for the initial prefix symbol.)

Note: If you specify your own prefix, CICS generates the session names as in earlier releases, which is the same as for LUTYPE6.1 sessions.

prefix **(LUTYPE6.1 sessions)**
If the sessions are on LUTYPE6.1 ISC connections, you must specify a 1-or 2-character prefix. Do not use the default < symbol for LUTYPE6.1 sessions.

For LUTYPE6.1 sessions (and MRO if you specify your own 1-or 2-character prefix), CICS generates session names by appending a number to the prefix, either in the range 1 through 99, or 1 through 999. The number begins with 1 and is incremented by 1 until the specified RECEIVECOUNT is reached.

RECEIVESIZE({4096|*value*})
specifies the maximum VTAM request unit (RU) size that these sessions are capable of receiving. The value must be between 1 and 30720 for LU61 sessions, or 256 and 30720 for APPC sessions. The default is 4096.

The value specified is transmitted to the connected logical unit. This value may be rounded down by CICS, depending on what value you specified, because the value must be transmitted in an architected form. The value may be negotiated down still further at BIND time.

If CICS is the secondary LU session, this indicates the maximum VTAM request unit (RU) size that these sessions are capable of sending.

RECOVNOTIFY({NONE|MESSAGE|TRANSACTION})
The RECOVNOTIFY attribute is not valid in CICS Transaction Server for OS/390. See "Appendix A. Obsolete attributes retained for compatibility" on page 497 for more information.

RECOVOPTION({SYSDEFAULT|CLEARCONV|

RELEASESESS|UNCONDREL|NONE})
This option applies to the recovery of sessions in a CICS region running with VTAM persistent sessions, or with XRF.

VTAM persistent sessions: In a CICS region running with persistent session support, this option specifies how you want CICS to recover the session, and return the terminal to service on system restart within the persistent session delay interval.

XRF: In a CICS region running with XRF support, this option specifies how you want CICS to recover the session, and return the terminal to service after an XRF takeover.

For all recovery options other than NONE, if the action taken is a VTAM UNBIND, the UNBIND is followed by a VTAM SIMLOGON.

CLEARCONV
VTAM persistent sessions: CLEARCONV is not supported for APPC sessions. It defaults to SYSDEFAULT.

XRF: If AUTOCONNECT(YES) is specified, the session is restarted. If AUTOCONNECT(NO) is specified, the session is unbound.

NONE
VTAM persistent sessions: In a CICS region running with persistent session support, this specifies that the session is not to be recovered at system restart within the persistent session delay interval: in effect, the sessions on the modegroup have no persistent session support. LU6.2 sessions are unbound and the modegroup CNOS value is reset to zero. After system restart, the session is reconnected automatically if you specify AUTOCONNECT(YES).

XRF: In a CICS region running with XRF support, this specifies that the logon state is not tracked by the alternate system, and the terminal session is not automatically recovered after a takeover; in effect, the terminal has no XRF support. After takeover, the terminal is reconnected automatically by the alternate system, if you specify AUTOCONNECT(YES).

RELEASESESS
VTAM persistent sessions: RELEASESESS is not supported for APPC sessions. It defaults to SYSDEFAULT.

XRF: If AUTOCONNECT(YES) is specified, the session is restarted. If AUTOCONNECT(NO) is specified, the session is unbound.

SYSDEFAULT
VTAM persistent sessions: In a CICS region running with persistent session support, this specifies that CICS is to select the optimum procedure to recover a session on system restart within the persistent session delay interval, depending on the session activity and on the characteristics of the terminal.

Although sessions are recovered, any transactions in-flight at the time of the failure are abended and not recovered. Transactions are also abended if the recovered session is being used by another CICS region over an APPC connection.

CICS recovers the session with the least possible impact, in one of the following ways:

- If the session was not busy at the time that CICS failed, no action is required.
- If the session was busy at the time that CICS failed, CICS issues a DEALLOCATE(ABEND) (equivalent to an EXEC CICS ISSUE ABEND) for the APPC conversation in progress at the time of the failure.
- If neither of the above applies, the session is unbound.

XRF: If AUTOCONNECT(YES) is specified, the session is restarted. If AUTOCONNECT(NO) is specified, the session is unbound.

UNCONDREL
Requires CICS to send an UNBIND request to release the active session. The UNBIND is sent whether or not the session was busy at the time of system restart (in the case of persistent session support) or takeover (in the case of XRF).

RELREQ=({NO | YES})
specifies whether CICS is to release the logical unit.

NO CICS is not to release the logical unit upon request by another VTAM application program.

YES CICS is to release the logical unit, if the logical unit is not currently part of a transaction.

SENDCOUNT(blank | ***number*****)**
specifies the number of MRO or LUTYPE6.1 sessions that usually send before receiving.

For MRO, send sessions must send before they can receive.

blank These sessions can receive only; there are no send sessions.

You must leave this field blank when the sessions are on an external CICS interface (EXCI) connection.

number
The number of send sessions on connections that specify blank or LU61 on the PROTOCOL attribute of the CONNECTION definition. CICS uses the number to generate the last two or three characters of the session names (see SENDPFX for details).

If you are using the default send prefix (>), or your own 1-character prefix, specify a number in the range 1 through 999.

If you specify a 2-character prefix, the number is restricted to the range 1 through 99.

Except for external CICS interface (EXCI) connections, the SENDCOUNT in this system should equal RECEIVECOUNT in the other system.

SENDPFX(> | ***prefix*****)**
specifies a 1-or 2-character prefix that CICS is to use as the first one or two

characters of the send session names (the names of the terminal control table terminal entries (TCTTEs) for the sessions).

Prefixes must not cause a conflict with an existing connection or terminal name.

> (MRO sessions)
For MRO sessions, if you do not specify your own send prefix, CICS enforces the default prefix—the greater-than symbol (>), which is used in conjunction with the send count to generate send session names.

CICS creates the last three characters of the session names from the alphanumeric characters A through Z, and 1 through 9. These 3-character identifiers begin with the letters AAA, and continue in ascending sequence until the number of session entries reaches the limit set by the SENDCOUNT value.

For example, using the default prefix (>), CICS generates session names as >AAA, >AAB, >AAC, and so on. (This method of generation of session identifiers is the same as for APPC sessions, except for the initial symbol.)

Note: If you specify your own prefix, CICS generates the session names as in earlier releases, which is the same as for LUTYPE6.1 sessions.

prefix **(for LUTYPE6.1 sessions)**
If the sessions are on LUTYPE6.1 ISC connections, you must specify a 1-or 2-character prefix. Do not use the default > symbol for LUTYPE6.1 sessions.

For LUTYPE6.1 sessions (and MRO if you specify your own 1-or 2-character prefix), CICS generates session names by appending a number to the prefix, either in the range 1 through 99, or 1 through 999. The number begins with 1 and are incremented by 1 until the specified SENDCOUNT is reached.

SENDSIZE({4096 | value})
specifies the maximum VTAM request unit (RU) size that these sessions are capable of sending. The value must be between 1 and 30720 for LU61 sessions, or between 256 and 30720 for APPC sessions. The default is 4096. The value may be negotiated down at bind time. Increasing the value of SENDSIZE causes more storage to be allocated for the session but may decrease the number of physical messages sent between the two nodes.

If CICS is the secondary LU session, this attribute indicates the maximum VTAM request unit (RU) size that these sessions are capable of receiving. The value must be between 256 and 30720.

SESSIONS(*name*)
specifies the name of this SESSIONS definition. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ # . / - _ % & ¢ ? ! : | " = ¬ , ; < and >.

This name is used to identify the SESSIONS definition on the CSD file. It is not used within the active CICS system.

SESSNAME(*name*)
specifies the symbolic identification to be used as the local half of a session qualifier pair in a CICS intercommunication parallel session.

The name can be up to four characters in length. The acceptable characters are: A-Z a-z 0-9 $ @ # . & ! : ; " , and ¬. If you type a name of fewer than four characters, it is padded with trailing blanks to four characters.

SESSPRIORITY({0|*number*})
specifies the terminal priority. This decimal value (0 through 255) is used in establishing the overall transaction processing priority. (Transaction processing priority is equal to the sum of the terminal priority, transaction priority, and operator priority; this must not exceed 255.)

TRANSACTION(*name*)
This attribute is now obsolete, but is supported to provide CSD file compatibility for earlier releases of CICS where it is still valid. See "Appendix A. Obsolete attributes retained for compatibility" on page 497 for more information.

USERAREALEN({0|*value*})
Specify the length, in bytes, of the user area for this session, in the range 0
through 255. It should be made as small as possible. The TCT user area is
initialized to zeros when the session is installed.

The TCT user area may be located above or below the 16Mb line in virtual
storage. Where it is located depends on the value of the TCTUALOC operand
# of the DFHSIT macro. You should ensure that this is specified correctly to
allow successful operation of any programs that are not capable of handling
31-bit addressing.

USERID(*name*)
specifies a user identifier used for sign-on (SEC=YES or MIGRATE) and referred to in security error messages, security violation messages, and the audit trail. It must be a valid userid defined to the security manager, or operators will be unable to sign on. All access to protected resources depends on USERID.

This USERID overrides a SECURITYNAME specified on the CONNECTION definition.

The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

TCPIPSERVICE

Use this resource definition to define which TCP/IP services are to use CICS internal sockets support. The internal CICS services that can be defined are IIOP and the CICS Web Interface.

The TCPIPSERVICE definition allows you to manage these internal CICS interfaces, with CICS listening on multiple ports, with different flavors of CICS Web or IIOP support on different ports.

To select the dynamic name server (DNS), you need to use a TCPIPSERVICE whose name begins with '**D**'. The Listener registers with MVS workload management services (WLM) using the following values:

- The TCPIPSERVICE TRANSID is passed to MVS as the **Group** name. If the TCPIPSERVICE name is of the form `Dxx.yyyy` then CICS uses `yyyy` as a prefix to TRANSID name.For example, a TCPIPSERVICE of DEV.CICS with a TRANSID of CIOR generates a group name of CICSCIOR.
- The APPLID specified in the system initialization table (SIT) is passed to MVS as the **Server**. If the APPLID=(gname,sname) format is used in the SIT, then **sname** is the value passed to MVS.

Notes:

1. Both the client and the CICS server must use the same TCP/IP **nameserver**
2. The **nameserver** must be able to perform a reverse look-up, that is, it must be able to translate the IP address of the server into a full hostname

Defining a TCPIPSERVICE

```
OVERTYPE TO MODIFY
 CEDA  DEFine TCpipservice(         )
  TCpipservice ==>
  Group        ==>
  Description  ==>
  Urm          ==>
  Portnumber   ==> 00000            1-65535
  Certificate  ==>
  STatus       ==> Open             Open | Closed
  SSl          ==> No               Yes | No | Clientauth
  Authenticate ==> AUTOregister     No | Basic | Certificate | Autoregister
                                    | AUTOMatic
  TRansaction  ==>
  Backlog      ==> 00001            0-32767
  TSqprefix    ==>
  Ipaddress    ==>
  Socket Close ==>                  No | 0-240000
```

Figure 37. The DEFINE panel for the TCPIPSERVICE resource definition

Options

AUTHENTICATE(<u>NO</u>|BASIC|CERTIFICATE|AUTOREGISTER|AUTOMATIC)

specifies the level of authentication required on connections associated with this TCPIPSERVICE definition.

<u>NO</u> No authentication of the client is required. However, if a registered certificate is provided by the client, it is used. This is the default.

BASIC

HTTP Basic Authentication of the client is attempted. If the client has sent an HTTP Authorization header, its contents are decoded as a user id and password. If these are valid, the user id is passed to the user-replaceable module for this TCPIPSERVICE definition. Otherwise, an HTTP 401 response is returned, together with a WWW-authenticate header, which causes the browser program to prompt the user for a new user id and password. These are returned in the required Authorization header. This process continues until the client either supplies a valid user id and password, or cancels the connection.

CERTIFICATE
A valid X.509 client certificate is required from the client, and it must map to a valid trusted user id in the external security manager's database. If such a certificate is not received, the connection is rejected with an HTTP 403 response. Otherwise, the derived user id is passed to the user-replaceable module for this TCPIPSERVICE definition.

This attribute cannot be specified unless SSL(CLIENTAUTH) is also specified.

AUTOREGISTER
This allows the client to register a certificate automatically. If the client presents a certificate that is not registered, an HTTP Basic Authentication dialogue is entered, in which the client must enter the user id for the certificate to be registered, together with its corresponding password. If this dialogue is completed successfully, the certificate is registered to the specified user id.

This attribute cannot be specified unless SSL(CLIENTAUTH) is also specified.

AUTOMATIC
This combines the AUTOREGISTER and BASIC functions. It attempts to authenticate the client as best it can. If a registered certificate is available, it is used. Otherwise, Basic Authentication is used to prompt the client for a user id and password. If an unregistered certificate is used, and Basic Authentication is successful, the certificate is registered.

BACKLOG(*number*)
specifies the number of TCP/IP connections for this service which are queued in TCP/IP before TCP/IP starts to reject incoming client requests.

CERTIFICATE(*text*)
specifies the name of a certificate within the key ring file that is to be used in the SSL handshake for this TCP/IP service. If this attribute is omitted, the certificate nominated as the default by the **mkkf** utility program for this keyring file is used.

DESCRIPTION(*text*)
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see "How the CSD file is organized—groups and lists" on page 8.

The GROUP name can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and $. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

IPADDRESS
specifies the dotted decimal IP address on which this TCPIPSERVICE will listen for incoming connections. It must be of the form nnn.nnn.nnn.nnn where nnn is 0 through 255. Possible values are:

blank When no IPaddress is specified, the default is *INADDR_ANY*

value The TCPIPSERVICE accepts connections on this particular address. If the address specified is not known to TCP/IP on the OS/390 host, the TCPIPSERVICE will not open. If you enter a specific address here, this definition may not be valid for CICS servers running on other OS/390 regions, and you may not be able to share the definition with those servers.

INADDR_ANY
The TCPIPSERVICE listens on any of the addresses known to TCP/IP for the OS/390 host. It is possible to have multiple IP addresses defined for a host. Specifying INADDR_ANY also allows for the TCPIPSERVICE definition to be shared among CICS servers.

PORTNUMBER(*number***)**
specifies, in the range 1 through 65535, the decimal number of the port on which CICS is to listen for incoming client requests.

SOCKETCLOSE(NO | *0-240000*
specifies if, and for how long, CICS should wait before closing the socket, after issuing a receive for incoming data on that socket.

No The socket is left open until data is received. While the socket is open it is unavailable to other tasks, and its associated CICS task is suspended indefinitely.

0-240000
The period of time (in HHMMSS format) after which CICS is to close the socket. Specifying 000000 closes the socket immediately if data is available

If you are using the TCPIPSERVICE for CICS Web Support and are processing only standard HTTP requests, SOCKETCLOSE(0) should be specified to avoid unnecessary CWXN transactions remaining in the system.

After the TCPIPSERVICE is installed, this value cannot be modified using CEMT; you must set the TCPIPSERVICE out of service, then re-install the TCPIPSERVICE with the modified definition.

CICS ignores the SOCKETCLOSE attribute on the first SOCKET RECEIVE issued after a connection is made. When a connection is accepted by the socket listener, it attaches a task to handle to connection. The first thing this task does is to issue a SOCKET RECEIVE. This first RECEIVE waits indefinitely for data, whatever you have specified on SOCKETCLOSE. This prevents a socket connection being closed as soon as it is initiated, even if no data is immediately available, and so prevents connection reset error at the client.

TRANSACTION(*name***)**
specifies the 4-character ID of the CICS transaction attached to process new

requests received for this service. For a Web TCPIPSERVICE definition, specify CWXN or an alias of CWXN. For an IIOP TCPIPSERVICE definition, specify CIOR or an alias of CIOR.

SSL(YES|NO|CLIENTAUTH)
: specifies whether the service is to use SSL. The default is NO. YES means that SSL flows other than client authentication are accepted for this service. CLIENTAUTH means that all SSL flows, including client authentication are accepted for this service.

STATUS({OPEN|CLOSED})
: Indicates the initial status of the service after installation. Set it to OPEN if CICS is to begin listening for this service after installation. Set to CLOSE if CICS is not to listen on behalf of this service after installation.

TCPIPSERVICE(*name*)
: specifies the 8-character name of this service.

TSQPREFIX
: specifies the 6-character prefix of the temporary storage queue used to store inbound data and Web documents created by applications. The TS queue prefix must be matched by a corresponding TSMODEL definition to meet your system and application requirements. Most Web transactions exchange relatively small amounts of data, so a TSMODEL defining the TS queues as main TS might be appropriate. For those applications handling large amounts of data, a model specifying main TS might be unacceptable due to storage constraints.

URM(*name*)
: specifies the name of the user-replaceable module to be invoked by this service. Implementation of this depends on the service being defined; this can be the name of any CICS program.

TDQUEUE

The following transient data resources can be managed using RDO:
- Intrapartition
- Extrapartition
- Indirect
- Remote

You use the TDQUEUE definition to describe the physical and operational characteristics of a transient data queue.

Intrapartition definitions contain attributes that provide information about recovery characteristics, trigger levels, associated transactions, facilities, and userids.

Extrapartition definitions contain information about the associated QSAM data set, and the number of buffers that are to be used.

Indirect definitions identify the underlying queue name.

Remote definitions contain the name of the remote system and the name by which the queue is known on that remote system.

Before a transient data queue can be used by an active CICS system, you must install its definition in the running system. CICS uses the definition to access the data set associated with the queue, and records the number of read and write operations on the queue.

Remote transient data queues can be defined using the CEDA transaction in one of two ways:
- If the queue TYPE is omitted and data is entered into only the REMOTE ATTRIBUTES section of the definition, a remote definition will be created.
- If a TYPE of INTRA or EXTRA is specified and the REMOTE ATTRIBUTES section is completed, both a local and a remote resource definition will be established at the same time.

  See Figure 39 on page 271 for an example of defining a dual-purpose transient data resource definition.

Installing transient data queue definitions

The destination control table (DCT) contains copies of all transient data resources installed on a CICS system. Entries can be added:
- During an initial or a cold start, using a DCT load module
- From entries retrieved from the CSD file as part of a GRPLIST install

Although you can use a mixture of RDO and macro definitions, it is better to convert your existing DFHDCT macro definitions to RDO. This is because the DFHDCT macro will not be supported in any subsequent release of CICS, and because there are some facilities available through RDO that are not supported by the DFHDCT macro; for example, the indoubt attributes WAIT and WAITACTION.

After CICS has been initialized, you can install additional resources using the CEDA transaction, or the EXEC CICS CREATE commands. During a warm start or an emergency restart, the DCT is constructed entirely from information in the global catalog, and any DCT load module will be ignored.

A transient data queue is **always** installed in an enabled state. Any queues that were disabled when a CICS system terminated, will still be enabled when the system is restored using a warm start or emergency restart.

Replacing existing transient data queue definitions: You can replace an existing transient data queue definition if the following rules are satisfied:
- CICS initialization is complete
- The queue TYPE is the same as the existing definition
- The intrapartition queue is disabled
- The extrapartition queue is disabled and closed

Existing definitions **cannot** be replaced during a cold start of CICS. If you are using a mixture of RDO and macro definitions, or are using multiple groups, take great care to ensure that duplicate names do not exist. Duplicate names cause error messages to be issued. The duplicate definition is not used to replace the existing one.

You can use the following transactions and commands to inquire about, set, and discard transient data definitions after they have been installed:
- CEMT INQUIRE TDQUEUE (or EXEC CICS INQUIRE TDQUEUE)
- CEMT SET TDQUEUE (or EXEC CICS SET TDQUEUE)
- CEMT DISCARD TDQUEUE (or EXEC CICS DISCARD TDQUEUE)
- The CECI transaction

Disabling transient data queues: A transient data queue cannot be disabled when:
- It is in use
- Other tasks are waiting to use it

This section relates only to intrapartition and extrapartition queues.

If tasks are waiting to use an extrapartition queue, a physically recoverable queue, or a nonrecoverable intrapartition queue, the queue enters a “disable pending” state. The last task to use the queue fully disables it.

If you try to disable a logically recoverable intrapartition transient data queue when units of work are enqueued on it, the queue enters a “disable pending” state. The last unit of work to obtain the enqueue fully disables the intrapartition queue.

If a unit of work owns an enqueue on a queue that is in a “disable pending” state, it is allowed to continue making updates.

When a queue is in a “disable pending” state, no new tasks can alter the queue’s state, or its contents. CICS returns a disabled response when you issue a READQ TD, WRITEQ TD, or DELETEQ TD request against a queue that is in a “disable pending” state.

Defining a transient data queue

You can use CEDA to define and redefine transient data resources in groups on the CSD file, from where they can be installed on the CICS region. You can also use CEDA to remove these resources. Figure 38 on page 270 shows the TDQUEUE resource definition screen with the various default values displayed. These defaults are appropriate only for specific queue types.

Defining a TDQUEUE

```
TDqueue      ==>
Group        ==>
DESCription  ==>
TYPE         ==>                    Extra|INTra|INDirect

EXTRA PARTITION PARAMETERS
DAtabuffers  ==> 1                  1 - 255
DDname       ==>
DSname       ==>
Sysoutclass  ==>
Erroroption  ==> Ignore             Ignore|Skip
Opentime     ==> Initial            Initial|Deferred
REWind       ==>                    Leave|Reread
TYPEFile     ==> Input              Input|Output|Rdback

RECORDSize   ==>                    0 - 32767

BLOCKSize    ==>
RECORDFormat ==> blank              Fixed|Variable|blank
BLOCKFormat  ==>                    Blocked|Unblocked|blank
Printcontrol ==>                    A|M
DIsposition  ==> Shr                Shr|Old|Mod

INTRA PARTITION PARAMETERS
Atifacility  ==> Terminal           Terminal|File|System
RECOVstatus  ==> No                 No|Physical|Logical
Facilityid   ==>
TRAnsid      ==>
TRIggerlevel ==> 1                  0 - 32767
Userid       ==>

INDOUBT ATTRIBUTES
WAIT         ==> Yes                Yes|No
WAITAction   ==> Reject              Queue|Reject

INDIRECT PARAMETERS
Indirectname ==>

REMOTE PARAMETERS
REMOTEName   ==>
REMOTESystem ==>
REMOTELength ==>
```

Figure 38. The CEDA DEFINE panel for TDQUEUE resources

Dual-purpose resource definition for transient data: You cannot specify TYPE=REMOTE for transient data queues.

Instead, you may wish to consider dual-purpose resource definition (see page 15). Dual-purpose resource definition can be used with transient data definitions. Figure 39 on page 271 gives an example of dual-purpose resource definition for transient data resources.

```
Group        ==> Example
DESCription  ==>
TYPE         ==> Intra   Extra|INTra|INDirect

EXTRA PARTITION PARAMETERS
DAtabuffers  :                             1 - 255
DDname       :
DSname       :
Sysoutclass  :
Erroroption  :                             Ignore|Skip
Opentime     :                             Initial|Deferred
REWind       :                             Leave|Reread
TYPEFile     :                             Input|Output|Rdback

RECORDSize   :                             0 - 32767

BLOCKSize    :
RECORDFormat :                             Fixed|Variable|Undefined
BLOCKFormat  :                             Blocked|Unblocked
Printcontrol :                             A|M
DIsposition  :                             Shr|Old|Mod

INTRA PARTITION PARAMETERS
Atifacility  ==> Terminal          Terminal|File|System
RECOVstatus  ==> Logical           No|Physical|Logical
Facilityid   ==> FR1
TRAnsid      ==>
TRIggerlevel ==> 00001             0 - 32767
Userid       ==>

INDOUBT ATTRIBUTES
WAIT         :                             Yes|No
WAITAction   :                             Queue|Reject

INDIRECT PARAMETERS
Indirectname :

REMOTE PARAMETERS
REMOTEName   ==> FR1
REMOTESystem ==> CICQ
REMOTELength ==>
```

Figure 39. Dual-purpose resource definition for transient data

If the definition shown in Figure 39 is installed in a system called CICQ, that definition becomes a local intrapartition queue (the value of REMOTESYSTEM is CICQ).

If the definition shown in Figure 39 is installed in a system other than CICQ, the definition becomes a REMOTE queue.

Options

Atifacility=({TERMINAL|FILE|SYSTEM}) (intrapartition queues only)
specifies the type of destination the queue represents.

FILE The transient data queue is to be used as a file of data records that are not associated with a particular terminal or system. ATI does not require a terminal to be available.

SYSTEM
The transient data queue is to be associated with the specified system identifier. The system must be defined to the local CICS system using an RDO CONNECTION definition.

Specifying Atifacility=System initiates a distributed transaction processing (DTP) session. For more information about DTP considerations in application programming, see the *CICS Application Programming Guide*.

TERMINAL
: The transient data queue is to be associated with the terminal. The terminal must be defined to CICS. If you do not specify TERMINAL, it defaults to the value of FACILITYID. If ATI is used, as specified in the TRANSID and TRIGGERLEVEL attributes, the transaction that is initiated is associated with the specified terminal, which must be available before the transaction can be initiated.

BLOCKFormat({BLOCKED|UNBLOCKED|*blank*}) (extrapartition queues only)
specifies the block format of the data set. There is no default. If you specify the record format (RECORDFORMAT attribute) as undefined (or allow it to default), you cannot specify anything for the BLOCKFORMAT attribute.

blank Indicates that no block format is defined for this data set. Leave this field blank if you leave RECORDFormat blank.

BLOCKED
Blocked record format.

UNBLOCKED
Unblocked record format.

BLOCKSize({length}) (extrapartition queues only)
specifies the length of the block, in bytes.

SYSOUT data sets only
When SYSOUTCLASS has been specified or defaulted and if a value has not been specified on the RECORDSIZE attribute, the value of BLOCKSIZE must not be greater than 8972 bytes if RECORDFormat(Variable) has been specified. If the value of RECORDFORMAT is not variable, the value of BLOCKSIZE should not be greater than 8986 bytes.

V format data sets
Each block consists of a block descriptor word followed by one or more logical records. The value specified for BLOCKSIZE must include four bytes for the block descriptor word, and allow for the largest possible logical record.

DAtabuffers({1|number}) (extrapartition queues only)
specifies the number of buffers to be provided, up to a maximum of 255.

DDname(*name*)
specifies a 1-to 8-character value that may refer to a data set defined in the startup JCL.

DESCription(*text*)
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

DISPosition({SHR|OLD|MOD}) (extrapartition queues only)
specifies the disposition of the data set.

MOD CICS first assumes that the data set exists. For an existing sequential data set, MOD causes the read/write mechanism to be positioned after the last record in the data set. The read/write mechanism is positioned after the last record each time the data set is opened for output.

If CICS cannot find volume information for the data set:
- On the DD statement
- In the catalog
- Passed with the data set from a previous step

it assumes that the data set is being created in this job step.

For a new data set, MOD causes the read/write mechanism to be positioned at the beginning of the data set.

OLD The data set existed before this job step.

SHR The data set existed before this job step and can be read by other concurrent jobs.

DSname({name | DUMMY}) (extrapartition queues only)
specifies the name of the QSAM data set that is to be used to store records written to this extrapartition queue.

When CICS receives a request to open an extrapartition transient data queue, the startup JCL is referenced to check if a data set definition has been created. If one is not found, the 44-character name specified on the DSNAME attribute is used to dynamically allocate the required data set.

If you have JCL that preallocates this queue's DSCNAME to a DSNAME, the DSNAME in the resource definition is overridden by the DSNAME from the JCL. JCL allocation always takes priority.

Partitioned data sets (PDS) are not supported on the DSNAME attribute. If you want to use a PDS member for an extrapartition queue data set, code it explicitly in your JCL. Bear in mind that if you inquire upon this queue, the DSNAME returned will not give you any indication of the member name.

DUMMY
A dummy data set name.

name The 44-character name of a physical data set.

Note: You are strongly recommended **not** to use log streams for extrapartition queue data sets. If you do, unpredictable results may occur.

ERROROPTION({IGNORE | SKIP}) (extrapartition queues only)
specifies the action to be taken if an I/O error occurs. This can be one of the following:

IGNORE
The block that caused the error is accepted.

SKIP The block that caused the error is skipped.

FACILITYID({*name*}) (intrapartition queues only)
specifies a 4-character field that contains either:
- The system identifier for an intrapartition queue that specifies ATIFACILITY(SYSTEM)
- The terminal identifier where ATIFACILITY(TERMINAL) is specified.

If you do not specify anything in the FACILITYID field, it defaults to the name of the queue in each case.

If ATIFACILITY(FILE) is specified, the FACILITYID field must be left blank.

GROUP(*groupname*)
Every resource definition **must** have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS region when the group is installed. For information about groups, see "How the CSD file is organized—groups and lists" on page 8.

The group name can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and $. Lowercase characters are treated as uppercase characters. Do not use group names beginning with "DFH", because these characters are reserved for use by CICS.

INDIRECTNAME(*name*) (indirect queues only)
specifies the name of a transient data queue. The queue can be intrapartition, extrapartition, remote, or indirect. If a DCT entry does not exist for the queue with this name, you get an error when you try to use the queue.

OPENTIME({INITIAL|DEFERRED}) (extrapartition queues only)
specifies the initial status of the data set. The initial status can be one of the following:

DEFERRED
| The data set remains closed until you indicate that you want to open it
| by using the CEMT INQUIRE|SET TDQUEUE command.

INITIAL
The data set is to be opened at install time.

PRINTCONTROL({ASA|MACHINE|*blank*}) (extrapartition queues only)
specifies the control characters to be used. There is no default.

If you allow RECORDFormat to default to blank, you cannot specify anything in the PRINTCONTROL field. The control characters that can be used are:

ASA ASA control characters.

blank No control characters are to be used.

MACHINE
Machine control characters.

RECORDFORMAT({FIXED|VARIABLE|*blank*) (extrapartition queues only)
specifies the record format of the data set.

blank If RECORDFormat is not specified (that is, left blank), the BLOCKFORMAT and PRINTCONTROL fields must also be left blank. If the RECORDFormat is not specified in the resource definition, TD will attempt to derive this attribute from the CICS startup JCL or from the QSAM dataset definition at the time it attempts to open the queue. The open request will fail if this information cannot be derived from either of these sources.

FIXED
Fixed records. If you specify `RECORDFormat(Fixed)`, you must also specify a block format.

VARIABLE
Variable records. If you specify RECORDFormat(Variable), you must also specify a block format.

RECORDSIZE({1 | *number*}) (extrapartition and remote queues)
| specifies the length in bytes, in the range 0 through 32767.

1 The default record length is 1 byte.

number
The record length, in bytes, up to 32767.

SYSOUT data sets only
When the SYSOUTCLASS attribute has been specified, the value of the
RECORDFORMAT attribute must not be greater than 8968 bytes.

V format data sets
Each logical record consists of a record descriptor followed by a data record. The value entered in the RECORDSIZE field must include 4 bytes for the record descriptor word (or LLBB), and should allow for the largest possible data record.

RECOVSTATUS=({NO | PHYSICAL | LOGICAL}) (intrapartition queues only)
specifies the recoverability attributes of the queue in the event of an abnormal termination of either CICS or the transaction that is processing the queue. The recoverability attributes are:

LOGICAL
This queue is logically recoverable. Automatic logging is to be performed to keep track of accesses by application programs. If a transaction that accessed this queue was in-flight at the time of abnormal termination, or in the subsequent emergency restart or dynamic transaction backout, the queue is restored to the status it was in before the in-flight UOW modified it.

When this queue is accessed, the task that issued the DELETEQ TD, WRITEQ TD, or READQ TD command is enqueued on the input, the output, or both ends of the transient data queue. The enqueue is maintained until the task terminates (or issues a syncpoint request to signal the end of a UOW) to ensure the integrity of the data being accessed. This means that enqueues can be maintained for a longer time, and can result in a queue lockout if an application program accessing the queue performs more than one UOW against the queue without defining each separate UOW to CICS by issuing a syncpoint request.

Furthermore, when a DELETEQ request is issued for a logically recoverable queue, both the input and output ends of the queue are enqueued upon. This can increase the possibility of an enqueue lockout.

Note: CICS provides an enqueuing protection facility for logically recoverable (as distinct from physically recoverable) TD queues similar to that for recoverable files. However, CICS regards each logically recoverable destination as two separate recoverable resources—one for writing and one for reading.

In the case of a file record, a record is treated as a single resource and requires only one lock. The TD queue, on the other hand, has two 'ends'—the read end and the write end, and these can be enqueued on (locked) independently. This is because, to control both reading and writing from the TD queue (at the same time), CICS has to maintain two pointers (cursors)—one for reading and one for writing, and these need to be protected from conflicting transactions.

Queue records are held on one or more control intervals (CIs). Each CI is marked for release as soon as the last record on it has been read. However, the release does not occur until the end of task, or until after the next user syncpoint.

NO This queue is not recoverable. Automatic logging is not performed to keep track of accesses to this queue. Queue records are held on one or more control intervals (CIs). Each CI is released as soon as the last record on it has been read.

PHYSICAL
This queue is physically recoverable. Automatic logging is to be performed to keep track of accesses by application programs. If emergency restart occurs, this queue is to be recovered to its status at the time CICS terminated.

The queue is **not** recovered to its status at the time CICS terminated if the last action on the queue was a READQ request, and if the associated unit of work (UOW) did not commit the changes. On emergency restart, the last read operation is backed out, and appears never to have taken place.

Queue records are held on one or more control intervals (CIs). Each CI is released as soon as the last record on it has been read.

REMOTENAME(*name*) (remote queues only)
specifies the 4-character name by which the queue is known in the system or region on which the queue resides.

REMOTELENGTH({1|*number*}) (remote queues only)
specifies the length in bytes, in the range 1 through 32767.

For SYSOUT data sets, the value entered in the REMOTELENGTH field must not be greater than 8968 bytes (when the SYSOUTCLASS attribute has been specified).

1 The length is 1 byte.

number
The length in bytes, up to 32767.

If the queue is defined with TYPE=EXTRA, and no value is specified for REMOTELENGTH, the value on the RECORDSIZE attribute is used at installation time.

REMOTESYSTEM(*name*)
specifies the 4-character alphanumeric name of the system or region in which the remote transient data queue resides. The name entered must be the same as the name specified on the RDO CONNECTION definition. For more information about the CONNECTION definition, see “CONNECTION” on page 157.

When the transient data queue definition is installed, the name entered in the REMOTESYSTEM attribute is compared with the system identifier. If the names are different, the system or region is remote. If the names are the same, the value specified in the TYPE attribute is used. If the TYPE attribute is blank, the installation fails.

REWIND({LEAVE | REREAD}) (extrapartition queues only)
specifies the disposition of a tape data set. The disposition can be one of the following:

LEAVE
The current tape is positioned at the logical end of the data set.

REREAD
The current tape is positioned at the logical start of the data set.

SYSOUTCLASS (extrapartition queues only)
Instead of allocating an extrapartition queue to a physical data set, you can allocate it to a system output data set (referred to as SYSOUT).

Use the SYSOUTCLASS attribute to specify the class of the SYSOUT data set.

A single alphabetic character
This represents an output class that has been set up on the MVS system on which the CICS job is to run. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

An asterisk
This is the default class. SYSOUTCLASS defaults to an asterisk if you leave the DSNAME attribute blank and specify TYPEFILE=OUTPUT.

blank SYSOUTCLASS defaults to a blank character if you leave the DSNAME attribute blank and specify TYPEFILE=INPUT or TYPEFILE=RDBACK. In the latter case, the open operation will fail because a DSNAME **must** be specified for TYPEFILE=INPUT or TYPEFILE=RDBACK.

You can use SYSOUTCLASS as an alternative to DSNAME. As with DSNAME, the queue may already be preallocated to SYSOUT using a JCL DD statement. A JCL DD statement overrides any specification made using the TDQUEUE resource definition.

When CICS receives a request to open an extrapartition transient data queue, the startup JCL is referenced to check if a data set definition has been created. If one is not found, the 44-character name specified on the DSNAME attribute is used to dynamically allocate the required data set.

When SYSOUT is specified for a queue in the JCL, attributes other than class can also be specified (for example, form types).

Note: Specifying SYSOUT data sets using RDO supports the class parameter only. If you require other parameters, you specify SYSOUT data sets in the JCL.

For more information about SYSOUT and its associated classes, see the *OS/390 MVS JCL User's Guide*.

TDQUEUE(*name*)
specifies the 1- to 4-character name of a transient data queue. The acceptable characters are A-Z a-z 0-9 $ @ # . / _ % & ¢ ? ! : | " = ¬ , ; < >. If the name supplied is fewer than four characters, it is left-justified and padded with blanks up to four characters.

| **Note:** If you use a comma (,) in a name, you will be unable to use those
| commands such as

```
CEMT INQUIRE TDQUEUE(value1,value2)
CEMT SET     TDQUEUE(value1,value2)
```

| where the comma serves as a list delimiter. See the *CICS Supplied*
| *Transactions* manual for information about using lists of resource
| identifiers.

| **TRANSID(*name*) (intrapartition queues only)**

specifies the name of the transaction that is to be automatically initiated when the trigger level is reached. Transactions are initiated in this way to read records from the queue. If the TRANSID attribute is not specified (or if TRIGLEV=0 is specified), you must use another method to schedule transactions to read records from transient data queues.

The transaction specified must not reside in a remote CICS system. If it does, transaction initiation fails and a warning message is issued to the console.

TRIGGERLEVEL({1|*number*}) (intrapartition queues only)

specifies the number of records to be accumulated before a task is automatically initiated to process them. (This number is known as the trigger level.)

If you specify the TRANSID attribute, TRIGGERLEVEL defaults to 1. Specify a
# trigger level of 0 if you want to disable ATI processing. If you do not specify a
# transaction id, the trigger level is ignored.

If you have specified ATIFACILITY(TERMINAL), the task is not initiated until the specified terminal is available. If you have specified ATIFACILITY(FILE), a terminal is not necessary for the task to be initiated.

If, at maximum task, a short-on-storage or a no-space condition exists for a nonterminal destination, the task is not initiated. This is also true during stages 1 and 2 of initialization, and during the final stage of shutdown. The task is initiated when the stress condition no longer exists, and a subsequent TD WRITE occurs.

For logically recoverable transient data queues, the ATI task is not attached until the task commits forward. This may mean that the trigger level is far exceeded before ATI occurs.

If a VTAM terminal is defined as the destination for the CSTL transaction on two ISC CICS systems with a trigger level of 1, a performance problem may arise when both systems repeatedly acquire and release the terminal to write the session-started and session-ended messages.

You can change the trigger level when CICS is running using the CEMT transaction. If you reduce the trigger level to a number that is equal to (or less than) the number of records accumulated so far, the task is initiated when the next record is successfully put on the queue.

1 Only one record can accumulate.

number
The number of records that can accumulate (up to a maximum of 32767) before ATI occurs.

TYPE({EXTRA | INTRA | INDIRECT | })
specifies the following types of transient data queue:

EXTRA
A queue that is outside the CICS region is allocated to CICS.

Extrapartition queues are used for:

- Sending data outside the CICS region: for example, data created by a transaction for processing by a batch program.
- Retrieving data from outside the region: for example, data received from terminals as input to a transaction.

Extrapartition data is sequential and is managed by QSAM.

INDIRECT
An indirect queue is a queue that does not point to an actual data set, but to another queue. An indirect queue can be extrapartition, intrapartition, remote, or even another indirect queue.

For example, you can give a different symbolic name, INDIRECTDEST, to each of several different message types. You can then send all these message types to the same physical queue (INDIRECTDEST), or to different physical queues.

The DFH$TDWT sample program demonstrates how you can use indirect queues to send different categories of message to the same terminal. For programming information about DFH$TDWT, see the *CICS Customization Guide*. DFH$TDWT sample definitions are given in the *CICS 4.1 Sample Applications Guide*.

If the QUEUE operand of an EXEC CICS WRITEQ TD, EXEC CICS READQ, or EXEC CICS DELETEQ command specifies an indirect queue, access is determined by the security setting of the final target queue.

INTRA
A queue for data that is to be stored temporarily.

An intrapartition destination can be a terminal, a file, or another system. A single data set, managed by VSAM, is used to hold the data for all intrapartition queues.

You can specify a transaction to process the records and a trigger level for each intrapartition queue. The trigger level represents a number of records that are allowed to accumulate before the specified transaction is initiated. See page 278 for more information about trigger levels.

The intrapartition queue can be defined as logically recoverable, physically recoverable, or not recoverable.

A logically recoverable queue is restored (after an individual transaction failure or a total system failure) to the status it had at the end of the last completed unit of work (UOW). (A UOW begins at start of task or at a syncpoint, and ends at end of task or at a syncpoint).

Physically recoverable queues are restored (after a total system failure) to the statuses they had when the system failure occurred.

TYPE=REMOTE cannot be specified on the TDQUEUE resource definition. If you want to define a remote transient data queue, leave the TYPE attribute blank and specify values for the remote attributes, REMOTELENGTH, REMOTENAME, and REMOTESYSTEM. Alternatively, you can include the remote attributes as part of the resource definitions for the other transient data queue types. See “Dual-purpose resource definition for transient data” on page 270 for further information.

TYPEFILE({INPUT|OUTPUT|RDBACK})
specifies the type of data set the queue is to be associated with.

INPUT
An input data set.

OUTPUT
An output data set.

RDBACK
An input data set that is to be read backward.

Note: This is appropriate only for data sets that have been defined on magnetic tape.

An extrapartition queue can be input or output, but not both.

For more information about the DCB macro fields, see the *DFSMS/MVS Macro Instructions for Data Sets*.

USERID=(*name*) (intrapartition queues only)
specifies the userid you want CICS to use for security checking when verifying the trigger-level transaction specified in the TRANSID field.

The value entered in the USERID field is valid only when ATIFACILITY(FILE) is also specified.

When security is active, the trigger-level transaction runs under the authority of the specified userid. This userid must be authorized to all the resources used by the trigger-level transaction.

If you omit the userid from a transient data queue definition, CICS uses the CICS default userid, specified on the DFLTUSER system initialization parameter. Security checking takes place when you are installing an intrapartition definition containing a userid. If the security check fails, the resource definition for that intrapartition queue is not installed.

For further information about surrogate user security, see the *CICS RACF Security Guide*.

WAIT(YES|NO) (Intrapartition queues only)
specifies whether an in-doubt unit of work (UOW) that has modified a logically recoverable queue should wait for resynchronization with its coordinator to determine whether to commit or back out the changes.

NO The UOW is not to wait. Any changes made to recoverable resources are to be backed out or committed, as specified by the ACTION attribute on the TRANSACTION resource definition.

YES The UOW is to wait, and any action required while waiting is determined by the WAITACTION attribute.

This attribute overrides the WAIT attribute defined on the UOW’s transaction definition. See Table 16 on page 328 for an explanation of the interactions of in-doubt attributes on the TDQUEUE and TRANSACTION definitions.

WAITACTION(REJECT|QUEUE) (Intrapartition queues only)
specifies the action CICS is to take for an in-doubt unit of work (UOW) if the definition for this queue specifies WAIT(YES). The possible actions are:

QUEUE
The UOW is in-doubt and waiting; any locks held by the UOW for this queue remain active until the final state of the UOW is known. This means that tasks are suspended rather than receiving the LOCKED response. When the final state of the UOW is known, any changes that it has made are committed or backed out. Until then, any further requests of the following types that need one of the active locks must wait:

- READQ, if the in-doubt UOW had issued READQ or DELETEQ requests.
- WRITEQ, if the in-doubt UOW had issued WRITEQ or DELETEQ requests.
- DELETEQ, if the in-doubt UOW had issued READQ, WRITEQ or DELETEQ requests.

REJECT
The UOW is in-doubt and is waiting. Any lock held by the UOW for this queue is retained until the final state of the UOW is known. When the final state is known, any changes the UOW has made are committed or backed out. Until then, any further request that needs one of the retained locks is rejected, and a LOCKED response is returned. WAITACTION=REJECT causes LOCKED to be raised in exactly the same circumstances as those in which QUEUE causes a transaction to wait.

Required entries in the DCT

CICS needs certain definitions for use by some of its transactions. You can find them in the sample group DFH$DCTG, in DFHLIST, and you can install them during cold start. To make these resources available as early as possible, this should be the first group installed as part of this process, if DFHLIST is not used.

The queue definitions are:

DESTID=CADL (needed to log VTAM resource definitions)
For VTAM resources, this destination keeps a log of each RDO definition installed in the active CICS system. The log records both the installation of entries in the TCT, and the deletion of autoinstalled entries from the TCT. It records definitions installed:

- By autoinstall
- Using CEDA INSTALL
- At system initialization

For details about defining CADL and CSDL, see the *CICS System Definition Guide*.

DESTID=CAFF (needed for the Transaction Affinities Utility)
CAFF is the destination for Transaction Affinities Utility messages, allowing the Detector component of the utility to maintain an audit trail of its activities.

DESTID=CAIL (needed to log autoinstall terminal model definitions)
The autoinstall terminal model manager (AITM) uses this destination to log all autoinstall terminal model entries installed in, and deleted from, the TCT.

DESTID=CCPI (needed for CPI Communications messages)
The common programming interface for communications (CPI Communications) writes messages to this destination.

DESTID=CCSE (needed for C language support)
Under CICS, C/370 directs the C standard streams to transient data queues. (Queue names are fixed in CICS: thus, the C standard streams cannot be redirected to other queues.) C programs write to the CCSE queue by writing to stderr. You may code this destination as extrapartition, intrapartition, or indirect. If you do not provide a DCT entry for the CCSE queue, writing to stderr in C programs will fail.

DESTID=CCSI (optional, for C language support)
The CCSI queue is reserved for stdin, the C standard stream for input data. Although the CCSI queue name is reserved for stdin, any attempt to read from stdin in CICS results in EOF being returned. For this reason, this destination has been described as an "optional" entry in the DCT. You may code it as extrapartition, intrapartition, or indirect.

DESTID=CCSO (needed for C language support)
CCSO is associated with stdout, the C standard stream for output data. You may code this destination as extrapartition, intrapartition, or indirect. If you do not provide a DCT entry for the CCSO queue, writing to stdout in C programs will fail.

The copybook DFH$DCTR has an entry provided for both output queues directed to //COUT. If you use the CICS-supplied sample definitions as the basis for your own DCT, ensure that you add a DD statement for the COUT data set to your CICS start-up job streams.

DESTID=CDBC (needed for DBCTL DFHDB81xx messages)
CDBC is defined as an indirect queue, which points to the CSML extrapartition queue. Only DBCTL DFHDB81xx messages use this data log; other messages use either the terminal or the console.

DESTID=CDUL (needed for transaction dump messages)
CDUL is the destination for transaction dump messages. If a transaction dump is requested, for example after a transaction abend, a message is written to this destination to show that a dump has been taken or to give a reason why the dump was suppressed.

DESTID=CESE (needed for run-time output from Language Environment)
For Language Environment, all run-time output is written to this transient data queue. For further information, see the *OS/390 Language Environment Programming Guide*.

DESTID=CMIG (needed for migration log)
CMIG is a **migration log**, which receives messages reporting the use of functions that are no longer supported in CICS (for example, the EXEC CICS ADDRESS CSA command). You can define CMIG as an intrapartition, extrapartition, or indirect destination.

DESTID=CPLI, DESTID=CPLD (both needed for CICS PL/I support)
CPLI is the destination for SYSPRINT output, and CPLD for PLIDUMP output. The minimum logical record size is 137 for CPLI and 125 for CPLD. If these destinations are extrapartition (direct or indirect), they must be V format. See the installation manual for your PL/I compiler for more details.

DESTID=CRDI (needed to log program resource definitions)
This destination provides a log of installed resource definitions for programs, transactions, maps, and mapsets.

DESTID=CSCS (needed for the sign-on transaction)
CSCS receives a message giving details of each sign-on and sign-off. It also receives a message about each rejected attempt at sign on and each resource authorization failure. This destination can be of any type.

DESTID=CSDL (needed to log RDO commands)
The resource definition online (RDO) transactions write to this destination all commands that result in changes to the CICS system definition (CSD) file or active CICS system.

You need CSDL only if you use RDO and want to keep a log of commands.

The maximum length of data records written to CSDL is 128 bytes. If you define CSDL as extrapartition, the associated SDSCI or DD statement should specify V format records with a minimum blocksize of 136 bytes.

DESTID=CSFL (needed to log file resource definitions)
CSFL is a log of all file resource definitions installed in the active CICS system. Deletions of file resource entries are also logged here.

DESTID=CSKL (needed to log transaction and profile resource definitions)
CSKL is a log of all transaction and profile resource definitions installed in the active CICS system. Deletions are also logged here.

DESTID=CSML (needed for the sign-off transaction)
CICS sign-off writes data to this destination.

DESTID=CSMT (needed for terminal error and abend messages)
The terminal abnormal condition program (DFHTACP) and abnormal condition program (DFHACP) write terminal error and ABEND messages, respectively, to this destination. You may code this destination as extrapartition, intrapartition, or indirect.

DESTID=CSNE (needed for node error messages)
The node abnormal condition program (DFHZNAC) and the node error program (DFHZNEP) write terminal error messages and data to this destination. You can code this destination as extrapartition, intrapartition, or indirect.

DESTID=CSPL (needed to log program resource definitions)
CSPL is a log of all program resource definitions installed in the active CICS system. Deletions are also logged here.

DESTID=CSRL (needed to log partner resource definitions)
CSRL is a log of all partner resources installed in the active CICS system. Deletions are also recorded here. For more information about partner resources, see “PARTNER” on page 229.

DESTID=CSSL (needed for recovery utility statistics)
The recovery utility program (DFHRUP) writes statistics to this destination. This destination needs a minimum logical record length of 132 bytes and a minimum blocksize of 136 bytes.

DESTID=CSTL (needed for terminal I/O error messages)
The terminal abnormal condition program (DFHTACP) writes terminal I/O error messages to this destination. You may code this destination as extrapartition, intrapartition, or indirect.

DESTID=CSZL (needed for the Front End Programming Interface)
If you have installed the CICS FEPI feature, CSZL is used as the destination for FEPI messages. For information on FEPI transient data destinations, see the *CICS Front End Programming Interface User's Guide*.

DESTID=CSZX (needed for the Front End Programming Interface)
If you have installed the CICS FEPI feature, CSZX is intended for use with a triggered transaction. For information on FEPI transient data destinations, see the *CICS Front End Programming Interface User's Guide*.

TERMINAL

CICS requires a definition for each terminal device with which it communicates. Terminal devices include visual display units, printers, operating system consoles, and more specialized devices such as facsimile (FAX) machines.

The unique and possibly dynamic properties of terminals are defined in the TERMINAL definition in the CSD file.

However, many of your terminals have identical properties, and you do not need to define each of them separately and fully to CICS. There are two ways you can reduce the time and effort needed to define each terminal. They are:

1. **TYPETERM definitions**, with or without the QUERY function. Each TERMINAL definition must refer to a TYPETERM definition that defines the properties that are common, often more complex, and usually static. Together, information from the TERMINAL and TYPETERM definitions makes up a terminal entry in the TCT (a TCTTE).

   One TYPETERM can represent a lot of the properties of many terminals. Some of these properties can be left undefined at the time of creating the TYPETERM definition. These properties can be determined at logon time for each terminal, from the QUERY structured field.

   There are, however, still more properties that many terminals have in common, to the extent that their TERMINAL definitions would all be identical. CICS provides a facility that avoids the need for each terminal to have its own resource definition installed in the TCT the whole time CICS is active.
2. **Autoinstall**, using one TERMINAL definition to represent many terminals. You can let CICS create and install the resource definition dynamically when the terminal is needed, at logon time. To do this, CICS uses a **model TERMINAL definition** from the CSD file. This process is known as automatic installation, or autoinstall.

   Autoinstall reduces the virtual storage required for the terminal control table (TCT) if some of your terminals are not logged on when CICS is active.

   If you are involved in planning for and managing CICS communications resources such as terminals, read “Chapter 8. Autoinstall for VTAM terminals” on page 121 for further information.

Installing TERMINAL definitions

For groups of TERMINAL definitions, use the following procedure:

1. Make sure that nobody is using the terminals that you want to install. Remember that CEMT INQUIRE TERMINAL puts a lock on the TCT entry, and this also prevents installation of a group containing that terminal.
2. Make sure that there are no ATIs outstanding for the terminals.
3. Use CEMT, with a generic name if appropriate:
   ```
   CEMT SET TERMINAL(trmid) RELEASED OUTSERVICE NOATI
   ```
4. Install the resource definitions:
   ```
   CEDA INSTALL GROUP(groupname)
   ```
5. Use CEMT to make the terminals available again:
   ```
   CEMT SET TERMINAL(trmid) ACQUIRED INSERVICE ATI
   ```

TERMINAL and TYPETERM definitions are resolved during installation to become a TCT entry, sometimes known as a TCT terminal entry or TCTTE. Each definition contains only some of the information necessary to build a TCT entry.

Each TYPETERM definition must be installed before or at the same time as the TERMINAL definitions that reference it. When you are using CEDA to install groups, if the TYPETERMs are in a separate group from the TERMINALs, you must install the TYPETERMs group before the TERMINALs. You may include TYPETERMs and TERMINALs in the same group for testing purposes, although it is not recommended for long-term use. (See "What should be in a group?" on page 11.)

Because the TERMINAL definition is not necessarily in the same group as its associated TYPETERM definition, the global catalog is used to store the TYPETERMs.

Changing a TYPETERM definition and then reinstalling it has no effect on an already installed terminal entry, even though the TERMINAL definition used to create it refers to the TYPETERM. To change the terminal entry, you must reinstall both the TYPETERM and the TERMINAL.

Installing autoinstall model TERMINAL definitions: If you define a model TERMINAL for autoinstall, you install it just as you would an ordinary resource definition; that is, by installing the group containing it. However, TERMINAL definitions specified as AUTINSTMODEL(ONLY) are stored only in the autoinstall model table (AMT) at this time; they do not result in a TCTTE on the active CICS system. These model TERMINAL definitions are stored in the AMT until needed by CICS to create a definition for an actual terminal logging on through VTAM using autoinstall. The resulting TERMINAL definition is "automatically installed" at this time.

This is what happens when you install a TERMINAL definition, either at system initialization or using INSTALL:

AUTINSTMODEL(NO)
A TCT entry is created for the terminal.

AUTINSTMODEL(YES)
A TCT entry is created for the terminal, and the model definition is stored for later use by the autoinstall process.

AUTINSTMODEL(ONLY)
The model definition is stored for later use by the autoinstall process.

If you install two model TERMINAL definitions with the same AUTINSTNAME, the second one replaces the first.

For further information about autoinstall and model TERMINAL definitions, see "Chapter 8. Autoinstall for VTAM terminals" on page 121.

Types of terminals

Different types of terminal have different characteristics, and the way you use them may vary, too. Here is some guidance on defining terminals that are used in particular ways:

- Terminals for printing
- Pipeline terminals for VTAM pooled sessions
- Devices with LDC lists
- APPC single-session terminals

Terminals for printing

A TERMINAL definition for a display device can name TERMINAL definitions for printers, using the PRINTER and ALTPRINTER attributes. Such a reference is not resolved when the TERMINAL definitions are installed. Instead, the reference is resolved when the printer is needed by the display device.

There are several ways in which printed output can be created and sent to a printer.

- BMS page building
- Screen copying, using one of the following:
  - A hardware copy key
  - A local copy key
  - The ISSUE PRINT command

For programming information about creating output by these methods, see the *CICS Application Programming Reference* manual.

The TYPETERM and TERMINAL definitions are used for both printers and display devices. A number of attributes apply only to printers, or have special meanings for printers. There are also some attributes that you need to specify for a display device that is to be used for screen-copying.

Printers: Supply a TERMINAL definition for each printer. Specify NO for AUTINSTMODEL, unless you are using autoinstall for printers. (For more information about this, see "Autoinstall and output-only devices" on page 126.)

ALTPAGE

For BMS, the PAGESIZE attribute determines the default page size, and also the size of the print buffer. You specify the number of lines in the page (the length) and the number of characters in each line (the width).

Another attribute, ALTPAGE, indicates the page size to be used when the alternate screen size (ALTSCREEN) is selected. The width you specify in ALTPAGE must be the same as the width specified in the ALTSCREEN attribute. However, the length of ALTPAGE and ALTSCREEN can be different. This could be useful if you are using the same BMS map to display and to print. For instance, you could make the screen one line longer than the page, to reserve the bottom line of the screen for error messages.

The ALTPAGE, DEFSCREEN, and ALTSCREEN attributes do not normally apply to printers.

AUTOPAGE

AUTOPAGE must be YES for printers, but you do not need to worry about it, because RDO fills it in for all printer DEVICE types. Autopaging means that BMS multiple page messages are printed continuously, without operator intervention. This is what is normally required for a printer. (Contrast the requirement for multiple page messages, displayed on a 3270-type display, when the operator wants to finish reading a page before requesting the next page.)

Only BMS SEND commands with the PAGING option use autopaging. BMS SEND with TERMINAL or SET does not use autopaging.

You need at least one TYPETERM definition for each type of printer you use. You may need more, if you want to allow printers to be used only for some functions and not for others.

DEVICE

The TERMINAL definition for each printer must refer to a TYPETERM with an appropriate DEVICE type. The DEVICE attribute and, in one case, the

SESSIONTYPE attribute, determine whether a TYPETERM defines printers or display devices. The values that you can specify for printers are:

DEVICE	SESSIONTYPE	Printers
3270P	-	All printers that support the 3270 data stream (not SNA-connected).
LUTYPE3	-	All printers that support the 3270 data stream (SNA-connected).
SCSPRINT	-	All printers that support the SNA character set (SNA-connected).
3790	SCSPRINT	IBM 3793 keyboard-printers that support the SNA character set (SNA-connected).

FORMFEED
Define FORMFEED as YES for BMS page building.

PAGESIZE
specifies the default page size for this terminal. The product of lines and columns must not exceed 32767, where lines = the number of lines in the page, and columns = the number of characters in each line.

If PGESIZE is not coded, the following defaults are used:

TW33, TW35	(12,80)
3270 display model 1	(12,40)
3270 display model 2	(24,80)
3270 printer	(12,80)

TERMINAL
The name of the printer is the TERMINAL name on the resource definition for that printer.

Note: The PRINTER attribute is used on a display device definition to refer to a printer device to be used for output from the display. Do not specify PRINTER on the printer definition. See “Associating printers with display devices”.

Associating printers with display devices: If you want to copy the contents of a screen direct to a printer, associate a specific printer with the display device in the TERMINAL definition. You do this using the PRINTER and ALTPRINTER attributes.

ALTPRINTCOPY
Specify YES for ALTPRINTCOPY if you want to use the hardware COPY feature on the alternative printer.

For programming information about screen copying, see the *CICS Application Programming Reference* manual.

ALTPRINTER
The alternative printer is used if the primary printer is unavailable. ALTPRINTER and ALTPRINTCOPY can be set dynamically by the autoinstall control program, if the display device definition is autoinstalled.

PRINTER
The primary printer is to be used. PRINTER and PRINTERCOPY can be set dynamically by the autoinstall control program, if the display device definition is autoinstalled.

PRINTERCOPY
Specify YES for PRINTERCOPY if you want to use the hardware COPY feature on the primary printer.

Pipeline TERMINAL definitions

When you define a 3600 pipeline logical unit, you generate a TCTTE that is associated with a pool of TCTTEs. As messages enter CICS from the 3600 pipeline logical unit, a task is attached to process this message, using as an anchor block one of the TCTTEs from the pool. In this way, consecutive messages sent via the pipeline logical unit can be processed concurrently, with the number of concurrent transactions being limited by the number of TCTTEs in the pool. The number of TCTTEs in the pool should represent the high-water mark of inquiry activity. In this way, the pipeline facility allows fewer TCTTEs to be defined to CICS than the total number of pipeline inquiry terminals.

All the TERMINAL definitions within a named POOL must be in the same group in the CSD file. TASKLIMIT must have been specified on at least one of the definitions in the group. If it is specified on more than one definition, the value used is the maximum value of TASKLIMIT over the definitions in the group.

TERMINAL and TYPETERM definitions are resolved as for ordinary terminals.

If you do not install PIPELINE terminals by GROUP, the results can be unpredictable.

Defining pipeline terminals: Define pipeline terminals with the following attributes:

NETNAME
The VTAM session that is used.

POOL
All the TERMINAL definitions having the same POOL name belong to the same pipeline pool. The presence of a value for the POOL attribute distinguishes these from ordinary TERMINAL definitions.

SESSIONTYPE
Use this attribute on the TYPETERM definition to identify the TYPETERM as representing pipeline terminals. Specify PIPELINE as the value.

TASKLIMIT
Specify the maximum number of concurrent tasks that can be active for the pool of terminals on at least one of the TERMINAL definitions.

One TYPETERM would normally suit all the definitions. The TYPETERM may be in another group.

Pipeline transactions are associated with a PROFILE definition that has the ONEWTE attribute. A program associated with these transactions is permitted only one write or EXEC CICS SEND operation, or else it is terminated with an ATCC abend code. This means that CICS tasks rapidly appear and disappear across the pool of sessions.

There is an example of definitions for a pool of pipeline terminals on page 305.

Notes:

1. If you install a pipeline terminal naming a pipeline that already exists in CICS, both the old pipeline and all its related terminals are deleted before the new definitions are installed.
2. If you discard the terminal that is defined as owning the existing pipeline, the existing pipeline and all its related terminals are deleted.
3. If you discard a terminal that is not the pipeline owner or change it to a different pipeline, or to a nonpipeline terminal, the rest of the pipeline definition is unchanged. (The owning terminal of a pipeline is the terminal with the first name in alphanumeric sequence that is related to the pipeline in the group from which the pipeline was installed.)
4. You cannot change a terminal in an existing pipeline to a nonpipeline terminal, or change it to a new pipeline, if the old pipeline is also being reinstalled in the same group. To do this, you divide the installation into two stages. If you are installing the group twice, remember to set the relevant terminals out of service in the meantime.

Devices with LDC lists

For 3600, 3770 batch, 3770, and 3790 batch data interchange, and LUTYPE 4 logical units, you can specify the name of an LDC (logical device code) list. The list specifies which LDCs are valid for this logical unit and, optionally, which device characteristics are valid for each LDC. The first LDC in this list is the default when CICS must choose a default LDC for a logical unit.

All the TERMINAL definitions for devices that reference a particular LDC list must name the same TYPETERM, because the LDCLIST attribute is on the TYPETERM definition.

The LDC list itself must be defined using a DFHTCT TYPE=LDCLIST macro. It may be a local LDC list or an extended local LDC list. There is an example of the coding for each on page 351, and further guidance in “DFHTCT logical device codes: VTAM non-3270” on page 444. You use the label coded on the macro instruction to identify the LDC list on the TYPETERM definition, specifying it in the LDCLIST attribute.

APPC (LUTYPE6.2) single session terminal

An APPC (LUTYPE6.2) single session terminal can be defined as a TERMINAL, with a reference to a TYPETERM with DEVICE(APPC). When these definitions are installed, the resources they define are known to CICS as a connection and a modeset, just as they are when defined in RDO as CONNECTION and SESSIONS. The name of the connection is the TERMINAL name and the name of the modeset is the MODENAME on the TERMINAL definition.

An APPC terminal may be a PS/2, an Application System/400® (AS/400®), a System/38, or similar. You can define an APPC terminal either by a TERMINAL-TYPETERM definition or by a CONNECTION-SESSIONS definition. Both kinds of definition can be autoinstalled (see “Chapter 8. Autoinstall for VTAM terminals” on page 121 and “Chapter 10. Autoinstall for APPC connections” on page 141 for information about autoinstall). If you decide to use the TERMINAL-TYPETERM method, the following attributes are important:

DEVICE
The TERMINAL definition references a TYPETERM with APPC (advanced program-to-program communications) specified as the DEVICE. One such TYPETERM definition suffices for many terminals.

MODENAME
This is the name that CICS uses to identify the session when the definition is installed in the active system.

Terminals for transaction routing

Transaction routing enables terminals in one CICS system to invoke transactions in another CICS system.[4] You can use transaction routing between systems connected by MRO or by an APPC link. (See the *CICS Intercommunication Guide* for further information.) If you use transaction routing, there are three possible methods of defining the terminals.

When a terminal belonging to (local to and fully defined in) one system invokes a transaction belonging to another system, it is known to the application-owning region as a **remote terminal**. The application-owning system needs to have access to at least a partial definition of the remote terminal. This partial definition is often known as a **remote definition**.

The terminal-owning system has a complete definition of the terminal: this is known as a **local definition**.

Making a partial definition of a terminal available to the application-owning CICS system(s) can be done in one of the following ways:

- Having a separate terminal definition for each system involved, each on a CSD file that is accessible to that system. One is created as a local definition and one or more are created as remote definitions. This is referred to as **duplicating terminal definitions**, because there is more than one resource definition for the same terminal (the definitions are not necessarily exact duplicates of each other).
- Having one dual-purpose definition in one, shared CSD file available to all the systems involved, both terminal-owning and application-owning. This method is known as **sharing terminal definitions**.
- Having, in the CSD file available to the terminal-owning system, one **local** terminal definition that can be shipped to other systems when required. This method is known as **shipping terminal definitions**.

These three different methods of making a partial definition available, involve three different ways of using the TYPETERM and TERMINAL definitions:

- "Local definition"
- "Remote definition" on page 292
- "Dual-purpose definition" on page 292

Local definition

This is a full definition of the terminal, installed in the terminal-owning system. It is used in the **duplicating** and in the **shipping** methods.

- No REMOTESYSTEM, REMOTESYSNET, or REMOTENAME is needed on the TERMINAL or CONNECTION definition.

  If the REMOTESYSTEM named on the TERMINAL definition is actually the name of the local system (as specified by the SYSIDNT system initialization operand), the definition is treated as a local terminal when it is installed, rather than a remote one.
- SHIPPABLE on the TYPETERM is NO if duplicating definitions.
- SHIPPABLE on the TYPETERM is YES if shipping definitions.
- SHIPPABLE on the TYPETERM is obligatory for APPC devices.

4. In this section, the word "system" means a CICS system communicating with other systems using MRO or ISC.

- All other necessary attributes on the TERMINAL and TYPETERM definitions must be specified.

Remote definition

This is a partial definition of the terminal, installed in the application-owning region and intermediate regions. It contains the minimum information necessary for the terminal to access a transaction in that system. You create remote definitions only if you are using the **duplicating** method.

- The REMOTESYSTEM name must be the name of the CONNECTION definition for the next region in the transaction routing path to the terminal-owning region.
- REMOTESYSNET must be the netname (generic applid) of the terminal-owning region. If REMOTESYSTEM names a direct CONNECTION to the terminal-owning region, REMOTESYSNET is not required unless the terminal-owning region is a member of a VTAM generic resource group, and the direct connection is an APPC link.
- The REMOTENAME is the name by which the terminal or APPC device is known by in the terminal-owning region. It may be the same as or different from the TERMINAL or CONNECTION name, which must be the name the terminal is known by in the application-owning system. REMOTENAME defaults to the TERMINAL name if not specified.
- SHIPPABLE on the TYPETERM is NO for non-APPC devices.
- SHIPPABLE on the TYPETERM is obligatory for APPC devices.
- Some of the attributes on the TERMINAL and TYPETERM definitions may be omitted. For further guidance about these definitions, see the *CICS Intercommunication Guide*.

The following terminals and logical units cannot use transaction routing and therefore cannot be defined as remote:
- Pooled 3600 or 3650 pipeline logical units
- IBM 2260 terminals
- The MVS console
- Pooled TCAM terminals

Dual-purpose definition

This is a full definition of the terminal, installed in the terminal-owning system and in the application-owning and intermediate regions. It is used in the **sharing** method only.

- The REMOTESYSTEM name must be the SYSIDNT of the terminal-owning region. This must also be the name of the CONNECTION that you define from the application-owning region to the intermediate region (if any), and from the intermediate region to the terminal-owning region. All the CONNECTION definitions on the path from the application-owning region to the terminal-owning region must be given this same name.
- REMOTESYSNET must be the netname (generic applid) of the terminal-owning region.
- The REMOTENAME is the same as the TERMINAL or CONNECTION name.
- SHIPPABLE on the TYPETERM is NO for non-APPC devices.
- SHIPPABLE on the TYPETERM is obligatory for APPC devices.
- All other necessary attributes on the TERMINAL and TYPETERM definitions must be specified.

APPC devices for transaction routing

APPC transaction routing allows an APPC device belonging to one CICS system to invoke transactions in another CICS system. The three methods of defining a terminal for transaction routing, described in “Terminals for transaction routing” on page 291, are also applicable to APPC devices. These methods can involve the use of either TERMINAL-TYPETERM or CONNECTION-SESSION definitions. For APPC single session terminals, the TERMINAL-TYPETERM definition method is recommended (see “APPC (LUTYPE6.2) single session terminal” on page 290.)

Duplicating terminal definitions

1. Create a local definition for the terminal, in the CSD file of the terminal-owning system, or on a shared CSD file.
2. Create a remote definition for the terminal, in the CSD file of the application-owning system, or in a shared CSD file. If you have more than one application-owning system, you may need more than one remote definition, but if the systems are sharing a CSD file, you may be able to use the same remote definition for them all.
3. If your systems share a CSD file, make sure the definitions are in different groups, because:
   - You want to install them in different systems
   - The TERMINAL names are probably the same, so the definitions cannot be in the same group.
4. Install the local definition in the terminal-owning system. This definition can be autoinstalled.
5. Install the remote definition in the application-owning system(s).

See Figure 40 on page 294 for an illustration of this process.

Advantages:

- You can use this method whether or not your systems share a CSD file, so you can use it for transaction routing between different MVS images.
- This is the **only** method you can use if your terminals are known by different names in different systems.
- You can use automatic transaction initiation (ATI) in the application-owning system for a remote terminal without having to set up XALTENF and XICTENF exits. (See the *CICS Customization Guide* for programming information on these exits.) This is especially useful for printers, because they must be acquired **before** any ATI can be successful.

Disadvantages:

- The CSD file uses at least twice the necessary amount of disk storage for your definitions.
- The TCT uses more than the necessary amount of virtual storage for your definitions.
- Unnecessary work is involved in maintaining the definitions.
- You can autoinstall the terminal in the terminal-owning system, but you cannot autoinstall it in the application-owning systems.

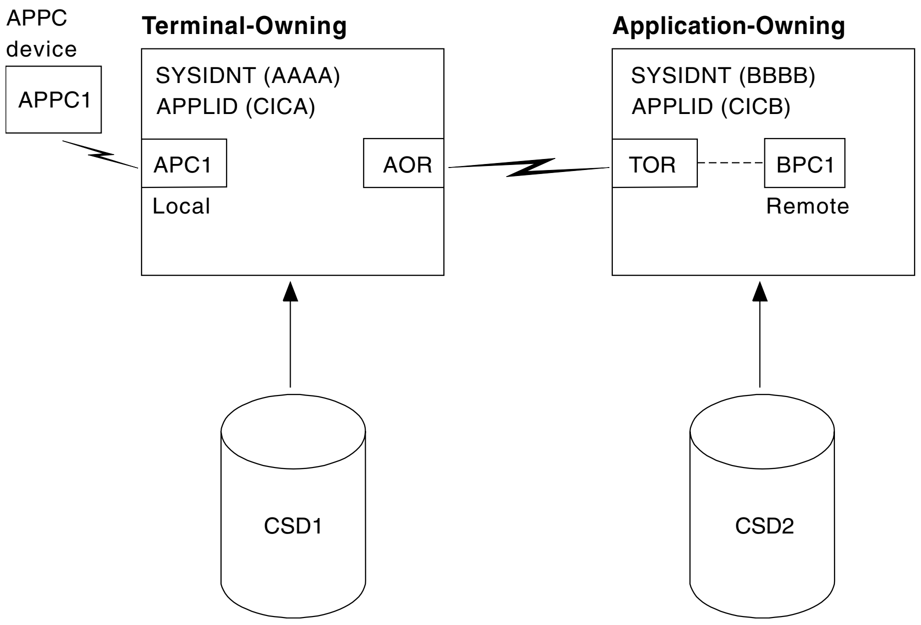


Figure 40. Maintaining local and remote definitions separately. The definitions may be on the same CSD file.

Connections	Definitions in the AOR	Definitions in the TOR

Connection between the systems	CONNECTION(TOR)
	NETNAME(CICA)
		CONNECTION(AOR)
		NETNAME(CICB)

Connection to APPC device, using TERMINAL and TYPETERM	TERMINAL(BPC1)
	TYPETERM(APPC0001)
	NETNAME(APPC1)
	REMOTESYSTEM(TOR)
	REMOTENAME(APC1) and
	TYPETERM(APPC0001)
	DEVICE(APPC)
		TERMINAL(APC1)
		TYPETERM(APPC0001)
		NETNAME(APPC1) and
		TYPETERM(APPC0001)
		DEVICE(APPC)

Connection to APPC device, using CONNECTION and SESSIONS	CONNECTION(BPC1)
	NETNAME(APPC1)
	REMOTESYSTEM(TOR)
	REMOTENAME(APC1)
		CONNECTION(APC1)
		NETNAME(APPC1) and
		SESSIONS(APPC0001)
		CONNECTION(APC1)

Note: The REMOTESYSNET option is not required because there is a direct connection between the AOR and the TOR.		

Sharing dual-purpose definitions

1. Create a dual-purpose definition for the terminal, in the shared CSD file.
2. Install the definition in all the systems: it becomes a remote definition in a system whose SYSIDNT is different from the REMOTESYSTEM name, and a local definition in a system whose SYSIDNT is the same as the REMOTESYSTEM name.

See Figure 41 on page 295 for an illustration of this process.

Advantages:

- You can use automatic transaction initiation (ATI) in the application-owning system for a remote terminal without having to set up XALTENF and XICTENF exits. (See the *CICS Customization Guide* for programming information on these exits.) This is especially useful for printers, because they must be acquired **before** any ATI can be successful.

- Disk storage use is reduced because you need only one CSD file record.
- Maintenance is reduced because you need only one CSD file record.

Disadvantages:

- The TCT uses more than the necessary amount of virtual storage for your definitions.
- You cannot use this method if your systems do not share a CSD file, so you can use it only for transaction routing within the same MVS image.
- You cannot use this method if your terminals are known by different names in different systems.

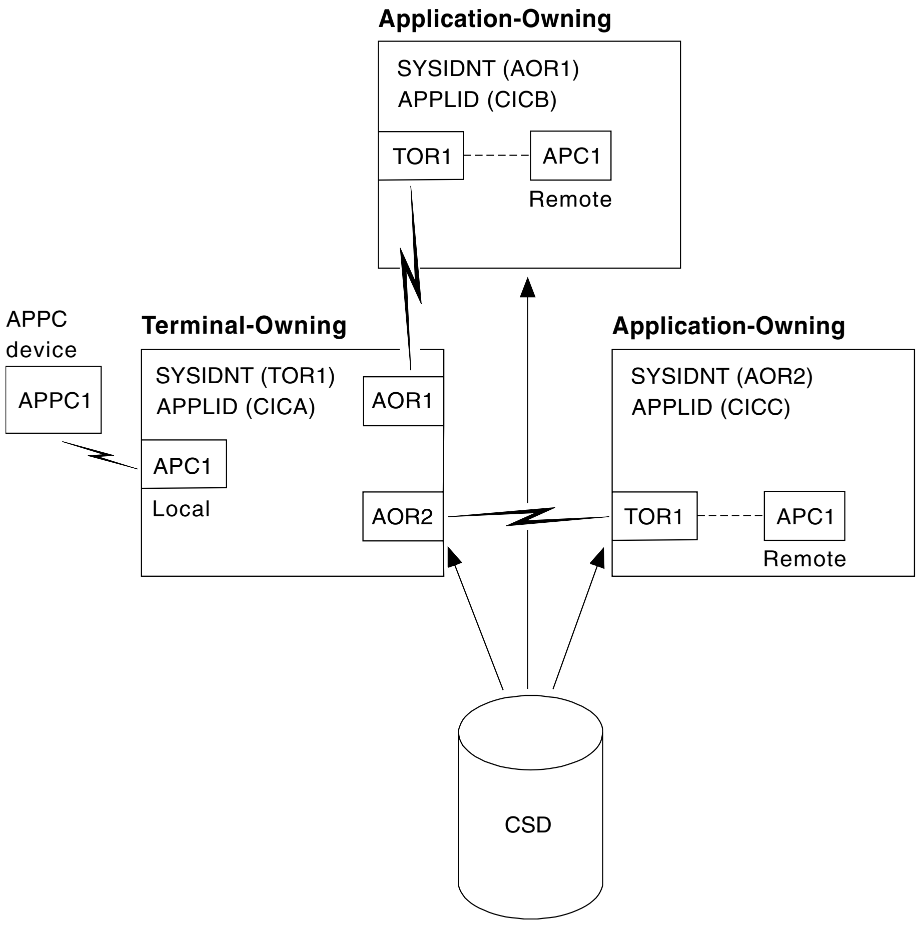


Figure 41. Sharing dual-purpose definitions

Connections	Definitions in the AORs	Definitions in the TOR

Connection between the systems	CONNECTION(TOR1)
	NETNAME(CICA)
		CONNECTION(AOR1)
		NETNAME(CICB)

Connection to APPC device, using TERMINAL and TYPETERM	-	TERMINAL(APC1)
		TYPETERM(APPC0001)
		NETNAME(APPC1)
		REMOTESYSTEM(TOR1)
		REMOTENAME(APC1) and
		TYPETERM(APPC0001)
		DEVICE(APPC)

Connection to APPC device, using CONNECTION and SESSIONS	-	CONNECTION(APC1) NETNAME(APPC1) REMOTESYSTEM(TOR1) REMOTENAME(APC1) and SESSIONS(APPC0000) CONNECTION(APC1)
Note: The REMOTESYSNET option is not required because there are direct connections between the AOR and the TOR.		

Shipping terminal definitions to application-owning systems

1. You create a local definition for the terminal, in the CSD file of the terminal-owning system. (This can be a shared CSD file.)
2. You install the local definition in the terminal-owning system. (This definition can be installed at system initialization, or by using CEDA INSTALL, or by autoinstall.)
3. When the terminal invokes a transaction belonging to another system, the information necessary to create a remote definition is shipped to that system, and a temporary definition is installed there automatically.

If the local definition was autoinstalled, the shipped definition lasts until the terminal is logged off. Otherwise, the shipped definition lasts until the local definition is installed again, or until the link between the systems is broken.

See Figure 42 on page 297 for an illustration of this process.

Advantages:

- You can use this method whether or not your systems share a CSD file, so you can use it for transaction routing between different MVS images.
- Disk storage use is reduced because you need only one CSD file record.
- Maintenance is reduced because you need only one CSD file record.
- Virtual storage use is reduced because you install the definition in only one system.
- You can autoinstall the terminal in the terminal-owning system, and **in effect** autoinstall it in the application-owning systems. (Shipping terminal definitions has the same effect as autoinstall, but does not itself involve the autoinstall process.)

Disadvantages:

- You cannot use this method if your terminals are known by different names in different systems.
- You may need to use the global exits XALTENF and XICTENF if you use ATI in the transaction owning system. See the *CICS Customization Guide* for programming information on these exits.

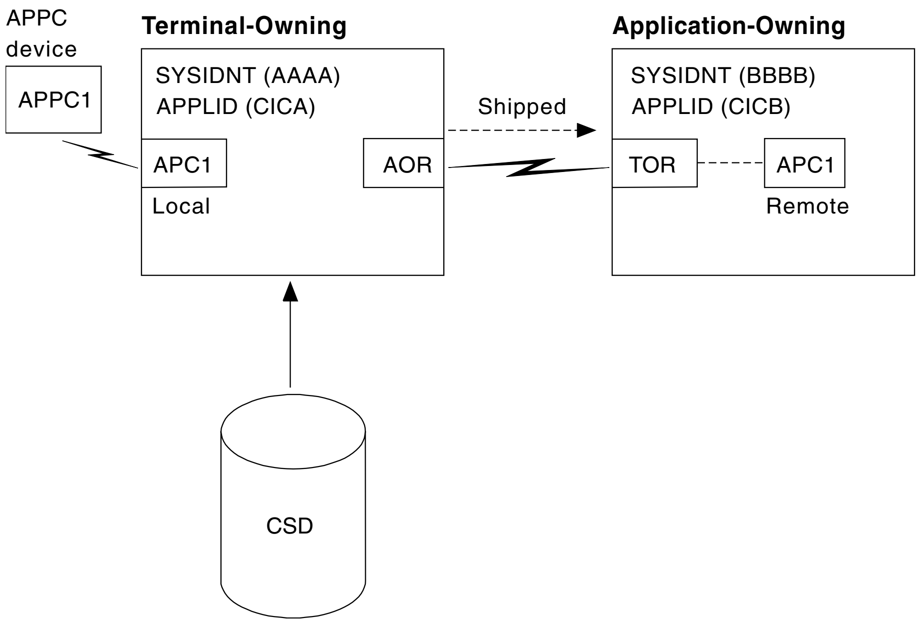


Figure 42. Shipping local definitions to an application-owning system

Connections	Definitions in the TOR
Connection to APPC device, using TERMINAL and TYPETERM	TERMINAL(APC1) TYPETERM(APPC0001) NETNAME(APPC1) and TYPETERM(APPC0001) DEVICE(APPC) SHIPPABLE(YES) (SHIPPABLE is mandatory for device type APPC)
Connection to APPC device, using CONNECTION and SESSIONS	CONNECTION(APC1) NETNAME(APPC1) and SESSIONS(APPC0000) CONNECTION(APC1)

Transaction routing—summary

Consider carefully these methods of defining terminals for transaction routing, and decide which is most suitable for your network before you start to use RDO.

Use the **shipping method**, unless you use terminals that are known by different names in different systems. For ATI to work with the shipping method in a transaction owning system, you may need to use the XALTENF and XICTENF global exits. See the *CICS Customization Guide* for programming information on these exits.)

If you use the same names in different systems and you do not want to use global exits to ensure that ATI works, use the **sharing method** for systems with a shared CSD file.

Use the **duplicating method** if you use terminals that are known by different names in different systems, or if you use ATI to acquire terminals but do not have a shared CSD file, and you do not want to use the XALTENF and XICTENF global user exits.

You could use a mixture of methods: perhaps shipping for display terminals, and duplicating for printers that need ATI to acquire them but without the use of the XALTENF and XICTENF global user exits.

Before you start creating definitions for intercommunication resources, see the *CICS Intercommunication Guide* for further information. There, you can find useful examples of the attributes you must specify for different types of links and sessions.

Extended recovery (XRF) for terminals

XRF-capable, or class 1, terminals are SNA terminals connected to both active and alternate CICS systems through a boundary network node communications controller, using NCP and VTAM with XRF capability.

Three TYPETERM attributes relate specifically to the use of XRF for terminals:

RECOVNOTIFY

This applies only to class 1 terminals. You use it to specify how the terminal user is notified when an XRF takeover has occurred. You can specify either a simple message to be displayed, or a transaction that can do more than that. In either case, all the terminals for which you specify MESSAGE get the same message, and all the terminals for which you specify TRANSACTION get the same transaction.

There are two messages, DFHXRC1 and DFHXRC2, that you may edit in mapset DFHXMSG. Alternatively you can change the name of the map passed to the node error program. You specify the transaction using the RMTRAN system initialization operand, which defaults to the 'good morning' transaction specified in GMTRAN system initialization operand.

MESSAGE is more efficient than TRANSACTION, and minimizes the takeover time.

RECOVOPTION

specifies how CICS is to recover the terminal session and return the terminal to service, after an XRF takeover. The **NONE** option leaves the terminal with no XRF support: the logon status of the terminal is not tracked by the alternate system and the terminal session is not automatically recovered after a takeover. The system default allows CICS to optimize the recovery of the terminal. However, there are circumstances in which you must specify one of the other options.

For XRF-capable, or class 1, terminals there are five options: NONE and four others. If, at takeover, a terminal is executing a transaction in the active CICS, the transaction is backed out, but the session remains busy. **SYSDEFAULT** allows CICS to clean up this busy session in the most efficient way. CICS uses one of three methods, in this order of preference:

1. Send an end-bracket (EB) indicator to close the in-bracket state.
2. Send a CLEAR request to reset all session states.
3. Send an UNBIND request to release the session.

If the terminal cannot handle an unexpected EB, specify **CLEARCONV**: CICS sends a CLEAR or an UNBIND request. This might apply if the terminal is programmable (intelligent).

If the terminal cannot handle an unexpected EB indicator or a CLEAR request, specify **RELEASESESS**: CICS sends an UNBIND request to release the session.

If the terminal must know at all times to which system it is connected, specify **UNCONDREL**: CICS sends an UNBIND request to release the session, **whether it is busy or not**. This might apply if the terminal is programmable (intelligent).

If, for any reason, an UNBIND request is sent, a new session is initiated by CICS, if the terminal is capable of performing a SIMLOGON.

For other terminals (class 2 and class 3), there is effectively a choice between NONE (no XRF support) and any one of the other options. Any option other than NONE causes the alternate system to track the logon status of the terminal and, if the terminal was logged on, to initiate a logon, and to re-establish a session, if possible.

XRFSIGNOFF
specifies the sign-on characteristics of a group of terminals. You can either allow the group to be signed on after takeover, or you can force the group to be signed off. You might choose to force sign-off to prevent unauthorized use of the terminal if it became active again after a takeover while the authorized user was absent.

There is a hierarchy of options for sign-on/sign-off.
- DFHSIT XRFSOFF=FORCE|NOFORCE
- TYPETERM XRSIGNOFF(NOFORCE|FORCE)
- CICS RACF segment

Defining a TERMINAL

```
  TErminal     ==>
   Group        ==>
   Description  ==>
  AUTINSTModel ==> No                No | Yes | Only
  AUTINSTName  ==>

 TERMINAL IDENTIFIERS
  TYpeterm     ==>
   NEtname      ==>
  CONSOle      ==> No                No | 0-99
  CONSName     ==>
  REMOTESYSTem ==>
  REMOTEName   ==>
  REMOTESYSNet ==>
  Modename     ==>

 ASSOCIATED PRINTERS
  PRINTER      ==>
  PRINTERCopy  ==> No                No | Yes
  ALTPRINTEr   ==>
  ALTPRINTCopy ==> No                No | Yes

 PIPELINE PROPERTIES
  POol         ==>
  TAsklimit    ==> No                No | 1-32767

 OPERATOR DEFAULTS
  OPERId         :
  OPERPriority   : 000               0-255
  OPERRsl        : 0                                         0-24,...
  OPERSecurity   : 1                                         1-64,...

 PRESET SECURITY
  Userid       ==>
  NAtlang      ==>

 TERMINAL USAGES
  TRansaction  ==>
  TErmpriority ==> 000               0-255
  Inservice    ==> Yes               Yes | No

 SESSION SECURITY
  Securityname ==>
  ATtachsec    ==> Local             Local | Identify | Verify | Persistent
                                     | Mixidpe
  BINDPassword ==>                   PASSWORD NOT SPECIFIED
  BINDSecurity ==> No                No | Yes
  Usedfltuser  ==> No                No | Yes
```

Figure 43. The DEFINE panel for TERMINAL

Options

ALTPRINTCOPY({NO|YES})
specifies whether the hardware COPY feature is to be used to satisfy a PRINT request on the printer named in ALTPRINTER. For further details, see the PRINTERCOPY attribute.

ALTPRINTER(*name*)
specifies the name of a 3270 printer to be used as an alternative to the PRINTER named in this TERMINAL definition, if the PRINTER is unavailable. The name may be up to four characters in length. For further details, see the PRINTER attribute. If you specify an ALTPRINTER without specifying a PRINTER, ALTPRINTER is ignored.

The printer you name must be owned by the same CICS system that owns this TERMINAL definition.

If you want to specify the hardware COPY feature for the alternative printer, specify YES for ALTPRINTCOPY on this TERMINAL definition.

ATTACHSEC({LOCAL|IDENTIFY|VERIFY| PERSISTENT|MIXIDPE}) (APPC only)
specifies the level of attach time user security required for the connection. PERSISTENT and MIXIDPE are valid only with VTAM as the access method and when APPC is being used.

LOCAL
The authority of the user is taken to be that of the link itself, and you rely on link security alone to protect your resource.

IDENTIFY
Incoming attach requests must specify a user identifier. Specify IDENTIFY when the connecting terminal has a security manager.

MIXIDPE
A connection is able to support attaches using either or both of the IDENTIFY and PERSISTENT security types. The security type used depends on the incoming attach.

As in previous releases, IDENTIFY implies a degree of trust between the two systems that allows this system to accept the sign-on logic of the other system. In effect, this is a distributed security manager, with one system doing the sign-on and the other doing the security check.

PERSISTENT
This involves a user sign-on to a remote system that persists over multiple conversations until the user signs off from the remote system. In this way, the user’s ID and password are passed only on the first (sign-on) attach. Subsequent attach requests require only the user’s ID.

VERIFY
Incoming attach requests must specify a user identifier and a user password. Specify VERIFY when the connecting terminal has no security manager and therefore requires verification.

AUTINSTMODEL({NO|YES|ONLY})
specifies whether this TERMINAL definition can be used as a model terminal definition for autoinstall. For more information on autoinstall and model TERMINAL definitions, see “Chapter 8. Autoinstall for VTAM terminals” on page 121.

NO This definition is not used as a model for autoinstall. It is used only as a definition for a specific device that is not autoinstalled.

ONLY This definition is used only as a model for autoinstall. It is not used as a definition for a specific device.

YES This definition is used for a specific device that is not autoinstalled. The definition is also used as a model for automatic installation.

AUTINSTNAME(*name*)
specifies the name by which this model definition is known in the autoinstall control program. The name can be up to eight characters in length. The acceptable characters are A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

You need specify this only if AUTINSTMODEL is YES or ONLY. You can let it default to the TERMINAL name followed by four blanks, as long as this is acceptable to the autoinstall control program. For more information about autoinstall models, autoinstall names, and the autoinstall control program, see "Part 4. Autoinstall" on page 117.

BINDPASSWORD(*password*) (APPC only)
specifies a password of up to 16 hexadecimal characters (0-9, A-F). A password of fewer than 16 characters is padded on the right with hexadecimal zeros. A password of more than 16 characters is ignored.

CICS masks the password to avoid unauthorized access. You should therefore find a safe way of recording the password.

If you supply a password, an identical password must be supplied in the CONNECTION definition for the remote system to ensure bind time security.

BINDSECURITY({NO | YES}) (APPC only)
specifies whether an external security manager (ESM) is being used for bind-time security.

No No external bind-time security is required. However, if you define a password on the BINDPASSWORD attribute, CICS defaults to using its own internal security mechanism for bind security checking.

Yes If security is active and the XAPPC system initialization parameter is set to YES, CICS attempts to extract the session key from RACF in order to carry out bind-time security. If no RACF profile is available, the bind fails.

CONSOLE({NO | number})
For migration purposes, it is possible to define a console using CONSOLE(number). However, when several MVS images are united to form a sysplex, the assignment of console identification numbers depends on the order in which the MVS images are IPLed. The identification numbers are determined from the sequence in which they are encountered in the several CONSOL*nn* members for the MVS images. Therefore, you are recommended to identify console devices attached to the sysplex by CONSNAME instead of CONSOLE. The results of using CONSOLE may be unpredictable.

Code a number in the range 01 through 250, but not 128. However, before you can use the console, it must be either defined to MVS in the CONSOL*nn* member of SYS1.PARMLIB or dynamically allocated by a product such as NETVIEW.

CONSNAME(*name*)
Using CONSNAME, you can install a CICS console definition without having an existing console, and without the console being previously defined in the CONSOL*nn* member of the MVS SYS1.PARMLIB. However, before you can use the console, you must define the name to MVS, either in the CONSOL*nn* member of SYS1.PARMLIB or by dynamic allocation.

Notes:
1. Define either CONSOLE or CONSNAME; you cannot define both.
2. If the name DFHCONxx (where xx can be any valid characters) is specified, the terminal will be used by CICS as a 'pool' console for use when processing a modify command from a console with a name which is not defined to CICS.

The length of CONSNAME must be 2–8 characters and must begin with an alphabetic character or one of #, @, or $. It uniquely identifies the console
# device within a within a CICS region, regardless of the MVS image to which it
# is connected; that is, you cannot install two console definitions with the same
CONSNAME. The CONSNAME corresponds to the name defined for the console in the MVS SYS1.PARMLIB member, CONSOL*nn*.

To define a TSO user as a console device, specify CONSNAME(*name*), where *name* is the TSO userid. This enables a TSO user authorized to use the TSO CONSOLE command to initiate CICS transactions. The TSO userid does not have to be defined in the CONSOL*nn* member of SYS1.PARMLIB member.

| The equivalent of CONSOLE(00) is CONSNAME(INTERNAL) or
| CONSNAME(INSTREAM), depending on the service level of CICS and the
| release of MVS being used; specify this if you want to initiate a CICS
| transaction and issue a command to it in a JCL statement. For guidance about using JCL to issue CICS commands, see the *CICS Operations and Utilities Guide*.

DESCRIPTION(*text*)
You can provide in this field a description of the resource you are defining. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see “How the CSD file is organized—groups and lists” on page 8.

The GROUP name can be up to eight characters in length. The characters allowed are A-Z 0-9 $ @ and #. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

INSERVICE({YES|NO})
specifies the status of the terminal that is being defined.

YES Transactions may be initiated and messages may automatically be sent to the terminal.

NO The terminal can neither receive messages nor transmit input.

MODENAME(*name*) (APPC single session terminals only)
specifies the name that is passed to VTAM as the LOGMODE name. The name may be up to eight characters in length, but may not have the reserved name SNASVCMG. The name follows assembler language rules. It must start with an alphabetic character. The acceptable characters are A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

For further guidance on the LOGMODE name, see the *CICS Transaction Server for OS/390 Installation Guide*.

NATLANG({blank|E|K})
specifies the language in which all NLS-enabled messages are displayed for this terminal.

Use only one character, which can be A-Z 1-9

blank If you leave this blank and do not supply a value, CICS uses the system default as specified in the system initialization table (SIT).

E English

K Kanji

NETNAME(*name*)
specifies the network name that identifies the terminal to ACF/VTAM. The name may be up to 8 characters in length. The name follows assembler language rules. It must start with an alphabetic character. The acceptable characters are A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

If you do not specify a name, the NETNAME defaults to the TERMINAL name.

| The NETNAME must be unique except in the case of a remote terminal. That
| is, you **cannot** install two local terminals with the same NETNAME, or a local
| terminal and any connection with the same NETNAME. However, the
| NETNAME for a remote terminal **can** be the same as the NETNAME for any
| other terminal or the NETNAME for any connection.

OPERID(*code*)
The OPERID attribute is not valid in CICS Transaction Server for OS/390. See "Appendix A. Obsolete attributes retained for compatibility" on page 497 for information.

OPERPRIORITY({0|*number*})
The OPERID attribute is not valid in CICS Transaction Server for OS/390. See "Appendix A. Obsolete attributes retained for compatibility" on page 497 for information.

OPERRSL({0|*number*[,...]})
The OPERRSL attribute is not valid in CICS Transaction Server for OS/390. See "Appendix A. Obsolete attributes retained for compatibility" on page 497 for information.

OPERSECURITY({1|*number*[,...]})
The OPERSECURITY attribute is not valid in CICS Transaction Server for OS/390. See "Appendix A. Obsolete attributes retained for compatibility" on page 497 for information.

POOL(*name*)
specifies the pool name for a 3600 or 3650 pipeline terminal pooled with other pipeline terminals.

When you define a 3600 pipeline logical unit, you generate a TCTTE that is associated with a pool of TCTTEs. A pool of pipeline TCTTEs can be used by one pipeline logical unit, or it can be shared by a number of pipeline logical units.

The pool name is used only as a method of identifying the related TERMINAL definitions on the CSD file. It is not used within the active CICS system. The name can be up to 8 characters in length. The acceptable characters are A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

For a pipeline terminal, you must specify a TYPETERM with SESSIONTYPE(PIPELINE) specified. You must specify a TASKLIMIT on at least one of the pool of pipeline terminals. You must name the same group for each of the pipeline terminals in a pool.

An example of the definitions for a pool of pipeline terminals follows.

```
DEFINE TERMINAL(ttt1) GROUP(g) POOL(poolid)
       TYPETERM(xxxxxxxx) NETNAME(nnnnnnn1)...
DEFINE TERMINAL(ttt2) GROUP(g) POOL(poolid)
       TYPETERM(xxxxxxxx) NETNAME(nnnnnnn2)...
DEFINE TERMINAL(ttt3) GROUP(g) POOL(poolid)
       TYPETERM(xxxxxxxx) NETNAME(nnnnnnn3)...
DEFINE TERMINAL(ttt4) GROUP(g) POOL(poolid)
       TASKLIMIT(nn) TYPETERM(xxxxxxxx)
       NETNAME(nnnnnnn4)...
DEFINE TYPETERM(xxxxxxxx) GROUP(g)
       DEVICE(3650) SESSIONTYPE(PIPELINE)
```

PRINTER(*name*)
specifies the name of the primary 3270 printer to be used to respond to an ISSUE PRINT command, or a PRINT request from an operator pressing a program access (PA) key. The name may be up to four characters in length. The name is the TERMINAL name on the definition for the printer. If you name a PRINTER here, the TYPETERM referenced by this TERMINAL definition must have PRINTADAPTER(NO).

The printer you name must be owned by the same CICS system that owns this TERMINAL definition.

You can name a PRINTER if this TERMINAL definition is for one of the following:
- A 3270 display without the printer-adapter feature
- A 3270 display attached to a 3274 control unit
- A 3276 control unit display station
- A 3790 in 3270 compatibility mode

If you want to specify the hardware COPY feature, specify PRINTERCOPY(YES) on this TERMINAL definition.

Note that SNA character string (SCS) printers accept only 3790 data streams; they do not accept 3270 data streams. They therefore cannot be used to print the contents of a display unit’s buffer.

If you use a program attention key (for example, PA1) to print the contents of the screen on an associated VTAM printer, the screen size of the printer is chosen according to the SCRNSIZE operand as defined in the CICS-supplied default profile DFHCICST. This profile is defined with SCRNSIZE(DEFAULT) and, if you want to use the alternate screen size of the printer, you have to change the profile entry for transaction CSPP.

PRINTERCOPY({NO|YES})
specifies whether the hardware COPY feature is to be used to satisfy a print request on the PRINTER named in this TERMINAL definition.

CICS uses the hardware COPY feature of the 3270 to perform the print, unless a task is currently attached to the display.

You need not specify COPY(YES) on the TYPETERM definition, because this is implied by PRINTERCOPY(YES) on the TERMINAL definition.

If you have named an ALTPRINTER as well as a PRINTER, you may specify ALTPRINTCOPY(YES).

To use the COPY feature, both the printer and the display terminal must be on the same 3270 control unit. Otherwise, either the COPY may fail, raising an error condition or, if the display device address is valid for the printer's control unit, copying might be performed from a different display.

Do not specify PRINTERCOPY(YES) if, in a networking environment, the 3270 control unit is connected to a TCAM system in one domain, and a CICS system in another domain has access to the control unit via VTAM. This is because the hardware COPY address is not available to CICS and cannot therefore be used by terminals attached to such a control unit.

The COPY command is invalid for a 3270 compatibility mode display.

REMOTENAME(*name*)
specifies the name by which the terminal is known in the system or region that owns the terminal. The name can be up to four characters in length. The acceptable characters are A-Z a-z 0-9 $ @ # . / - _ % & ¢ ? ! : | " = ¬ , ; < >.

REMOTESYSNET(*name*)
specifies the network name (APPLID) of the region that owns the terminal. The name can be up to eight characters in length. It follows assembler language rules, and must start with an alphabetic character. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

REMOTESYSNET is used where there is no direct link between the region in which this definition is installed and the terminal-owning region. You do not need to specify REMOTESYSNET if either of the following is true:

- You are defining a local terminal (that is, REMOTESYSTEM is not specified, or specifies the sysid of the local system).
- REMOTESYSTEM names a direct link to the terminal-owning region. However, if the terminal-owning region is a member of a VTAM generic resources group and the direct link is an APPC connection, you may need to specify REMOTESYSNET. REMOTESYSNET is needed in this case if the NETNAME specified on the CONNECTION definition for the direct link is the generic resource name of the TOR (not the applid).

REMOTESYSTEM(*name*)
specifies the name that identifies the intercommunication link to the system that owns the terminal. The name can be up to 4 characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

This is the CONNECTION name on the CONNECTION definition for the intercommunication link. If it is not specified, or if it is specified as the sysid of the local system, this terminal is local to this system. If the name is that of another system, the terminal is remote. You can therefore use the same definition for the terminal in both the local system and a remote system.

If there are intermediate systems between this CICS and the terminal-owning region, REMOTESYSTEM should specify the first link in the path to the TOR. If there is more than one possible path, it should specify the first link in the preferred path.

REMOTESYSTEM is ignored if you specify AUTINSTMODEL(YES) or (ONLY).

SECURITYNAME(*name*)
specifies the security name of the remote system.

In a CICS system with security initialized (SEC=YES or MIGRATE), the security name is used to establish the authority of the remote system.

Note: If USERID is specified in the session definition (DEFINE SESSIONS command) associated with the connection definition, it overrides the userid specified in the SECURITYNAME attribute, and is used for link security.

The security name (or USERID on the sessions definition) must be a valid RACF userid on your system. Access to protected resources on your system is based on the RACF user profile and its group membership.

For further information on defining connections for MRO, LUTYPE6.1, and APPC, see the *CICS RACF Security Guide*.

TASKLIMIT(NO | *number*)
specifies the number of concurrent tasks allowed to run in a pipeline session or in a pool of pipeline sessions. The number can be in the range 1 through 32767.

When you define a 3600 pipeline logical unit, you generate a TCTTE that is associated with a pool of TCTTEs. As messages enter CICS from the 3600 pipeline logical unit, a task is attached to process this message, using as an anchor block one of the TCTTEs from the pool. In this way, consecutive messages sent via the pipeline logical unit can be processed concurrently, the number of concurrent transactions being limited by the number of TCTTEs in the pool. The number of TCTTEs in the pool should represent the high-water mark of inquiry activity. In this way, the pipeline facility allows fewer TCTTEs to be defined to CICS than the total number of pipeline inquiry terminals.

TERMINAL(*name*)
specifies the terminal identifier. For a terminal, the name can be up to four characters in length. The acceptable characters are A-Z a-z 0-9 $ @ # . / _ % & ¢ ? ! : | " = ¬ , ; < >. For an APPC LU6.2 single session terminal, the valid characters are A-Z 0-9 $ @ #. If the name supplied is fewer than four characters, it is left-justified and padded with blanks up to four characters. You should not have a TERMINAL definition and a CONNECTION definition with the same name.

Note: If you use a comma (,) in a name, you will be unable to use those commands such as

```
CEMT INQUIRE TERMINAL(value1,value2)
CEMT SET     TERMINAL(value1,value2)
```

where the comma serves as a list delimiter. See the *CICS Supplied Transactions* manual for information about using lists of resource identifiers.

Also, if you are allowing CICS to generate automatically session names, or terminal IDs for consoles, avoid using any of the following symbols as the first character in the terminal name:

-
: The hyphen is used by CICS for automatically generated terminal names for APPC sessions

> *and* **<**
: The > (greater than) and < (less than) symbols are used by CICS for automatically generated terminal names for IRC sessions

¬
: The ¬ (logical not) symbol is used by CICS for automatically generated terminal names for MVS consoles.

The name specified becomes the name of the TCT entry, when this TERMINAL definition is installed. For this reason, the TERMINAL name should be unique (note that the value CERR is reserved for the identification generated for the error console). If you specify AUTINSTMODEL(ONLY), you need not worry about making the TERMINAL name unique, because it is **not** used as the name of a TCT entry. If you specify AUTINSTMODEL(YES), the TERMINAL name is used as the name of the TCT entry that is installed in the TCT when the TERMINAL definition is installed; the names of the TCT entries for the autoinstalled terminals are determined by the autoinstall user program.

If the terminal is to be associated with a transient data destination, the terminal name and the destination identification in the DCT must be the same.

TERMPRIORITY({0|*number*})
specifies the terminal priority. This decimal value (0 through 255) is used in establishing the overall transaction processing priority. (Transaction processing priority is equal to the sum of the terminal priority, transaction priority, and operator priority, not exceeding 255.)

TRANSACTION(*name*)
specifies a 1-to 4-character transaction name. The acceptable characters are A-Z a-z 0-9 $ @ # . / - _ % & ¢ ? ! : | " = ¬ , ; < >. For VTAM non-3270 devices, a TRANSACTION name of fewer than four characters requires a delimiter.

It specifies the name of the transaction that is to be initiated each time input is received from the terminal when there is no active task. For information on what happens when a transaction is initiated, see the *CICS Application Programming Guide*.

If you specify this operand for a 3270 display, the only CICS functions the operator can invoke, other than this transaction, are paging commands and print requests.

You are unlikely to specify the TRANSACTION attribute for a 3270 display or SCS printer. It is optional for 3601 logical units, but is mandatory for 3614 logical units.

If this operand is specified for a 3790 Communication System, and multiple sessions are used to connect the same 3791, specify the same TRANSACTION name for all sessions.

TYPETERM(*name*)
specifies the name of the TYPETERM definition to be associated with this TERMINAL definition. The name can be up to eight characters in length. The

acceptable characters are A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

The TYPETERM definition specifies many attributes for a number of similar terminals. For more information, see "TYPETERM" on page 333 and "Installing resource definitions" on page 12. This attribute is mandatory for all TERMINAL definitions.

The TYPETERM definition should already be installed when you install this TERMINAL definition.

All TERMINAL definitions used to define pooled consoles (those with a Consname of DFHCONxx) should specify the same TYPETERM.

USEDFLTUSER ({NO | YES}) (APPC only)
specifies the kind of security checking that will take place for each inbound attach FMH.

NO Indicates that each inbound attach FMH will be checked for the presence of those fields required by the ATTACHSEC option and if the required fields are not present a protocol violation message will be issued and the attach will fail. NO is the default.

YES Indicates that some checks on the validity of the attach FMH are bypassed. This provides the same level of security as in releases of CICS prior to CICS/ESA 4.1. See *'Attach Time Security and the USEDFLTUSER option'* in the *CICS RACF Security Guide*.

USERID(name | *EVERY | *FIRST)
specifies a user identifier used for sign-on and referred to in security error messages, security violation messages, and the audit trail. It must be a valid userid defined to the security manager.

This is the only way to specify a user identifier for devices such as printers that are unable to sign on using CESN. You can also specify USERID for a display device; if so, the display is permanently signed on. Operators are unable to sign on. All access to protected resources depends on the authorizations permitted by RACF for the specified USERID.

For an APPC single session terminal, this USERID overrides any SECURITYNAME that you have specified for the connection.

name This can be up to eight characters in length. The acceptable characters are A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

***EVERY**
This is a special operand for autoinstalled consoles only. It means that CICS is to use the userid passed on the MVS MODIFY command every time a MODIFY command is received. The console is signed on using the MVS userid as the preset userid for the console being autoinstalled. The console remains signed on with this userid until the console is deleted or another MODIFY is received with another userid. If a MODIFY command is received without a userid, CICS signs on the default CICS userid until a MODIFY command is received that has a valid userid. For non-console terminals, or if security is not enabled, this value is ignored.

***FIRST**

This is a special operand for autoinstalled consoles only. It means that CICS is to use the userid passed on the first MVS MODIFY command that requires the console to be autoinstalled. The console is signed on with the MVS userid as the preset userid. The console remains signed on with this userid until the console is deleted. If a MODIFY command is received without a userid, CICS signs on the default CICS userid. For non-console terminals, or if security is not enabled, this value is ignored.

Notes:

1. If this terminal definition defines a console, the userid must be authorized to the appropriate profile in the CONSOLE general resource class, (see the *CICS RACF Security Guide* for information about preset security on consoles and terminals).
2. Preset Security, if defined for pooled consoles (those with a Consname of DFHCONxx) will allow each pooled console to be assigned only once as each will then be permanently signed-on. This means that, once used, each pool entry will not be returned to the pool but will be permanently assigned to the console name which first used it. Preset security, if used for pooled consoles, should be defined with the same level of security for all consoles in the pool as any pooled console may be selected for use when an unknown console requires a transaction to be run.

TRANCLASS

The TRANCLASS definition allows you to define a transaction class. Transactions that are defined as belonging to a transaction class are subject to scheduling constraints before they are allowed to execute. If transactions belonging to an active transaction class are already running, any new transactions are queued. Use the MAXACTIVE attribute to specify the maximum number of transactions that you want to run. To limit the size of the queue, you can use the PURGETHRESH attribute.

By putting your transactions into transaction classes, you can control how CICS dispatches tasks. For example, you can separate transactions into those that are heavy resource users and those that are of lesser importance, such as the “Good morning” broadcast messages. You can then use the attributes on the TRANCLASS definition to control the number of active tasks allowed from each transaction class.

The DEFINE panel for TRANCLASS is shown in Figure 44.

Defining a TRANCLASS

```
CEDA  DEFine TRANClass( XXXX     ) GROUP(MYGROUP)
 TRANClass      : XXXX
 Group          : MYGROUP
 Description  ==>
CLASS LIMITS
 Maxactive    ==>                0-999
 Purgethresh  ==> No                No | 1-1000000
```

Figure 44. CEDA DEFINE TRANCLASS panel

Options

Description(*text*)
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Each resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see “How the CSD file is organized—groups and lists” on page 8.

The GROUP name can be up to eight characters in length. The characters allowed are A-Z 0-9 $ @ and #. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

MAXACTIVE(*value*)
specifies the maximum number of transactions in this transaction class that are allowed to be active. You must specify a MAXACTIVE value when you define a transaction class, in the range 0 through 999.

New transactions attached when the number of active transactions has reached the MAXACTIVE limit are considered for queueing subject to the PURGETHRESH limit.

Defining a transaction class with a zero MAXACTIVE value signifies that **all** tasks are to be queued.

PURGETHRESH({NO|*number*})
This is an optional purge threshold for the transaction class; it defines a threshold number at which transactions queuing for membership of the transaction class are purged. Specify it if you want to limit the number of transactions queueing in this transaction class. It can have the following values:

NO The size of the queue is unlimited (other than by the storage available to attach tasks).

number
The purge threshold number in the range 1—1 000 000.

If you specify this as 1, no transactions are allowed to queue. If you specify it as any other number (*n*), the size of the queue is restricted to *number–1*. All new transactions attached after the limit of *n–1* is reached are purged.

Example of PURGETHRESH: In the case of a transaction class where the maximum number of active tasks (MAXACTIVE) is set to 50, and the purge threshold (PURGETHRESH) is set to 10 to limit queuing transactions, CICS begins to abend new transactions for the class when:

- The number of active transactions reaches 50, and
- The number of transactions queuing for membership of the transaction class has reached 9

CICS accepts new transactions for this transaction class queue only when the number queued falls below the maximum size of the queue (9 in our example).

TRANCLASS(*name*)
specifies the name of the transaction class. Transactions belonging to a transaction class are subject to scheduling constraints before they are allowed to execute. The reserved TRANCLASS name DFHTCL00 is used to indicate that the transaction does not belong to any transaction class.

For compatibility with releases that support a TCLASS attribute, CICS provides the following TRANCLASS equivalents:

TCLASS	TRANCLASS
NO	DFHTCL00
1	DFHTCL01
2	DFHTCL02
3	DFHTCL03
4	DFHTCL04
5	DFHTCL05
6	DFHTCL06

7	DFHTCL07
8	DFHTCL08
9	DFHTCL09
10	DFHTCL10

Sample definitions for these transaction classes are in group DFHTCL, supplied as part of DFHLIST.

Note: If a transaction is run and its associated TRANCLASS definition is not installed, the transaction runs without any of the scheduling constraints specified in the TRANCLASS. Attention message DFHXM0212 is issued.

TRANCLASS can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and $. Lowercase characters are treated as uppercase characters.

TRANSACTION

A CICS application consists of one or more programs written to perform a specific function. A particular invocation of such an application is known as a **transaction**, and the CICS transaction manager identifies it by its transaction identifier (TRANSID). You tell CICS how you want your transaction to run, primarily in a TRANSACTION definition. The name of this definition, the TRANSACTION name, is the same as the TRANSID.

In the TRANSACTION definition, you specify attributes related to functions provided by CICS itself. The transaction priority, security key, and the length of the transaction work area (TWA) are examples of these options.

You also link the transaction with other resources by coding the names of their definitions in the TRANSACTION definition. These other resources are PROGRAM, PROFILE, PARTITIONSET, REMOTESYSTEM, and TRANCLASS.

PROGRAM
You specify options related to the software implementation of your application in the PROGRAM definition. This defines the program to which control is to be given to process the transaction. The TRANSACTION definition references the PROGRAM definition.

PROFILE
You do not have to specify, for each transaction, the attributes that control the interaction with a terminal or logical unit. Instead, the TRANSACTION definition references a PROFILE definition, which specifies them for a number of transactions.

REMOTESYSTEM
For transaction routing, instead of specifying a PROGRAM name in the TRANSACTION definition, you specify the name of a REMOTESYSTEM. This can be the name of another CICS system, which itself is defined to this CICS system in a CONNECTION definition of the same name.

If you name a REMOTESYSTEM, you may also supply a REMOTENAME, which is the name of the transaction to be run in the remote system. The remote system decides which program it gives control to.

If you specify a REMOTESYSTEM name that corresponds to the system in which the definition is installed, CICS installs a local transaction resource definition. Otherwise CICS installs a remote transaction resource definition.

TRANCLASS
This specifies the name of the transaction class to which the transaction belongs. Transactions belonging to a transaction class are subject to scheduling constraints before they are allowed to execute. The constraints are specified in the associated TRANCLASS definition. The TRANCLASS attribute is described on page 325.

The TRANSACTION definition is therefore the central resource definition for an application. The related resource definitions are discussed below.

Defining a TRANSACTION

Figure 45 on page 315 shows the CEDA DEFINE TRANSACTION screen.

```
 TRANSaction  ==>
 Group        ==>
 DEscription  ==>
 PROGram      ==>
 TWasize      ==> 00000            0-32767
 PROFile      ==> DFHCICST
 PArtitionset ==>
 STAtus       ==> Enabled          Enabled | Disabled
 PRIMedsize     : 00000            0-65520
 TASKDATALoc  ==> Below            Below | Any
 TASKDATAKey  ==> User             User | Cics
 STOrageCLear ==> No               No | Yes
 RUnaway      ==> System           System | 0 | 500-2700000
 SHutdown     ==> Disabled         Disabled | Enabled
 ISolate      ==> Yes              Yes | No
 BRexit       ==>
REMOTE ATTRIBUTES
 DYnamic      ==> No               No | Yes
 ROutable     ==> No               No | Yes
 REMOTESystem ==>
 REMOTEName   ==>
 TRProf       ==>
 Localq       ==> No               No | Yes
SCHEDULING
 PRIOrity     ==> 001              0-255
 TClass         : No               No | 1-10
 TRANCLASS    ==> DFHTCL00
ALIASES
 Alias        ==>
 TASKReq      ==>
 XTRanid      ==>
 TPName       ==>
              ==>
 XTPname      ==>
              ==>
              ==>

RECOVERY
 DTimout      ==> No               No | 1-6800
 RESTart      ==> No               No | Yes
 SPurge       ==> No               No | Yes
 TPUrge       ==> No               No | Yes
 DUmp         ==> Yes              Yes | No
 TRACe        ==> Yes              Yes | No
 COnfdata     ==> No               No | Yes
INDOUBT ATTRIBUTES
 ACtion       ==> Backout          Backout | Commit
 WAIT         ==> Yes              Yes | No
 WAITTime     ==> 00 , 00 , 00     0-99 (Days, Hours, Mins)
 INdoubt        : Backout          Backout | Commit | Wait

SECURITY
 RESSec       ==> No               No | Yes
 Cmdsec       ==> No               No | Yes
 Extsec         : No
 TRANSec        : 01               1-64
 RSl            : 00               0-24 | Public
```

Figure 45. The DEFINE panel for TRANSACTION

Options

ACTION({BACKOUT|COMMIT})

specifies the action to be taken when a CICS region fails, or loses connectivity with its coordinator, during two-phase commit processing after the unit of work has entered the in-doubt period. The action depends on the WAIT attribute. If WAIT specifies YES, ACTION has no effect unless the WAITTIME expires before recovery from the failure.

If WAIT specifies NO, the action taken is one of the following:

BACKOUT
: All changes made to recoverable resources are backed out, and the resources are returned to the state they were in before the start of the UOW.

COMMIT
: All changes made to recoverable resources are committed, and the UOW is marked as completed.

ALIAS(*name*)
allows you to specify an alias transaction name for this transaction. The name may be up to four characters in length. This useful if you wish to run on a terminal defined with UCTRAN(NO), or a transaction that allows mixed case input (PROFILE UCTRAN(NO)). For example, you can invoke via alias(**abcd**) the same transaction as **ABCD**.

BREXIT(*name*)
This is an optional parameter that defines the name of the default bridge exit to be associated with this transaction, if it is started in the 3270 bridge environment with a START BREXIT command, and BREXIT specifies no name. The name may be up to eight characters in length.

If BREXIT is defined, REMOTESYSTEM, REMOTENAME, DYNAMIC(YES), and RESTART(YES) should not be specified, and will be ignored.

Note: The meaning of BREXIT has changed since CICS Transaction Server for OS/390 Release 2. This has particular significance for those considering sharing the CSD between CICS Transaction Server for OS/390 Release 2 and this release. This issue is discussed in the *CICS Transaction Server for OS/390 Migration Guide* for CICS Transaction Server for OS/390 Release 3

CMDSEC({NO|YES})
specifies whether security checking is to be applied on system programming commands. For programming information on the system programming commands, see the *CICS System Programming Reference* manual.

NO No check is made. The commands are always executed.

YES A call is made to the external security manager (ESM). CICS either authorizes or prevents access. If the ESM cannot identify the resource or resource type, access is prevented.

DESCRIPTION(*text*)
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

DTIMOUT({NO|*value*})
specifies whether deadlock time-out is be applied to the task. If the execution of the task gets suspended (for example, through lack of storage), a purge of the task is initiated if the task stays suspended for longer than the DTIMOUT value. If the purge leads to a transaction abend, the abend code used depends on which part of CICS suspended the task. When using CEDF, the user task should, if possible, specify DTIMOUT(NO), or a large value. This is also now used as the timeout on all RLS file requests if DTIMOUT is non-zero,

otherwise the request gets the SIT FTIMEOUT value. FTIMEOUT applies to transactions that do not have a deadlock timeout interval active. If the DTIMOUT keyword of the TRANSACTION definition is specified, it is used as the file timeout value for that transaction.

Note: When using CEDF, if any DTIMOUT value has been specified for the user task, the DTIMOUT value is ignored while the user task is suspended and a CEDF task is active. Therefore the suspended user task cannot terminate with a deadlock timeout (abend AKCS) while a CEDF task is waiting for a user response.

For DTIMOUT to be effective in non-RLS usage, SPURGE must be set to YES.

CICS inhibits deadlock time-out at certain points.

DTIMOUT is not triggered for terminal I/O waits. Because the relay transaction does not access resources after obtaining a session, it has little need for DTIMOUT except to trap suspended allocate requests. However, for I/O waits on a session, the RTIMOUT attribute can be specified on PROFILE definitions for transaction routing on mapped APPC connections.

It is important that you define some transactions with a DTIMOUT value, because deadlock time-out is the mechanism that CICS uses to deal with short-on-storage (SOS) situations.

NO The deadlock time-out feature is not required.

value The length of time (MMSS for minutes and seconds) after which the deadlock time-out facility terminates a suspended task. The maximum value that you can specify is 68 minutes; this is accurate to one second.

CONFDATA({NO|YES})
specifies whether CICS is to suppress user data from CICS trace entries when the CONFDATA system initialization parameter specifies HIDETC. If the system initialization parameter specifies CONFDATA=SHOW, CONFDATA on the transaction definition is ignored.

If the system initialization parameter specifies CONFDATA=HIDETC, the following options are effective:

NO CICS does not suppress any user data. VTAM and MRO initial user data is traced in trace point AP FC92. FEPI user data is traced in the normal CICS FEPI trace points.

YES CICS suppresses user data from the CICS trace points.

DUMP({YES|NO})
specifies whether a call is to be made to the dump domain to produce a transaction dump if the transaction terminates abnormally.

YES CICS calls the dump domain to produce a transaction dump. Note that the final production or suppression of the transaction dump is controlled by the transaction dump table. For more information about the dump table, see the *CICS Problem Determination Guide*.

If no transaction dump table entry exists for the given dump code when a transaction abends, CICS creates a temporary entry for which the default is to produce a transaction dump.

You control dump table entries for transaction dumps using the CEMT transaction (for more information, see the *CICS Supplied Transactions* manual) or the EXEC CICS SET TRANDUMPCODE command (for programming information, see the *CICS System Programming Reference* manual).

NO No call is made to the dump domain, suppressing any potential transaction dump.

Note: This operand has no effect on the following:

- An EXEC CICS DUMP command, which always produces a dump.
- The system dumps for dump codes AP0001 and SR0001 that CICS produces in connection with ASRA, ARSB, or ASRD abends. If you specify NO on the transaction DUMP attribute, CICS suppresses the transaction dump, but not the system dump.

DYNAMIC({NO|YES})
specifies whether the transaction can be dynamically routed to a remote region, using the CICS dynamic transaction routing facility.

NO Creates a local or remote definition according to the REMOTESYSTEM attribute.

YES Allows the dynamic transaction routing program to determine the local or remote status dynamically at invocation time. For programming information about the dynamic transaction routing program, see the *CICS Customization Guide*.

EXTSEC({NO|YES})
The EXTSEC attribute is not valid in CICS Transaction Server for OS/390. See "Appendix A. Obsolete attributes retained for compatibility" on page 497 for more information.

INDOUBT({BACKOUT|COMMIT|WAIT})
This attribute is obsolete in CICS Transaction Server for OS/390, but is retained for compatibility with earlier releases. If you already use INDOUBT, you can still access it by using compatibility mode (see "Compatibility mode (CSD file sharing)" on page 12 for information). See "Appendix A. Obsolete attributes retained for compatibility" on page 497 for a description of INDOUBT.

ISOLATE({YES|NO})
specifies whether CICS is to isolate the transaction's user-key task-lifetime storage to provide transaction-to-transaction protection. (See the TASKDATAKEY attribute for a description of user-key storage.) Isolation means that the user-key task-lifetime storage is protected from both reading and writing by the user-key programs of other transactions—that is, from programs defined with EXECKEY(USER).

Note: The ISOLATE attribute does not provide any protection against application programs that execute in CICS key—that is, from programs defined with EXECKEY(CICS).

YES The transaction's user-key task-lifetime storage is isolated from the user-key programs of all other transactions—that is, from programs defined with EXECKEY(USER), but not from programs defined with EXECKEY(CICS).

Also, the user-key task-lifetime storage of **all** other transactions is protected **from** the user-key programs of transactions defined with ISOLATE(YES).

NO If you specify ISOLATE(NO), the transaction's task-lifetime storage is isolated from the user-key programs of those transactions defined with ISOLATE(YES). The transaction's storage is not, however, isolated from user-key programs of other transactions that also specify ISOLATE(NO) because, with this option, the transactions are all allocated to the common subspace.

Note also that the user-key task-lifetime storage of all transactions defined with ISOLATE(YES) is protected **from** the user-key programs of transactions defined with ISOLATE(NO).

Specify ISOLATE(NO) for those transactions that share any part of their user-key task-lifetime storage.

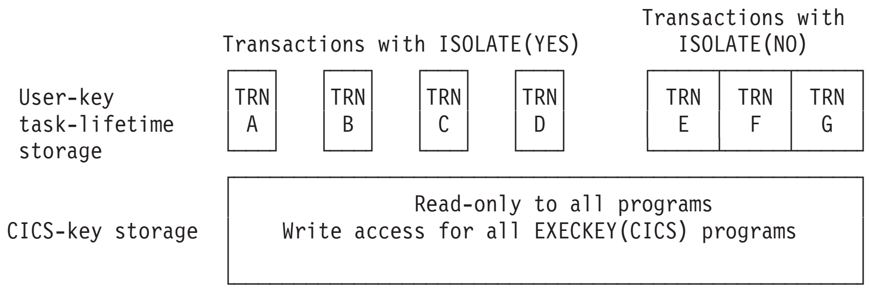

Transactions with ISOLATE(YES)
Transactions with ISOLATE(NO)
User-key task-lifetime storage
TRN A
TRN B
TRN C
TRN D
TRN E
TRN F
TRN G
CICS-key storage
Read-only to all programs
Write access for all EXECKEY(CICS) programs

Figure 46. Conceptual view of transactions with and without transaction isolation. In this diagram, the elements of user-key task-lifetime storage of each of the transactions A, B, C, and D are isolated from each other, and from transactions E, F, and G. For the transactions A, B, C and D, therefore, their storage is addressable by their own user-key programs only.

The user-key task-lifetime storage of transactions E, F, and G, however, is accessible by all the user-key application programs of transactions E, F and G, but is isolated from the user-key programs of transactions A, B, C, and D. All of the illustrated transactions have read-only access to CICS-key storage.

LOCALQ({NO|YES})
specifies whether queuing on the local system is to be performed.

NO No local queuing is to be performed.

YES Local queuing can be attempted for an EXEC START NOCHECK request when the system is not available and the system name is valid. A system is defined as not available when:

- The system is OUT OF SERVICE when the request is initiated.
- The attempt to initiate any session to the remote system fails and the corrective action taken by the abnormal condition program (DFHZNAC) or the node error program (DFHZNEP) is to place the system OUT OF SERVICE.
- No sessions to the remote system are immediately available, and your XISCONA global user exit program specifies that the request is not to be queued in the issuing region.

Local queuing should be used only for those EXEC START commands that represent time independent requests. The delay implied by local queuing affects the time at which the request is actually started. It is your responsibility to ensure that this condition is met.

| If you specify LOCALQ(YES), you must not also specify
| ROUTABLE(YES).

You can use the global user exit XISLCLQ for the intersystem communication program to override the setting of the LOCALQ attribute. For programming information on the user exits in the intersystem communication program, see the *CICS Customization Guide*.

PARTITIONSET(*name***|KEEP|OWN)**
specifies the name of the partition set that is to be the default application partition set. The name can be up to eight characters in length. The acceptable characters are A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

If you are not specifying a new name, delete a previously specified name by pressing the ERASE EOF key.

If you do not specify a partition set name or either of the reserved names, CICS destroys existing partitions before the first BMS output to the terminal from the transaction.

name CICS destroys existing partitions and loads the named partition set before the first BMS output to the terminal from the transaction. (Existing partitions are not destroyed if the terminal partition set matches the application partition set.)

This name must not be the same as that specified in PROGRAM(*name*).

KEEP The transaction uses the application partition set for this terminal, whatever it may be. This option is normally used for successor transactions in a chain of pseudo-conversational transactions.

OWN The transaction performs its own partition management.

PRIMEDSIZE({0|*value***})**
The PRIMEDSIZE attribute is obsolete in CICS Transaction Server for OS/390, but is supported to provide CSD file compatibility for earlier releases of CICS where it is still valid. See “Appendix A. Obsolete attributes retained for compatibility” on page 497 for more information.

PRIORITY({1|*value***})**
specifies the transaction priority. This 1-to 3-digit decimal value from 0 to 255 is used in establishing the overall transaction processing priority. (Transaction processing priority is equal to the sum of the terminal priority, transaction priority, and operator priority, not exceeding 255.) The higher the number, the higher the priority.

PROFILE({DFHCICST|*name***})**
is the name of the PROFILE definition that specifies the processing options used in conjunction with the terminal that initiated the transaction. The name can be up to eight characters in length. The acceptable characters are A-Z a-z 0-9 $ @ # . / - _ % & ¢ ? ! : | " = ¬ , ; < >.

The processing options provided by the default DFHCICST are shown in “PROFILE definitions in group DFHISC” on page 526. DFHCICST is not suitable for use with a distributed program link. Instead, specify DFHCICSA, which has INBFMH=ALL.

PROGRAM(*name***)**
specifies the name of the program to which CICS gives control to process this transaction. The name can be up to eight characters in length. The acceptable characters are A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

Ensure that this name is not the same as that specified in PARTITIONSET(name).

Note: If a name is specified for REMOTESYSTEM, and it differs from that of the current system, no name need be specified for PROGRAM. If, in these circumstances, a name is specified for PROGRAM, it may be ignored.

If this transaction definition is for use on a remote program link request, the program name you specify in this attribute must be the name of the CICS mirror program, DFHMIRS. See the TRANSID attribute on the PROGRAM definition on page 249.

REMOTENAME({*local-name* | *remote-name*})
specifies the name of this transaction as it is known in a remote system, if it is to be executed in a remote system or region using intersystem communication. The remote system can be another CICS region or an IMS system. REMOTENAME can be 1 through 4 characters in length if the REMOTESYSTEM attribute specifies another CICS region, or 1 through 8 characters in length if REMOTESYSTEM specifies an IMS system. IMS uses 8-character names and, if REMOTENAME has fewer than 8 characters, IMS translates it into a usable format. The acceptable characters are A-Z a-z 0-9 $ @ # . / - _ % & ¢ ? ! : | " = ¬ , ; < >.

If you specify REMOTESYSTEM and omit REMOTENAME, the value of REMOTENAME defaults to the local name; that is, the TRANSACTION name on this definition. Note that the transaction need not necessarily reside on the remote system or region.

REMOTESYSTEM(*name*)
specifies the name of the CONNECTION definition of the intercommunication link on which the transaction attach request is sent. The name can be up to four characters in length. The acceptable characters are A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

This attribute is used for CICS function request shipping (asynchronous processing and transaction routing). For more details of these methods of intercommunication, see the *CICS Intercommunication Guide*.

RESSEC({NO | YES})
specifies whether resource security checking is to be used for resources accessed by this transaction.

NO All resources are available to any user who has the authority to use this transaction.

YES An external security manager is used. For more details about external security checking, see the *CICS RACF Security Guide*.

RESTART({NO | YES})
specifies whether the transaction restart facility is to be used to restart those tasks that terminate abnormally and are subsequently backed out by the dynamic transaction backout facility.

If RESTART(YES) is specified, the task that failed is restarted from the beginning of the initial program. If dynamic transaction backout fails, or if restart is suppressed dynamically, DFHPEP is invoked in the normal way. The transaction restart facility is especially useful in such situations as a program isolation deadlock, where the task can be restarted automatically rather than resubmitted manually. For more details of automatic transaction restart, see the *CICS Recovery and Restart Guide*.

NO The restart facility is not required.

YES The restart facility is to be used.

ROUTABLE({NO|YES})
specifies whether, if the transaction is the subject of an eligible EXEC CICS START command, it will be routed using the enhanced routing method.

NO If the transaction is the subject of a START command, it will be routed using the “traditional” method.

YES If the transaction is the subject of an eligible START command, it will be routed using the enhanced method.

If you specify ROUTABLE(YES), you must not also specify LOCALQ(YES).

For details of the enhanced and “traditional” methods of routing transactions invoked by EXEC CICS START commands, see the *CICS Intercommunication Guide*.

RSL({0|value|PUBLIC})
The RSL attribute is not valid in CICS Transaction Server for OS/390. See “Appendix A. Obsolete attributes retained for compatibility” on page 497 for information.

RUNAWAY({SYSTEM|0|500–2700000})
The amount of time, in milliseconds, for which any task running under this transaction definition can have control of the processor before it is assumed to be in a runaway condition (logical loop). When this interval expires, CICS can abnormally terminate the task.

SYSTEM
CICS is to use the ICVR system initialization parameter value as the runaway time limit for this transaction.

0 There is no limit and no runaway task detection is required for the transaction.

500–2700000
The runaway time limit in the range 500 through 2700000.

SHUTDOWN({DISABLED|ENABLED})
applies to transactions associated with a terminal, and indicates whether the transaction can be run during CICS shutdown. This supplements the XLT option on EXEC CICS PERFORM SHUTDOWN. For a transaction to be attached during shutdown, it must either be defined as SHUTDOWN(ENABLED) or be named in the XLT specified in the EXEC CICS SHUTDOWN command.

DISABLED
The transaction is disabled from running during CICS shutdown.

ENABLED
The transaction is enabled to run during CICS shutdown.

SPURGE({NO|YES})
specifies whether the transaction is initially “system purgeable” or not.

SPURGE=NO prevents a transaction being purged by the deadlock time-out (DTIMOUT) facility, an EXEC CICS ... PURGE command, TWAOCT (Cancel Task) being set in the node error program (NEP), or a CEMT SET ... PURGE command.

SPURGE=YES allows such purges to go ahead as far as the user is concerned. CICS may, however, prevent the purge if it is not safe to allow a purge at the point the transaction has reached.

Note that SPURGE=NO does not prevent a transaction being purged by the read time-out (RTIMOUT) facility, an EXEC CICS SET ... FORCEPURGE command, or a CEMT SET TRANSACTION(tranid) FORCEPURGE command. SPURGE determines only the initial value, which can be changed by the transaction while it is running.

NO The transaction is not initially system purgeable.

YES The transaction is initially system purgeable.

STATUS({ENABLED|DISABLED})
specifies the transaction status.

ENABLED
Allows the transaction to be executed normally.

DISABLED
Prevents the transaction being executed.

STORAGECLEAR({NO|YES})
specifies whether task-lifetime storage for this transaction is to be cleared on release. This can be used to prevent other tasks accidentally viewing any confidential or sensitive data that was being stored by this transaction in task lifetime storage.

TASKDATAKEY({USER|CICS})
specifies the storage key of the storage CICS allocates at task initialization for the duration of the task (task-lifetime storage), and which is accessible by the application. These storage areas are the EXEC interface block (EIB) and the transaction work area (TWA).

TASKDATAKEY also specifies the key of the storage that CICS obtains on behalf of all programs that run under the transaction. The program-related storage that CICS allocates in the specified key includes:

- The copies of working storage that CICS obtains for each execution of an application program.
- The storage CICS obtains for the program in response to implicit and explicit GETMAIN requests. For example, the program can request storage by a GETMAIN command, or as a result of the SET option on other CICS commands.

You must specify TASKDATAKEY(USER) if any of the programs in the transaction is defined with EXECKEY(USER). If you specify TASKDATAKEY(CICS) for a transaction, an attempt to run any program in user key under this transaction leads to a task abend, with abend code AEZD.

USER CICS obtains user-key storage for this transaction. Application programs executing in any key can both read and modify these storage areas.

Note: User-key programs of transactions defined with ISOLATE(YES) have access only to the user-key task-lifetime storage of their own tasks.

User-key programs of transactions defined with ISOLATE(NO) also have access to the user-key task-lifetime storage of other tasks defined with ISOLATE(NO).

See the description of the EXECKEY attribute on the PROGRAM definition for more information about task storage protection.

CICS CICS obtains CICS-key storage for this transaction. Application programs executing in CICS key can both read and modify these storage areas. Application programs executing in user key can only read these storage areas.

TASKDATALOC({BELOW | ANY})
specifies whether task life-time storage acquired by CICS for the duration of the transaction can be located above the 16MB line in virtual storage. These areas, which relate to specific CICS tasks, include the EXEC interface block (EIB) and the transaction work area (TWA).

You must specify TASKDATALOC(BELOW) if any of the programs that make up the transaction runs in 24-bit addressing mode (this also applies to task-related user exits running on behalf of the transaction).

For transactions that do not satisfy any of these conditions, you can specify ANY to obtain the associated virtual storage constraint relief.

CICS polices the use of TASKDATALOC(ANY). In particular:

- An attempt to invoke an AMODE 24 program running under a transaction defined with TASKDATALOC(ANY) results in an AEZC abend.
- An attempt to issue an EXEC CICS command or call a task related user exit while running AMODE(24) with TASKDATALOC(ANY) specified results in an AEZA abend.
- An AMODE 31 program running as a transaction with TASKDATALOC(ANY), which attempts to invoke a task-related user exit that is forced to run AMODE(24), results in an AEZB abend.
- If a task-related user exit that is forced to run in AMODE 24 is enabled for task start, CICS forces TASKDATALOC(BELOW) for all transactions for the remainder of the CICS run.

BELOW
Storage areas that CICS acquires for the transaction must be located below the 16MB line.

ANY Storage areas that CICS acquires for the transaction can be located above the 16MB line in virtual storage.

TASKREQ(*value*)
specifies whether a transaction is to be initiated by pressing a PF key, by using a light pen, or by using a card. Possible values are:

- **PA1**, **PA2**, or **PA3** for PA keys.
- **PF1** through **PF24** for PF keys.
- **OPID** for the operator identification card reader.
- **LPA** for a light-pen-detectable field on a 3270 device.

- **MSRE** for the 10/63 character magnetic slot reader.

Here are some notes on the use of PF and PA keys:

- If a PA or PF key is specified in the PRINT system initialization parameter, you cannot use the same PF key as the TASKREQ to initiate a transaction.
- PA or PF keys specified in the SKR*xxxx* system initialization parameter as page retrieval keys are interpreted as such during a page retrieval session. You can use the same keys to initiate transactions at other times. The keys should be defined with the following values:

  ```
  TASKREQ=KEY-ID
  PROGRAM=DFHTPR
  TWASIZE=1024
  TPURGE=NO
  SPURGE=NO
  ```
- If you define a transaction with PROGRAM(DFHTPR), and define a TASKREQ key, the key initiates the transaction and opens the page retrieval session at the same time.

Note: For transactions that had a TASKREQ but no TRANSID before migration to RDO, there may be an incompatibility in the use of EIBTRNID by application programs. (See the *CICS Application Programming Guide* for information.)

TCLASS({NO|*value*})
As a result of the introduction of TRANCLASS, the TCLASS attribute is obsolete in CICS Transaction Server for OS/390. If you already use TCLASS, you can still access it by using compatibility mode (see “Compatibility mode (CSD file sharing)” on page 12 for information). See “Appendix A. Obsolete attributes retained for compatibility” on page 497 for a description of TCLASS.

TPNAME(*name*)
specifies the name of the transaction that may be used by an APPC partner if the 4-character length limitation of the TRANSACTION attribute is too restrictive. This name can be up to 64 characters in length. The acceptable characters are A-Z a-z 0-9 $ @ # . / - _ % & ¢ ? ! : | " = ¬ , ; < and >. If this range of characters is not sufficient for a name that you wish to specify, you may use the XTPNAME attribute instead of TPNAME.

TPURGE({NO|YES})
specifies (for non-VTAM terminals only) whether the transaction can be purged because of a terminal error.

NO The task cannot be purged when a terminal error occurs. Manual intervention by the master terminal operator is required when this happens.

YES The task can be purged when a terminal error occurs.

TRACE({YES|NO})
specifies whether the activity of this transaction is to be traced.

YES Trace the activity for this transaction.

NO Do not trace the activity for this transaction.

Note: The CICS-provided transaction definitions (previously PCT entries) for CEDF and CSGM have TRACE=NO coded.

TRANCLASS(*name*)
specifies the name of the transaction class to which the transaction belongs.

Transactions belonging to a transaction class are subject to scheduling constraints before they are allowed to execute. The reserved TRANCLASS name DFHTCL00 is used to indicate that the transaction does not belong to any transaction class.

In previous releases of CICS, you could specify a transaction class by using the now obsolete TCLASS attribute. TCLASS specified the value of the associated class. Compatibility with these classes is preserved if you specify the following TRANCLASS equivalents:

TCLASS	TRANCLASS
NO	DFHTCL00
1	DFHTCL01
2	DFHTCL02
3	DFHTCL03
4	DFHTCL04
5	DFHTCL05
6	DFHTCL06
7	DFHTCL07
8	DFHTCL08
9	DFHTCL09
10	DFHTCL10

Note: If a transaction is run and its associated TRANCLASS definition is not installed, the transaction runs without any of the scheduling constraints specified in the TRANCLASS. Message DFHXM0212 is issued as a warning.

TRANCLASS can be up to 8 characters in length. The characters allowed are A-Z 0-9 $ @ and #. Lowercase characters are treated as uppercase characters.

TRANSACTION(*name*)
specifies the name of the transaction, or transaction identifier (TRANSID). The name can be up to four characters in length. The acceptable characters are A-Z a-z 0-9 $ @ # . / - _ % & ¢ ? ! : | " = ¬ , ; < >. Do not use transaction names beginning with C, because these are reserved for use by CICS. Also, the '%' and '' characters can cause problems to RACF if transaction security is active. Check that the transaction names are acceptable to the external security manager that you are using.

Note: If you use a comma (,) in a name, you will be unable to use those commands such as

```
CEMT INQUIRE TRANSACTION(value1,value2)
CEMT SET     TRANSACTION(value1,value2)
```

where the comma serves as a list delimiter. See the *CICS Supplied Transactions* manual for information about using lists of resource identifiers.

If you wish to use other special characters in a transaction identifier, use the XTRANID attribute to specify another name that can be used to initiate the transaction. You must also specify a TRANSACTION name, because this is the name by which the TRANSACTION definition is known on the CSD file.

When defining a transaction, you must also name either a PROGRAM or a REMOTESYSTEM.

TRANSEC

The TRANSEC attribute is not valid in CICS Transaction Server for OS/390. See “Appendix A. Obsolete attributes retained for compatibility” on page 497 for more information.

TRPROF=({DFHCICSS|*name*})

specifies the name of the PROFILE for the session that carries intersystem flows during ISC transaction routing. The name can be up to eight characters in length. The acceptable characters are A-Z a-z 0-9 $ @ # . / - _ % & ¢ ? ! : | " = ¬ , ; < >.

You can specify this only for remote transactions.

TWASIZE({0|value})

specifies the size (in bytes) of the transaction work area to be acquired for this transaction. Specify a 1-to 5-digit decimal value in the range 0 through 32767.

Notes:

1. Your storage may be corrupted if your TWASIZE is too small.
2. Do not change the TWASIZE of the CICS-supplied transactions.

WAIT({YES|NO})

specifies whether an in-doubt unit of work (UOW) is to wait, pending recovery from a failure that occurs after the UOW has entered the in-doubt state.

Note: Old-style transaction definitions using INDOUBT(WAIT) are accepted by CICS, and are interpreted as WAIT(YES) ACTION(BACKOUT).

YES The UOW is to wait, pending recovery from the failure, to resolve its in-doubt state and determine whether recoverable resources are to be backed out or committed. In other words, the UOW is to be **shunted**.

Recoverable resources can include:

- DBCTL databases
- DB2 databases
- Temporary storage queues
- Logically-recoverable intrapartition transient data queues that specify WAIT(YES) in the TDQUEUE definition
- | VSAM data sets
- | BDAM data sets.

The WAIT(YES) option takes effect **provided that none of the following applies**:

- The transaction has subordinate MRO sessions to back-level systems.
- The transaction has LU6.1 subordinate sessions. (Note that, in this context, LU6.1 IMS sessions are not subordinates.)
- The transaction has more than one session and its coordinator session is to a back-level system, or LU6.1.
- The task-related user exits attached to the transaction do not support the CICS in-doubt protocols.

If none of the above exceptions applies, but there are subordinate LU6.2 sessions to non-CICS Transaction Server for OS/390 systems (for example, CICS/6000) that do not use the CICS Transaction Server for OS/390 in-doubt architecture, CICS can indicate that the subordinate should wait by forcing session outage.

If any resources cannot wait for the coordinator's in-doubt resolution, a decision is taken for the transaction in accordance with the ACTION attribute. In practice, the only circumstances that force decisions in this way are updates to transient data queues with WAIT(NO) specified in the TDQUEUE definition, and installations of terminal-related resources. The latter are normally installed using an INSTALL command.

Table 16 shows how the WAIT attribute defined on a TRANSACTION definition and a logically recoverable TDQUEUE definition are resolved when there is a conflict.

NO The UOW is not to wait. CICS immediately takes whatever action is specified on the ACTION attribute.

Table 16. Resolution of WAIT attributes on TRANSACTION and TDQUEUE definitions

TDQUEUE definition attributes	**TRANSACTION definition WAIT attributes**	
	WAIT(YES)	**WAIT(NO)**
WAIT(NO)	The TD WAIT(NO) overrides WAIT(YES) on the TRANSACTION definition. The UOW is forced to either commit or back out, in accordance with the transaction's ACTION attributes.	The UOW is forced to either commit or back out, in accordance with the transaction's ACTION attributes.
WAIT(YES) WAITACTION(QUEUE)	The UOW waits (that is, it is shunted). A request from another task for a lock on the TD queue must wait, and is queued by CICS.	The transaction WAIT(NO) overrides the TDQUEUE definition. The UOW is forced to either commit or back out, in accordance with the transaction's ACTION attributes.
WAIT(YES) WAITACTION(REJECT)	The UOW waits (that is, it is shunted). A request from another task for a lock on the TD queue is rejected with the LOCKED condition.	The transaction WAIT(NO) overrides the TDQUEUE definition. The UOW is forced to either commit or back out, in accordance with the transaction's ACTION attributes.

Note: If the UOW references more than one transient data queue, and the queues have inconsistent WAIT options, WAIT(NO) always takes precedence and overrides a WAIT(YES). Thus a WAIT(NO) on one TDQUEUE definition forces a failed in-doubt UOW to take either the BACKOUT or COMMIT attribute defined on the UOW's TRANSACTION definition.

WAITTIME({<u>00,00,00</u> | *dd,hh,mm*})
specifies how long a transaction is to wait before taking an arbitrary decision about an in-doubt unit of work, based on what is specified in the ACTION attribute.

<u>00,00,00</u>
The transaction waits indefinitely.

dd,hh,mm
The time, in days, hours, and minutes, for which the transaction is to wait. The maximum value is 93,23,59.

WAITTIME takes effect only if WAIT(YES) is specified.

XTPNAME(*value*)
This attribute may be used as an alternative to TPNAME. Enter a hexadecimal string up to 128 characters in length, representing the name of the transaction that may be used by an APPC partner. All hexadecimal combinations are

acceptable **except X'40'**. To specify an XTPNAME more than 72 characters long to DFHCSDUP, put an asterisk in column 72. This causes the following line to be concatenated to the current line.

XTRANID(*value***)**
You can use this optional attribute to specify another name to be used instead of the TRANSACTION name for initiating transactions. The name may be up to eight hexadecimal digits in length. Because XTRANID is specified in hexadecimal code, you can use a name that contains unprintable characters, or characters not allowed for TRANSACTION names.

(See also TASKREQ, another transaction alias that can be specified.)

value A 4-byte transaction identifier in hexadecimal notation (the identifier therefore uses up to eight hexadecimal digits). If you specify fewer than eight hexadecimal digits, the identifier is padded on the right with blanks. XTRANID should not begin with X'C3', or anything less than or equal to X'40' and should not end with X'FFFFFF', because all these values are reserved for use by CICS. A value of X'00000000' is also not allowed.

TSMODEL

The TSMODEL resource allows you to specify a TS queue name prefix, and associate attributes with that name. You can also map names directly to a shared TS pool (without the need for a shared sysid).

Defining a TSMODEL

```
 TSmodel      ==>
 Group        ==>
 Description  ==>
 PRefix       ==>
 XPrefix      ==>
 Location     ==> Auxiliary                    Auxiliary | Main
RECOVERY ATTRIBUTES
 RECovery     ==> No                           No | Yes
SECURITY ATTRIBUTES
 Security     ==> No                           No | Yes
SHARED ATTRIBUTES
 POolname     ==>
REMOTE ATTRIBUTES
 REMOTESystem ==>
 REMOTEPrefix ==>
 XRemotepfx   ==>
```

Figure 47. The DEFINE panel for TSMODEL

Options

DESCRIPTION(*text*)
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

LOCATION({AUXILIARY|MAIN})
specifies whether the queue is to be held in auxiliary or main storage:

AUXILIARY
queues matching this model are to be held on auxiliary storage. Whatever is specified on the API request is disregarded.

MAIN
queues matching this model are to be held in main storage. Whatever is specified on the API request is disregarded.

Notes:

1. TSMODEL definitions created using the Migrate command have their location attribute set to the default value *AUXILIARY*.
2. LOCATION is ignored for remote TSMODELs and shared TS pool models. Using LOCATION on a remote entry allows the same definition to be installed in both a remote region. See "Dual-purpose resource definition" on page 15.

POOLNAME(*name*)
specifies the 8-character name of the shared TS pool definition that you want to use with this TSMODEL definition. The name can be up to eight characters in length. The acceptable characters are A-Z 0-9 $ @ # and _. Embedded blanks are not acceptable and a name consisting entirely of blanks is treated as though no Poolname had been supplied.

You cannot specify POOLNAME if REMOTESYSTEM is also specified, in which case POOLNAME is ignored.

Note: CICS does not search for a matching TSMODEL if an application program specifies a SYSID on the EXEC CICS temporary storage command, or if a SYSID is added by an XTSEREQ global user exit program. To enable CICS to find the name of a temporary storage data sharing pool when the application program explicitly specifies a SYSID, you need to use a TST with a suitable TYPE=SHARED entry. See “MIGRATE” on page 102 for more information about migrating a TST with TYPE=SHARED entries.

PREFIX(*name*)
specifies the character string that is to be used as the prefix for this model. The prefix may be up to 16 characters in length. The acceptable characters are A-Z 0-9 $ @ % # ¢ ″ ! ¬ | _ and -.

Generic prefix names are allowed, using a single wildcard character, the + sign. Embedded blanks are not acceptable, but a name consisting entirely of blanks is allowed; such a model matches all queues.

Note: CICS searches the installed models to find the *best match*, and uses the TSMODEL with the most specific prefix. For example, if there are installed TSMODEL definitions for DF, and DFH, and DFHX, CICS selects the DFHX model for a TS request that specifies the DFHX123 queue, regardless of the order in which the TSMODELs are installed from the CSD. This behavior is different from that when CICS is searching a TST (see “TST—temporary storage table” on page 484 for information about how CICS matches requests against TST entries.)

RECOVERY({NO|YES})
specifies whether or not queues matching this model are to be recoverable.

NO queues matching this model are to be non-recoverable.

YES queues matching this model are to be recoverable.

RECOVERY(YES) not allowed with LOCATION(MAIN).

REMOTEPREFIX(*data-area*)
specifies the character string that is to be used as the prefix on the remote system. The prefix may be up to 16 characters in length. The acceptable characters are A-Z 0-9 $ @ % # ¢ ″ ! ¬ | _ and -.

Generic prefix names are allowed, using a single wildcard character, the + sign. Embedded blanks are not acceptable, but a name consisting entirely of blanks is allowed. In this case, the prefix value is used, but if that too is blank, the model matches all queues.

REMOTEPREFIX is not allowed unless REMOTESYSTEM is also specified.

If REMOTEPREFIX is specified:
- Its length must be the same as the length of Prefix.
- If wild characters are used, they must be in the same position in Prefix and Remoteprefix as in the following example:

```
Prefix:       A++D
Remoteprefix: X++Y
```

REMOTESYSTEM(*data-area*)
specifies a 4-character remote system name. The acceptable characters are A-Z 0-9 $ @ and #. Embedded blanks are not acceptable and a name consisting entirely of blanks is treated as though no Remotesystem had been specified.

REMOTESYSTEM and POOLNAME are mutually exclusive. If REMOTESYSTEM is specified, POOLNAME is ignored.

SECURITY({NO|YES})
specifies whether security checking is to be performed for queues matching this model.

NO security checking is not to be performed for queues matching this model.

YES security checking is to be performed for queues matching this model.

TSMODEL(*name*)
specifies the name of this TSMODEL definition. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ # . / - _ % & ¢ ? ! : | " = ¬ , ; < and >. Embedded blanks are not acceptable.

This name is used to identify the TSMODEL definition on the CSD file. It is not used within the active CICS system.

XPREFIX(*name*)
may be used as an alternative to PREFIX. Enter a hexadecimal string, up to 32-characters in length, that is to be used as the prefix for this model.

XREMOTEPFX(*name*)
may be used as an alternative to REMOTEPREFIX. Enter a hexadecimal string, up to 32-characters in length, that is to be used as the prefix on the remote system.

TYPETERM

A TYPETERM definition is an extension of a TERMINAL definition; it specifies a set of attributes common to a group of terminals. The TYPETERM attributes are listed on page 341. If you have a number of terminals with the same properties, you would define one TYPETERM with the required values, and then name that TYPETERM in each TERMINAL definition (or in the autoinstall model definition if you are using autoinstall).

Each TERMINAL definition must name a TYPETERM definition. This single attribute represents many other characteristics, and thus can save considerable effort, and reduce the chance of making mistakes. TYPETERMs make it easier to define your terminals if you have many terminals of the same kind.

Two TYPETERM attributes are worthy of note here, because they further simplify the terminal definition process:
- DEVICE
- QUERY

DEVICE

specifies the **device type** that the TYPETERM represents. This is a key attribute, because the default values for a number of other attributes depend on the value you supply for it. You must supply a value for the DEVICE attribute, because there is no default.

Some attributes are always the same for every device of the same type. You do not need to define all these attributes yourself, because RDO knows what they are. All you need to tell RDO is the device type of your terminals, when you define the TYPETERM for them. Values for the fixed attributes are supplied automatically.

For a list of terminals supported by RDO, see “Devices supported” on page 336. There is also a list of valid values for the DEVICE attribute of TYPETERM in Table 17 on page 335. This shows you the other attribute values supplied for different device types. In some cases, these values depend also on your values for SESSIONTYPE and TERMMODEL, but these too have defaults that depend on the DEVICE specified.

Apart from ordinary display devices, printers, and other more specialized input and output devices, you can create a TYPETERM definition for your CICS consoles.

QUERY

The QUERY attribute allows you to leave some features of your terminals undefined until they are connected. Information about these attributes can then be obtained by CICS itself using the QUERY structured field.

All attributes for which you can use QUERY are also TYPETERM attributes. They are:

ALTPAGE
ALTSCREEN
APLTEXT
BACKTRANS
CGCSGID
COLOR
EXTENDEDDS
HILIGHT
MSRCONTROL

OBFORMAT
OUTLINE
PARTITIONS
PROGSYMBOLS
SOSI
VALIDATION

The use of QUERY overrides any value that is explicitly defined for any of the TYPETERM attributes listed above, **except ALTSCREEN**. QUERY-supplied ALTSCREEN values are used only if no ALTSCREEN value is explicitly defined in the TYPETERM.

You can use QUERY for 3270 devices with the extended 3270 data stream. The DEVICE types for which you can use QUERY are:
3270
3270P
LUTYPE2
LUTYPE3
SCSPRINT

You can specify that QUERY be used in one of two ways:

- QUERY(COLD) specifies that the QUERY is to be issued only when the terminal is first connected after an initial or a cold start.
- QUERY(ALL) specifies that the QUERY is to be issued each time the terminal is connected.

The QUERY function is particularly useful with configurable devices, such as the IBM Personal System/2 (PS/2) and the IBM 3290. It enables you to reconfigure the device between logging off and logging on to CICS, without having to change any resource definitions.

The QUERY function is also particularly useful when used in conjunction with autoinstall.

Note that the QUERY facility obtains only the information required by CICS. If an application program needs to determine other device characteristics, it still needs to send a QUERY structured field and analyze the reply.

To summarize, you may need only one TYPETERM definition for each device type. If the attributes that can be determined by QUERY differ among the terminals, you still need only one TYPETERM for each device type. If other attributes of your terminals vary, you may need more than one TYPETERM definition for a device type.

There are some CICS-supplied TYPETERM definitions suitable for the more frequently used terminals. These are described in “TYPETERM definitions in group DFHTYPE” on page 517.

Terminal definition—summary

When all your terminals are basically the same, you can have only one TYPETERM definition, and one TERMINAL definition with AUTINSTMODEL(YES). You might like to use QUERY to deal with different features used by your terminals.

Dependent default values

See “Dependent default values” on page 20 for an explanation of dependent default values.

Table 17. Default values for different devices

Values specified by you:			Default values supplied by RDO:					
DEVICE	**SESSION TYPE**	**TERM MODEL**	**DEF. SCRN.**	**PAGE SIZE**	**AUTO PAGE**	**BRAC -KET**	**BUILD CHAIN**	**ROUTD MSGS**
3270 (3277, L3277) (a)		1	12,40	12,40	N	Y	N	All
3270		2 (dft)	24,80	24,80	N	Y	N	All
3275		1	12,40	12,40	N	Y	N	All
3275		2 (dft)	24,80	24,80	N	Y	N	All
3270P (3284, L3284, 3286, L3286) (b)		1	12,40	12,40	Y	Y	N	All
3270P		2 (dft)	24,80	24,80	Y	Y	N	All
APPC			0,0	1,40	Y	Y*	Y*	None*
CONSOLE			0,0*	1,124*	N	N*	N*	None
LUTYPE2		1	12,40	12,40	N	Y*	Y*	All
LUTYPE2		2 (dft)	24,80	24,80	N	Y*	Y*	All
LUTYPE3		1	12,40	12,40	Y	Y*	N	All
LUTYPE3		2 (dft)	24,80	24,80	Y	Y*	N	All
LUTYPE4			0,0	50,80	Y	Y*	N	All
BCHLU	(dft)		0,0	12,80	Y	Y*	N	All
BCHLU	BATCHDI		0,0	12,80	Y	Y*	N	All
BCHLU	USERPROG		0,0	12,80	Y	Y*	N	All
INTLU			0,0	12,80	Y	Y	N	All
SCSPRINT			0,0*	24,80	Y	Y*	N	All
TLX or TWX	CONTLU (dft)		0,0	1,40	Y	Y	N	All
TLX or TWX	INTLU		0,0	1,40	Y	Y	N	All
3600	(dft)		0,0	1,40	Y	Y	N	All
3600	PIPELINE		0,0	6,30	Y	Y	N	All
3614			0,0	1,40	Y	Y	N	All
3650	USERPROG (dft)		0,0	3,80	Y	Y	N	All
3650	3270		12,40	23,80	Y	Y	N	All
3650	3653		0,0	6,30	Y	Y	N	All
3650	PIPELINE		0,0	6,30	Y	Y	N	All
3767			0,0	12,80	Y	Y*	N	All
3767C			0,0	12,80	Y	Y*	N	All
3767I			0,0	12,80	Y	Y*	N	All
3770	(dft)		0,0	12,80	Y	Y*	N	All
3770	USERPROG		0,0	12,80	Y	Y*	N	All
3770	BATCHDI		0,0	12,80	Y	Y*	N	All
3770B	(dft)		0,0	12,80	Y	Y*	N	All
3770B	BATCHDI		0,0	12,80	Y	Y*	N	All

Table 17. Default values for different devices (continued)

Values specified by you:			Default values supplied by RDO:					
3770B	USERPROG		0,0	12,80	Y	Y*	N	All
3770C			0,0	12,80	Y	Y*	N	All
3770I			0,0	12,80	Y	Y*	N	All
3790	(dft)		0,0	1,40	Y	Y*	N	None*
3790	SCSPRINT		0,0	24,80	Y	Y*	N	All
3790	USERPROG		0,0	12,80	Y	Y*	N	All
3790	BATCHDI		0,0	12,80	Y	Y*	N	All
3790	3277CM (c)	1	12,40	12,40	N	Y*	Y*	All
3790	3277CM (d)	2 (dft)	24,80	24,80	N	Y*	Y*	All
3790	3284CM (e)	1	12,40	12,40	Y	Y*	N	All
3790	3284CM (f)	2 (dft)	24,80	24,80	Y	Y*	N	All
3790	3286CM (e)	1	12,40	12,40	Y	Y*	N	All
3790	3286CM (f)	2 (dft)	24,80	24,80	Y	Y*	N	All

Notes: * denotes that this value is forced; it is not just a default but is mandatory for the DEVICE type. (a) 3277 and L3277 are converted by RDO to 3270. (b) 3284, 3286, L3284, and L3286 are converted to 3270P. (dft) indicates the default SESSIONTYPE or TERMMODEL value assigned by RDO if you specify the DEVICE only. (c) is converted to DEVICE(LUTYPE2) TERMMODEL(1). (d) is converted to DEVICE(LUTYPE2) TERMMODEL(2). (e) is converted to DEVICE(LUTYPE3) TERMMODEL(1). (f) is converted to DEVICE(LUTYPE3) TERMMODEL(2).

Devices supported

This section lists the device names that you can use on the TYPETERM.

Table 18 is a list of terminal types supported by RDO.

To use Table 18, find the family number of your device in the leftmost column (headed **Terminal or System Type**). Then look across at the second column (headed **Units**) to see if your type of units is specifically mentioned. Next look at the third column (headed **Attachment**) to see what device type you should use on your TYPETERM definition.

There is further explanation of the information in Table 18 in the notes that follow. The following abbreviations have been used:

local	channel or adapter attached
s/s	start/stop transmission
SDLC	synchronous data link control
sw	switched
BSC	binary synchronous
nonsw	nonswitched communications

Table 18. VTAM terminals and subsystems supported by RDO

Terminal or System Type	Units	Attachment	Notes
3101		Supported as TWX 33/35	1,10

Table 18. VTAM terminals and subsystems supported by RDO (continued)

Terminal or System Type	Units	Attachment	Notes
3230		Supported as 2741 (BSC/BTAM) or INTLU (VTAM)	1
3270	3178, 3179, 3180, 3262, 3271, 3272, 3274, 3276, 3290	local, SDLC, BSC nonsw	1,2, 3
	3275 3277, 3278, 3279, 3284, 3286, 3287, 3288, 3289	BSC sw or nonsw	3
3270PC	3270PC, 3270PC/G, 3270PC/GX	Supported as 3270	
3287	models 11, 12	SDLC supported as SCSPRT	13
3600	3601, 3602, 3690, 3604, 3610, 3612, 3618, 3614, 3624	SDLC, BSC nonsw	1,4,5,14
3630	3631, 3632, 3643, 3604	attached as 3600	4,11
3640	3641, 3644, 3646, 3647	SDLC attached as INTLU	13
	3642	SDLC attached as SCSPRT	13
	3643	SDLC supported as LUTYPE2	13
	3645	SDLC supported as SCSPRT	13
3650	3651, 3653, 3275, 3284	SDLC	4
3680	3684	Supported as 3790/3650	4
3730	3791	Supported as 3790	4
3767		SDLC s/s supported as 2740/2741	1
3770	3771, 3773, 3774	SDLC	1,4,6
	3775, 3776, 3777	BSC supported as 2770	
3790	3791	SDLC or local	4,7
4300	4331, 4341, 4361, 4381	BSC or SDLC	4,8
4700	4701-1	Supported as 3600	1,4,5
5280		Supported as 3741 (BSC/BTAM) or 3270 (BTAM or VTAM)	1
5520		SDLC supported as 3790 full-function LU BSC supported as 2770 SDLC attached as APPC	1,4
5550		Supported as 3270	
5937		SDLC/BSC attached as 3270	1,3
6670		SDLC BSC supported as 2770	1
8100	8130/8140 processors with DPCX	Supported as 3790	4
	DPPX/BASE using Host Presentation Services or Host Transaction Facility	Attached as 3790	4
	DPPX/DSC or DPCX/DSC (including 8775 attach)	Supported as 3270	4,12
8775		SDLC supported as LUTYPE2	
8815		Supported as APPC	
Displaywriter		Supported as APPC SNA for EDDS Supported as 3270; attached as 2741 (s/s) or 3780 (BSC) SDLC attached as APPC	1

Table 18. VTAM terminals and subsystems supported by RDO (continued)

Terminal or System Type	Units	Attachment	Notes
Personal Computer		Supported as 3270 and as APPC	1,14
PS/2		Supported as 3270 and as APPC	1,14
Scanmaster		Supported as APPC	
Series/1		Attached as System/3;supported as 3650 Pipeline (VTAM) or 3790 (full function LU)	1,4
System/32	5320	SDLC supported as 3770	1,4,9
		BSC supported as 2770	4
System/34	5340	SDLC supported as 3770 BSC attached as System/3	1,4,94
System/36™		Supported as System/34 SDLC attached as APPC	1
System/38	5381	SDLC attached as 3770 SDLC attached as APPC BSC attached as System/3	1,4,9 4
AS/400	5381	SDLC attached as 3770 SDLC attached as APPC BSC attached as System/3	1,4,9 4
System/370		BSC or SDLC and APPC	1,4,8
System/390®		BSC or SDLC and APPC	1,4,8
TWX 33/35		VTAM (via NTO) ss sw BTAM ss sw	1,10
WTTY		VTAM (via NTO) ss nonsw BTAM ss sw	1,10

Notes:

1. These devices are also supported by BTAM, and you may connect them to CICS by using transaction routing from CICS releases before CICS/ESA 3.1.
2. CICS supports the 3290 through both the terminal control and the BMS interfaces. The 3290 can be in one of three states: default, alternate, or partitioned. Up to 16 partitions can be defined. The 3290 also has the programmed symbols and extended highlighting features, as well as two kinds of data validation feature, mandatory fill and mandatory enter. A 3290 terminal can be configured as from one to five logical units. You define the size of each logical unit when setting up the 3290. You must ensure that the resource definitions of each logical unit match the set-up size, to prevent unpredictable results. Up to four interactive screens in any configuration can be active at the same time, but only one interactive screen can be defined as having programmed symbols at any time; that is, all programmed symbol sets must be assigned to the interactive screen.

   To display long lines of data, such as the 132-character lines of CEMT output, specify a default screen width of 132 characters.

   If you intend to use the large buffer, you might have to specify a much larger value for IOAREALEN (see page 351). Whether you need to do this depends on whether operators are likely to modify, or enter, large quantities of data. If a terminal is used for inquiry only, or for limited data entry, IOAREALEN need not be large.
3. SDLC 3270s are supported only through VTAM, and the switched BSC 3275 (feature #3440) is supported only through BTAM. Printers attached to local or

SDLC 3274s and SDLC 3276s are supported through VTAM either as LU Type 3 using the 3270 printer data stream or as LU Type 1 using the SCS data stream (which is a subset of that used for SDLC 3767, 3770, and 3790 printers). The 3288 is supported as a 3286 Model 2. CICS supports the 3270 copy feature (#1550).

4. Devices and features supported by a system or programmable controller are generally transparent to CICS. In some cases, CICS provides specific device support, in which case the units are listed.
5. SDLC is supported through VTAM. BSC attachment (RPQ 8K0598) is supported through BTAM. The 3614 is supported both for loop attachment to the 3601 and SDLC attachment to the host via a 3704/3705 Communications Controller.

   The 3614 is supported by CICS as BSC only when loop-attached to the 3601/3602 Controllers. The 3624 is supported as a 3614.

   The 3690 is supported as a 3602.
6. CICS supports the Data Transfer Function of the SDLC Programmable Models of the 3770 Data Communication System. In using this function, you are responsible for allocating data sets and managing the program library.
7. CICS does not support the 3790/Data Entry Configuration using 3760s. The #9165 or #9169 configuration is required to support the CICS enhancement first made available in Version 1 Release 3.0.

   Printers on 3790 systems are supported with one of the following:
   - A function program that you provide
   - 3270 data stream compatibility with a 3270 printer data stream (LU3)
   - An SCS data stream supporting a subset of that for SDLC 3767 (LU1)

   When operating in 3270 mode, the 3288 Model 2 is supported as a 3286 Model 2.
8. System/370 and 4300 attachment by BSC requires a suitable telecommunications program (for example, the VSE/3270 Bisync Pass Through Program) in the system connected to CICS. CICS is **not** a suitable program for the remote CPU. Attachment by SDLC is supported by CICS intersystem communication.
9. The System/32 with its SNA/SDLC workstation system utility program, and the System/34 and System/38™, are supported as compatible versions of an appropriately featured 3770 Communication System operating as a batch logical unit. The System/34 or System/38 user-written program is responsible for supporting the correct SNA sequences of the attached subsystem.
10. TWX and WTTY are supported through VTAM via the Network Terminal Option program product (5735-XX7), with attachments as defined by NTO.

    TWX (Line Control Type) is attached through BTAM on eight-level code at 110 bps on common-carrier switched 150 bps networks.

    WTTY is attached at 50 bps on common-carrier switched networks where the terminals supported are those interfacing through IBM World Trade Corporation Telegraph Terminal Control with Telegraph Line Adapter.

    The transmission code used is International Telegraph Alphabet No. 2 (CCITT No. 2). CICS does not support autocall or automatic host disconnect via WRITE break.
11. The 3643 is supported as a 3604.
12. 8775 support includes validation of mandatory fill and mandatory enter field attributes.

13. Attachment is via the Loop Adapter #4830, #4831, and Data Link Adapter #4840 of the 4331 processor.
14. 3270 support requires that the 3278/3279 Emulation Adaptor be installed in the Personal Computer or PS/2.
15. The 3600 pipeline logical unit is designed to provide high throughput for particular types of transaction, such as credit card authorization or general inquiry applications. To achieve a high throughput of inquiry messages and their replies, the CICS pipeline session uses a restricted set of the communication protocols that are used with the 3601 logical unit.

    These restrictions result in a full duplex message flow whereby many inquiry messages are outstanding at any one time, and the replies may flow back in a different order from that of the original inquiries. The 4700/3600 application program controlling the inquiry terminals is responsible for maintaining the protocols as well as correlating replies with inquiries, and controlling message flow to the group of terminals associated with the pipeline logical unit.

Defining a TYPETERM

```
    TYpeterm     ==>
    Group        ==>
    DEScription  ==>
   RESOURCE TYPE
    DEVice       ==>                    (See
Table 17 on page 335)
    TERmmodel    ==>                    (See
Table 17 on page 335)
    SESsiontype  ==>                    (See
Table 17 on page 335)
    LDclist      ==>
    SHippable    ==> No                   No | Yes
   MAPPING PROPERTIES
    PAGesize     ==> 024 , 080          0-255
    ALTPage      ==> 000 , 000          0-255
    ALTSUffix    ==>
    FMhparm      ==> No                 No | Yes
    OBOperid     ==> No                 No | Yes
   PAGING PROPERTIES
    AUTOPage     ==>                    No | Yes
   DEVICE PROPERTIES
    DEFscreen    ==> 000 , 000          0-255
    ALTSCreen    ==>     ,              0-255
    APLKybd      ==> No                 No | Yes
    APLText      ==> No                 No | Yes
    AUDiblealarm ==> No                 No | Yes
    COLor        ==> No                 No | Yes
    COPy         ==> No                 No | Yes
    DUalcasekybd ==> No                 No | Yes
    EXtendedds   ==> No                 No | Yes
    HIlight      ==> No                 No | Yes
    Katakana     ==> No                 No | Yes
    LIghtpen     ==> No                 No | Yes
    Msrcontrol   ==> No                 No | Yes
    OBFormat     ==> No                 No | Yes
    PARtitions   ==> No                 No | Yes
    PRIntadapter ==> No                 No | Yes
    PROgsymbols  ==> No                 No | Yes
    VAlidation   ==> No                 No | Yes
    FOrmfeed     ==> No                 No | Yes
    HOrizform    ==> No                 No | Yes
    VErticalform ==> No                 No | Yes
    TEXTKybd     ==> No                 No | Yes
    TEXTPrint    ==> No                 No | Yes
    Query        ==> No                 No | Cold | All
    OUtline      ==> No                 No | Yes
    SOsi         ==> No                 No | Yes
    BAcktrans    ==> No                 No | Yes
    CGcsgid      ==> 00000 , 00000      0-65535
   SESSION PROPERTIES
    AScii        ==> No                 No | 7 | 8
    SENdsize     ==> 00000              0-30720
    RECEivesize  ==>                    0-30720
    BRacket      ==> Yes                Yes | No
    LOGMODE      ==>
    LOGMODECom   ==> No                 No | Yes
   DIAGNOSTIC DISPLAY
    ERRLastline  ==> No                 No | Yes
    ERRIntensify ==> No                 No | Yes
    ERRColor     ==> NO                 NO | Blue | Red | Pink | Green
                                        | Turquoise  | Yellow | NEutral
    ERRHilight   ==> No                 No  | Blink | Reverse | Underline
                                               continued...
```

Figure 48. The DEFINE panel for TYPETERM (Part 1 of 2)

```
...continued
OPERATIONAL PROPERTIES
 AUTOConnect  ==> No                 No | Yes | All
 ATi          ==> No                 No | Yes
 TTi          ==> Yes                Yes | No
 CReatesess   ==> No                 No | Yes
 RELreq       ==> No                 No | Yes
 DIscreq      ==> Yes                Yes | No
 Nepclass     ==> 000                0-255
 SIgnoff      ==> Yes                Yes | No | Logoff
 Xrfsignoff   ==> Noforce            Noforce | Force
MESSAGE RECEIVING PROPERTIES
 ROutedmsgs   ==>                    All | None | Specific
 LOGOnmsg     ==> No                 No | Yes
APPLICATION FEATURES
 BUildchain   ==> No                 No | Yes
 USerarealen  ==> 000                0-255
 Ioarealen    ==> 00000 , 00000      0-32767
 UCtran       ==> No                 No | Yes | Tranid
RECOVERY
 RECOVOption  ==> Sysdefault         Sysdefault | Clearconv | Releasesess
                                     | Uncondrel | None
 RECOVNotify  ==> None               None | Message | Transaction
```

Figure 48. The DEFINE panel for TYPETERM (Part 2 of 2)

Options

ALTPAGE({0|*rows*},{0|*columns*})
specifies the page size to be used by BMS for this terminal entry when ALTSCREEN has been selected as the screen size. The default is the PAGESIZE. The values for both *rows* and *columns* must be in the range 0 through 999.

You will get unexpected results if the *columns* value of ALTPAGE is different from that of ALTSCREEN. The *rows* value of ALTPAGE can usefully be less than that of ALTSCREEN, perhaps to reserve the bottom line of the screen for error messages.

If you use the QUERY structured field (see page 333), the alternate page size used is the size set up as the alternate screen size. For terminals that can be queried, you can set ALTPAGE to zero and have the ALTSCREEN value defined explicitly by the CINIT BIND. If ALTPAGE is not zero, it is possible to have different values for the ALTPAGE and the ALTSCREEN.

The product of *rows* and *columns* must not exceed 32767.

ALTSCREEN(*rows,columns*)
specifies the 3270 screen size to be used for a transaction that has SCRNSIZE(ALTERNATE) specified in its profile. The values that can be specified are:

Device	Alternate Screen Size
3276-1, 3278-1	(12,80)
3276-2, 3278-2	(24,80)
3276-3, 3278-3	(32,80)
3276-4, 3278-4	(43,80)

Device	Alternate Screen Size
3278-5	(27,132)
3279-2A, 3279-2B	(24,80)
3279-3A, 3279-3B	(32,80)

No validity checking is performed on the screen size selected, and incorrect sizes may lead to unpredictable results.

For BSC devices, both the alternate and default screen sizes are determined by the device hardware. The alternate screen size is the maximum screen size. For the 3290 display, both the default and alternate screen sizes are determined by the customer setup procedure. For further guidance, see the *IBM 3290 Information Panel Description and Reference* manual.

For SNA devices (LUTYPE2 and LUTYPE3), you can specify any value for both alternate and default screen sizes, up to the maximum physical screen size. In particular, both the alternate and default screen sizes can be the maximum screen size, or the default screen size can be the maximum screen size with no alternate screen size specified. The SNA bind is generated by CICS from this information. You do not need to provide logmode table entries, or to customize the device.

For non-SNA 3270 and LUTYPE2 devices, you can use the QUERY structured field to determine the alternate screen size that has been set up for the display. To use QUERY, leave the DEFSCREEN to default to (24,80) and leave ALTSCREEN unspecified. The alternate screen size is the size set up by the terminal user. Otherwise, QUERY(COLD) or QUERY(ALL) has no effect on the alternate screen size. Leaving ALTSCREEN unspecified without using QUERY under the conditions above results in an alternate screen size of (00,00). For more information about QUERY, see page 333.

If you use dual screen sizes, you can make a CICS transaction use the alternate screen size by coding SCRNSIZE(ALTERNATE) in its associated profile. If an application consists of several pseudo-conversationally linked transactions, specify SCRNSIZE(ALTERNATE) in the profile for each of these transactions if the application uses the alternate screen size.

For 3287 and 3289 printers, the value specified must equal the buffer size of the particular device. For non-SNA 3287 and 3289 printers, the sizes depend on the feature ordered, not on the model number. For SNA printers, there are no features, and any two sizes can be specified from the list of valid sizes. When printing to a printer whose associated TERMINAL definition has PRINTERCOPY(YES) specified, the ALTSCREEN value should match the screen size of the terminal whose screen is to be printed. If the values differ, unpredictable results may occur.

ALTSUFFIX({blank|*number*})
A 1-character numeric suffix (specified in the SUFFIX operand of the application programmer's DFHMSD TYPE={DSECT|MAP} macro).

blank Leave this attribute blank if you do not want a suffixed map set.

number
BMS appends this suffix to map set names if the screen size being used is the same value as the alternate screen size; that is, if the transaction

has SCRNSZE(ALTERNATE) specified in the PROFILE definition, or if the default and alternate screen size are the same. In this case, BMS map selection routines attempt to load the map set with the suffix specified in the ALTSUFFIX operand.

If there is no such map set, BMS tries to load a map set suffixed with M or L and, if this load fails, BMS tries to load an unsuffixed map set version.

If the transaction uses default screen size, BMS first tries to load a map set suffixed with M or L and, if this load fails, BMS tries to load an unsuffixed map set version.

To use a suffixed map set, you must specify the BMS=(,,,DDS) system initialization parameter.

APLKYBD({NO|YES})
specifies whether the 3270 device has the APL keyboard feature.

APLTEXT({NO|YES})
specifies whether the 3270 device has the APL text feature. Do not specify YES for a 3288 printer, with or without TEXTPRINT(YES). The APLTEXT feature is used in conjunction with the TEXTKYBD and APLKYBD operands.

You can use the QUERY structured field to determine whether the device is set up to use the APL text feature (see page 333).

ASCII({NO|7|8})
specifies whether the terminal has an ASCII feature.

NO This terminal does not have an ASCII feature.

7 Specify this to communicate with ASCII-7 terminals. Devices configured with the ASCII-7 feature must be LUTYPE2 or LUTYPE3 without extended 3270 features. Only the following devices are supported:
3274 Model 1C and 51C
3276 Model 12
3278
3287

Any terminal configured with the ASCII-7 option has all FM data outbound from CICS converted to ASCII-7, and all FM data inbound to CICS converted to EBCDIC. Only FM request data is translated. All other data in the RU such as LU status or sense data is assumed to be in EBCDIC on output. ASCII-7 does **not** support data streams that contain extended attributes, such as structured fields and function management headers.

The ASCII-7 support is available on 3274-1C as an option on the configuration of the standard microcode. The use of the ASCII-7 option is determined at session initiation by BIND parameters set by CICS as a result of the TCT definition described above.

8 Specify this to communicate with ASCII-8 terminals. Devices configured with the ASCII-8 feature can be LUTYPE1, LUTYPE2, or LUTYPE3 with or without extended 3270 and SCS data stream features.

Any terminal configured with the ASCII-8 option has all FM data outbound from CICS converted to ASCII-8, and all FM data inbound to

CICS converted to EBCDIC. All FM request data is translated. This includes the AID, cursor address, FM headers and structured fields. Any other form of the RU such as LU status or sense data is assumed to be in EBCDIC on input and is transmitted in EBCDIC on output.

Note that this ASCII-8 support is intended only for devices that operate in EBCDIC but translate or retranslate the data stream to or from ASCII-8, as is done by this CICS support. This is because the data stream is treated as a character string, and any binary number fields are translated byte by byte as though they were graphic characters. Thus they may not represent their true value while in ASCII form.

The ASCII-8 support is available as a microcode RPQ on the 3274 and is mutually exclusive with the ASCII-7 option. The use of the ASCII-8 option is determined at session initiation by BIND parameters set by CICS as a result of the TCT definitions described above.

ATI({NO | YES})

specifies whether transactions can start at the terminal by automatic transaction initiation. ATI(YES) allows transactions to be started at the terminal by transient data control or by an EXEC CICS START command issued by another transaction. If there is already a transaction at the terminal, the ATI transaction is held until it ends.

If you specify ATI(YES), you must specify an IOAREALEN of at least one byte, except for DEVICE(APPC) when ATI and IOAREALEN have forced default values of YES and 0.

If ATI is specified as YES and CREATESESS is specified as YES, and if a transaction is initiated when the terminal is not ACQUIRED, it is automatically acquired.

See also the TTI attribute.

AUDIBLEALARM({NO | YES})

specifies whether the audible alarm feature is installed for a 3270 display or for a 3270 printer attached to a 3651 controller.

AUTOCONNECT({NO | YES | ALL})

AUTOCONNECT(YES) or (ALL) specifies that the session with the terminal is to be established (that is, BIND is to be performed) during CICS initialization, or when communication with VTAM is started using the CEMT SET VTAM OPEN command. If the connection cannot be made at this time because the terminal is unavailable, the link must be subsequently acquired using the CEMT SET TERMINAL(*termid*) INSERVICE ACQUIRED command, unless the terminal becomes available in the meantime and itself initiates communications.

Note: If you use the VTAM LOGAPPL function, do not specify AUTOCONNECT(YES), because this can lead to race conditions causing errors or hung logical units.

NO CICS does not attempt to bind sessions when the connection is established.

YES CICS attempts to bind as a contention winner session, when the connection is established.

ALL Not applicable.

For background information about AUTOCONNECT, see the *CICS Intercommunication Guide*.

AUTOPAGE({NO | YES})
specifies whether BMS autopaging is to be used. See page 287 for details. Specify YES for printer TYPETERMs and NO for display device TYPETERMs. The default depends on the value you specify for the DEVICE attribute: the default values are indicated in Table 17 on page 335.

BACKTRANS({NO | YES})
specifies whether the device has the background transparency feature.

You can use the QUERY structured field to determine whether the device is set up to use the background transparency feature (see page 333).

BRACKET({YES | NO})
specifies whether bracket protocol is to be enforced for this logical unit. The default depends on the value you specify for the DEVICE attribute (see Table 17 on page 335).

YES Bracket protocol is to be used. This option is required for the 3790 inquiry and full function logical units. BRACKET(YES) is forced for many DEVICE types (see Table 17 on page 335).

NO Bracket protocol is not to be used. You must specify BRACKET(NO) for a 3614 logical unit and the 3650 Host Command Processor (HCP) session.

BUILDCHAIN({NO | YES})
specifies whether CICS is to perform chain assembly prior to passing the input data to the application program.

The default depends on the value you specify for the DEVICE attribute (see Table 17 on page 335).

NO Any terminal input/output area (TIOA) received by an application program from this logical unit contains one request unit (RU).

YES Any TIOA received by an application program from this logical unit contains a complete chain.

CGCSGID({0,0 | *value1,value2*})
The coded graphic character set global identifier (CGCSGID) enables application programs to determine the character set supported at the device.

You can get this information from a QUERY structured field for some devices (see page 333). For others, you must supply this information here, so that application programs can retrieve it using the EXEC CICS ASSIGN command.

0,0 No CGCSGID is specified.

value1,value2
The CGCSGID consists of two 5-digit decimal numbers which can take values in the range 1 through 65535. *Value1* is the graphic character set global identifier (GCSGID) and *value2* is a specification of the code points for the set, the code page global identifier (CPGID).

COLOR({NO | YES})
specifies whether the 3270 device or the SCS printer has the extended color feature, which allows colors to be selected for each field or character.

You can use the QUERY structured field to determine whether the device is set up to use the color feature. (See page 333).

COPY({NO|YES})
specifies whether the copy feature for a 3270 display or printer is included in the 3270 control unit. Leave it to default to COPY(NO) for 3270 compatibility mode logical units, because COPY(YES) is ignored.

See also the PRINTERCOPY and ALTPRINTCOPY attributes of the TERMINAL definition.

For further details about screen copying, see the *CICS 3270 Data Stream Device Guide*.

CREATESESS({NO|YES})
specifies whether sessions are to be created.

NO Specify this to prevent internally generated session requests from actually creating a session. During CICS execution, this can be changed only by a CEMT command.

CREATESESS(NO) prevents EXEC START requests and automatic transaction initiation (ATI) requests for this terminal causing a session to be created. This means that the requests are either queued or rejected when no session is currently established.

YES Specify this for a status that allows internally generated session requests to create a session. During CICS execution, this status can be generated only by a CEMT command.

CREATESESS(YES) allows EXEC START requests and automatic transaction initiation (ATI) requests for this terminal to cause a session to be created automatically.

DEFSCREEN({*rows*|*columns*})
specifies the 3270 screen size or 3270 printer page size to be used on this device when attached to a transaction or used by BMS for which SCRNSIZE(DEFAULT) has been specified in the PROFILE definition. The default depends on the value you specify for the DEVICE attribute (see Table 17 on page 335). The values that can be specified for a BSC 3270 are:

Device	**Screen size**
3278-1	(12,40)
3278-2	(24,80)
3276-3, 3278-3	(24,80)
3276-4, 3278-4	(24,80)
3278-5	(24,80)
3279-2A, 3279-2B	(24,80)
3279-3A, 3279-3B	(24,80)

For BSC devices, both default and alternate screen sizes are determined by the terminal hardware. The default screen size is (24,80), except for the 3278-1 where it is (12,40).

For SNA devices (LUTYPE2 and LUTYPE3), both default and alternate screen sizes can be any value you choose, up to the maximum physical screen size (see ALTSCREEN). In particular, both default and alternate screen sizes can be the maximum screen size; or the default screen size can be the maximum screen size with no alternate screen size specified. The SNA bind is generated by CICS from this TCT information. You do not need to provide logmode table entries, or to customize the device.

DESCRIPTION(*text*)
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you may use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

DEVICE(*name*)
specifies the device type which this TYPETERM defines. This attribute is mandatory for all TYPETERM definitions.

If you type DEVICE(*xxxx*), where *xxxx* is a valid device type, on the command line, together with SESSIONTYPE and TERMMODEL if appropriate, other attributes are given appropriate default values. For further guidance, see “Dependent default values” on page 20. Entering or overtyping the DEVICE, SESSIONTYPE, or TERMMODEL values on the overtype-to-modify panel does **not** provide these defaults.

The valid attributes and the defaults for each device type are listed in Table 17 on page 335. The recommended attributes for non-SNA VTAM 3270 devices are 3270 and 3270P for displays and printers, respectively. The following attributes can also be specified and are retained for compatibility with previous releases:
- Displays: 3277 and L3277
- Printers: 3284 and L3284, 3286 and L3286

For SNA VTAM 3270 devices, use the LUTYPE2 or LUTYPE3 attribute as appropriate. LUTYPE2 logical units are those defined by SNA, which accept a 3270-display data stream. LUTYPE3 logical units are those defined by SNA, which accept a data stream similar to that for a 3270 printer.

For a list of device types supported by CICS, see “DFHTCT: CICS terminals list” on page 477. See also Table 17 on page 335 for a list of valid device names and the default attributes that they generate.

DISCREQ=({YES | NO})
specifies whether disconnect requests are to be honored.

YES
CICS is to honor a disconnect request for a VTAM device, and issue a VTAM CLSDST macroinstruction to terminate the VTAM session with that logical unit.

In addition, CESF LOGOFF or GOODNIGHT from the terminal causes disconnection if you specify YES.

YES is essential if the TYPETERM definition is referenced by AUTINSTMODEL TERMINAL definitions, so that autoinstalled terminal entries can be deleted automatically.

NO
CICS is not to honor a disconnect request for a VTAM device.

DUALCASEKYBD({NO|YES})
specifies whether a 3270 display has a typewriter keyboard or an operator console keyboard. Both uppercase and lowercase data can be transmitted with either of these keyboards.

ERRCOLOR({NO|*color*})
specifies whether the error message is to be displayed in color. Coding ERRCOLOR(*color*) implies ERRLASTLINE(YES).

The colors you can specify are:

BLUE
RED
PINK
GREEN
TURQUOISE
YELLOW
NEUTRAL

ERRHILIGHT({NO|BLINK|REVERSE|UNDERLINE})
specifies the highlighting, if any, with which error messages are to be displayed.

ERRINTENSIFY({NO|YES})
specifies whether the error message is to be displayed in an intensified field. Coding ERRINTENSIFY(YES) implies ERRLASTLINE(YES).

ERRLASTLINE({NO|YES})
specifies where error messages are to be displayed.

NO	An error message is displayed at the current cursor position and without any additional attributes.
YES	An error message is displayed starting at the beginning of the line nearest the bottom of the screen so that the whole message fits on the screen. Because all error messages occupy the same line, if the messages are received in quick succession, they overlay one another and earlier messages may disappear before they have been read.

EXTENDEDDS({NO|YES})
specifies whether the 3270 device or the SCS printer supports extensions to the 3270 data stream. EXTENDEDDS(YES) is implied if you specify YES for any one of the COLOR, HILIGHT, PROGSYMBOLS, QUERY, or VALIDATION (3270 only) attributes.

| If extended data stream (EXTENDEDDS) is set to YES, the device will support
| the write structured field COMMAND and Outbound Query structured field.

You can use the QUERY structured field to determine whether the device is set up to use the extended data stream (see page 333).

Using the QUERY structured field sets EXTENDEDDS to YES if query is valid.

FMHPARM({NO|YES})
specifies whether BMS is to accept user-supplied parameters for inclusion in the function management header built by BMS. Specify YES only if the DEVICE type is 3650.

FORMFEED({NO|YES})
If the devices for which you are defining this TYPETERM have the form-feed feature, you can specify FORMFEED(YES). If you do so, BMS uses the form-feed character when formatting output documents.

If DEVICE(SCSPRINT) is specified, BMS inserts a form-feed character at the beginning of the data stream. This causes the device to skip to the top margin of a new page before starting to print.

The top margin is defined by a set vertical format (SVF) data stream, and may be a line number equal to or greater than one. If a SVF data stream has not been sent to the printer, the top margin is line one. The line counter in the device is set to 1 when the operator sets up the paper.

Note that the device may also perform an automatic form feed if you try to print beyond a bottom margin. The bottom margin is also determined by the SVF data stream and defaults to the maximum presentation line (MPL). The MPL is the last line on the page and its value represents the page or form length as a number of lines (that is, physical page size times the line density). Both the MPL and the line density can be determined by the SVF data stream. Otherwise the MPL (the number of lines) can be set up on the device by the operator.

If DEVICE(3270), DEVICE(3270P), DEVICE(LUTYPE2), or DEVICE(LUTYPE3) is specified, use FORMFEED(YES) in conjunction with the FORMFEED option in the BMS SEND commands. Using form feed on display devices provides for a skip to a new page when the screen data is copied to a printer.

The options discussed above for SCSPRINT operation do not apply when the devices are operating as 3270P or LUTYPE3 devices. In this case there is only the concept of a form length, and this can be set on the device only by the operator. See the *CICS Application Programming Reference* manual for programming information on the use of the FORMFEED option.

GROUP(*groupname*)
Each resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see "How the CSD file is organized—groups and lists" on page 8.

The GROUP name can be up to eight characters in length. The characters allowed are A-Z 0-9 $ @ and #. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

HILIGHT({NO|YES})
specifies whether the 3270 device or SCS printer has the extended highlight facility, which enables fields or characters to be displayed in reverse-video, underline mode, or blink (3270 only).

You can use the QUERY structured field to determine whether the device is set up to use the extended highlight facility (see page 333).

HORIZFORM({NO|YES})
If the devices for which you are defining this TYPETERM have the horizontal

form feature, specify HORIZFORM(YES). The devices that can use this feature are batch, batch data interchange, interactive, SCSPRT or LUTYPE4 logical units.

NO The HTAB option in the BMS map definition is ignored.

YES BMS uses horizontal tabbing when formatting output documents.

IOAREALEN({0|value1},{0|value2})
specifies the length in bytes of a terminal input/output area to be passed to a transaction.

If you specify ATI(YES), you must specify an IOAREALEN of at least one byte.

(*value1, value2***)**
Value1 specifies the minimum size of a terminal input/output area to be passed to an application program when a RECEIVE command is issued.

If *value2* is not specified, or is less than *value1*, it defaults to the value of *value1*.

You can specify *value2* as greater than or equal to *value1*. In this case, when the size of an input message exceeds *value1*, CICS uses a terminal input/output area *value2* bytes long. If the input message size also exceeds *value2*, the node abnormal condition program sends an exception response to the terminal.

The maximum value that may be specified for IOAREALEN is 32767 bytes.

KATAKANA({NO|YES})
specifies whether Katakana support is required. Katakana terminals cannot display mixed case output; uppercase characters appear as uppercase English characters, but lowercase characters appear as Katakana characters. If you have any Katakana terminals connected to your CICS system, specify the MSGCASE=UPPER system initialization parameter. For further information about the MSGCASE parameter, see the *CICS System Definition Guide*.

NO Katakana support is not required.

YES Katakana support is required. All lowercase characters sent to the terminal from the following transactions are translated to uppercase:

CBRC CDBC CDBI CEBR CECI CEDA CEDF CEMT CEOT CESN CEST CMSG CRTE CSPG CWTO

LDCLIST(*listname***)**
specifies the name of a logical device code (LDC) list. The name may be up to eight characters in length. The name follows assembler language rules. It must start with an alphabetic character. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command. Define the LDCLIST and its contents using macroinstruction(s).

A local LDC list is defined by:

```
listname  DFHTCT TYPE=LDCLIST,
LDC(aa=nnn,bb=nnn,....)
```

An extended local LDC list is defined by:

```
listname  DFHTCT TYPE=LDC,LOCAL=INITIAL
          DFHTCT TYPE=LDC=(aa=nnn)....
          DFHTCT TYPE=LDC=(bb=nnn)....
          DFHTCT TYPE=LDC,LOCAL=FINAL
```

You specify this *listname* as the value for the LDCLIST attribute on the TYPETERM definition.

This attribute applies only to 3600, 3770 batch, 3770, and 3790 batch data interchange, and LUTYPE4 logical units. The list specifies which LDCs are valid for this logical unit and, optionally, which device characteristics are valid for each LDC. CICS uses the first LDC generated in this list when choosing a default LDC for a logical unit. For further guidance, see "DFHTCT logical device codes: VTAM non-3270" on page 444.

LIGHTPEN({NO|YES})
specifies whether a 3270 display has the selector pen feature.

LOGMODE(blank|0|*name*)
specifies how CICS is to build the BIND to be sent to the logical unit.

blank A defined terminal definition uses the BIND image generated by the CICS definitions for this device by means of this TYPETERM definition and its associated TERMINAL definitions. An autoinstalled terminal uses the fields specified in the incoming CINIT.

name This is the LOGMODE name from a VTAM logon mode table that has been set up for use by this logical unit. The name may be up to eight characters in length and must follow assembler language rules. The name must start with an alphabetic character. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command. This allows you to override the BIND image provided by CICS for the logical unit. For further information, see the appropriate CICS subsystem guide.

You cannot code a LOGMODE name whenthe terminal is a
cross-domain resource.

The TCTTE is updated to reflect the logmode bind image fields. These include SEND and RECEIVE sizes and default and alternate screen sizes. If the logmode indicates that the terminal is not queriable, the extended data stream fields are all set to zero.

0 (zero)
This causes CICS to use some of the information from the BIND image contained in the CINIT coming from the logical unit. The BIND image in the CINIT was created by VTAM based on the LOGMODE entry defined for the logical unit requesting to log on to CICS. The node initialization block (NIB) is built with LOGMODE=0 and BNDAREA=0. When the TYPETERM's SENDSIZE and RECEIVESIZE have been specified as zero, CICS replaces them with the values from the LOGMODE's RUSIZES.

The TCTTE is updated to reflect the incoming CINIT fields. These include SEND and RECEIVE sizes and default and alternate screen sizes. If the logmode indicates that the terminal is not queriable, the extended data stream fields are all set to 0. Use LOGMODE(0) only in exceptional circumstances. Although the LU is bound with the VTAM definition, CICS keeps the main session characteristics from the CICS definition. For example, if a printer is defined to VTAM as LUTYPE1 but to CICS as an LUTYPE3 with LOGMODE(0), CICS accepts the bind but sends LUTYPE3 control characters to the printer, giving rise to incorrect results. This restriction does not apply to pipeline terminals.

Notes:

1. You should only need to use this value for the logmode attribute in exceptional circumstances.
2. For a logical unit in a cross-domain environment, specify LOGMODE(0) and provide the logical unit mode information in the DLOGMOD and MODETAB operands of the VTAM(LU) statement (see the *CICS Family: Communicating from CICS on System/390* manual. for more information about the VTAM(LU) statement). (In a cross-domain environment, LOGMODE with a name causes a VTAM error).

LOGMODECOM(*no* | *yes***)**
The LOGMODECOM attribute is not valid in CICS Transaction Server for OS/390. See “Appendix A. Obsolete attributes retained for compatibility” on page 497 for information.

LOGONMSG({NO | YES})
specifies whether the ‘good morning’ transaction, specified in the GMTRAN system initialization parameter, will be:

- Automatically initiated when the logical unit is first logged on to CICS through VTAM
- Initiated after the terminal user’s TIMEOUT period has expired under certain conditions.

If you have specified ERRLASTLINE(YES), the messages written by the transaction do not overwrite the error message line.

NO CICS does not run the ‘good morning’ transaction.

Note: If you are using a non-SNA terminal such as a Telnet 3270, LOGONMSG(NO) does not automatically release the keyboard lock. You need to press the Reset key to release the keyboard lock.

YES CICS runs the ‘good morning’ transaction when the OPNDST exit is successfully completed and a session is established. The transaction is initiated by automatic task initiation (ATI) and competes with other ATI transactions for use of the terminal. Specify ATI(YES) for this TYPETERM.

Note: If you are using a non-SNA terminal such as a Telnet 3270, LOGONMSG(YES) also automatically releases the keyboard lock.

MSRCONTROL({NO | YES})
specifies whether the terminal, an 8775 or 3643, has a magnetic slot reader. This option is not valid for SCS printers.

You can use the QUERY structured field to determine whether the device is set up to use a magnetic slot reader (see page 333).

NEPCLASS({0 | *value***})**
specifies the node error program transaction class.

0 This results in a link to the default node error program module.

value The transaction class for the (nondefault) node error program module.

Value can be in the range 1 through 255. For programming information about the node error program, see the *CICS Customization Guide*.

OBFORMAT({NO|YES})
specifies whether outboard formatting is used. If the devices for which you are defining this TYPETERM use BMS outboard formatting, specify OBFORMAT(YES). OBFORMAT(YES) can be specified for two DEVICE types only:

- 3650, SESSIONTYPE(3270)
- LUTYPE2, for an 8100 Information System using the DPPX operating system with DPPX/DPS Version 2 for presentation services

For further guidance, see the *CICS/OS/VS IBM 3650/3680 Guide*, the *CICS/OS/VS IBM 3790/3730/8100 Guide*, and the *DPPX/Distributed Presentation Services Version 2: System Programming Guide*.

Use the QUERY structured field to determine whether the device is set up to use outboard formatting (see page 333).

OBOPERID({NO|YES})
specifies whether CICS uses the outboard operator identifiers to support the BMS routing facilities required for this terminal. This option applies only to the 3790 and 3770 batch data interchange logical units.

OUTLINE({NO|YES})
specifies whether the device supports field outlining.

Use the QUERY structured field to determine whether the device is set up to use field outlining (see page 333).

PAGESIZE({*rows*|*columns*})
specifies the default page size for this printer. The default page size is used by BMS when the default screen size has been selected in the DEFSCREEN attribute. (See page 347.)

rows Indicates the number of lines in the page. The PAGESIZE *rows* value can usefully be less than the DEFSCREEN *rows* value, perhaps to reserve the bottom line of a screen for error messages (see the ERRLASTLINE attribute), if the same BMS map is being used for both printing and display.

columns
Indicates the number of characters in each line. Unexpected results occur if the *columns* value specified in PAGESIZE differs from the *columns* value specified in DEFSCREEN.

The product of *lines* and *columns* must not exceed 32767.

The default value depends on the value you specify for the DEVICE attribute. See Table 17 on page 335 for details.

BMS uses the page size values when preparing output data streams. The specified number of characters in each line of the page should not exceed the physical line width of the terminal. In the case of printers that automatically perform a new-line function on reaching the end of the carriage (for example, 3270 printers), the line width specified here should be less than the physical line width.

This ensures that the formatting of the output data is governed entirely by the new-line (NL) characters supplied by BMS or by you, not by new-line functions performed by the device itself, which would produce additional lines of output, resulting in a physical page depth greater than that specified here.

For 3270-type printers, the hardware limits the amount of data that BMS may transmit. If the map or application program request specifies L40, L64, or L80, or does not specify NLEOM on the SEND MAP command, the product of *lines* and *columns* specified in PAGESIZE must not be greater than the buffer size.

If the BMS request specifies NLEOM, the page length may be any number of lines, but the product of *lines* and *columns* specified in the DEFSCREEN or ALTSCREEN attributes must not exceed the buffer size of the device.

In other words, the number of characters that BMS transmits must not exceed the physical buffer size of the printer.

Note: BMS divides a large page into smaller segments for transmission. PAGESIZE should therefore correspond to the required **logical** page size (the product of *lines* and *columns*), and the DEFSCREEN value should correspond to the actual buffer size.

For a VTAM 3600, the PAGESIZE specified is used if a BMS page build operation is attempted without specifying a logical device code (LDC). A default device type of 3604 is assumed.

For 3770, LUTYPE4, or 3790 batch data interchange logical units, the PAGESIZE specified is used if a BMS page build operation is requested without specifying a logical device code (LDC). The default device type is the console printer.

Take care when routing a message to a list of terminals. If the PAGESIZE you have defined (or allowed to default) is too small to accommodate the message, the transaction abends.

| For cumulative text processing, the maximum allowed buffer size is 32767. If
| this is exceeded, BMS internally forces a reduced page length to ensure that the
| PAGESIZE stays within the limit.

PARTITIONS({NO|YES})
specifies whether a device is to use partitions. This option is not valid for SCS printers.

You can use the QUERY structured field to determine whether the device is set up to use partitions (see page 333).

PRINTADAPTER({NO|YES})
For the 3275: specifies whether the printer adapter feature and corresponding 3284 Printer Model 3 are present on the 3275 Display Station. This feature makes the 3284 eligible for print requests through the PA key from the host 3275.

For LUTYPE2 logical units: specifies whether, for print requests initiated by the PRINT key or by an ISSUE PRINT command, printer allocation is handled by the 3790, or by the 3274 or 3276, according to the printer authorization matrix for both VTAM and non-VTAM attachments. Further, 3270 printers attached to the same 3790 are available for print requests sent to the

3270-display logical unit by a terminal control print request or initiated by the operator. If PRINTADAPTER is NO, printer allocation is determined by the PRINTER and ALTPRINTER attributes of the TERMINAL definition.

If output is created on the screen by BMS requests with the PRINT option, by BMS requests with the NLEOM option, or by the CMSG command, the contents of the screen are automatically copied to a 3270 printer, whether or not the CICS-defined PRINT key (usually a PA key) was pressed.

PROGSYMBOLS({NO|YES})
specifies whether the programmed symbol (PS) facility can be used on this 3270 device or SCS printer. The facility enables up to six 191-character sets, with customer-defined and program-loaded fonts and codes, to be stored and accessed.

You can use the QUERY structured field to determine whether the device is set up to use programmed symbols (see page 333).

QUERY({NO|COLD|ALL})
specifies whether CICS should use the QUERY structured field to determine the characteristics of the device. For more information about this function, and a list of the attributes which can be determined by using it, see page 333.

NO CICS does not use the QUERY function.

COLD CICS uses the QUERY function to determine the characteristics of the device only when the device is first connected after an initial or a cold start of CICS. The device characteristics are stored in the CICS global catalog for use on subsequent warm and emergency starts.

ALL CICS uses the QUERY function to determine the characteristics of the device each time the device is connected.

RECEIVESIZE(*value*)
For a **defined nonautoinstalled terminal**, specify the maximum size of a request unit that can satisfy a VTAM RECEIVE request. The RECEIVESIZE value is transmitted to the connected logical unit, and must be in the range 0 through 30720. It may be rounded down by CICS, because it must be transmitted in an architected form.

The effect of RECEIVESIZE depends on whether a RECEIVE RUSIZE is present in the VTAM LOGMODE table. Table 19 shows the RECEIVE RUSIZE used to bind a session for each possible combination of TYPETERM and LOGMODE values.

Table 19. RECEIVE RUSIZE for defined (nonautoinstalled) terminals

RECEIVE RUSIZE (VTAM)	TYPETERM RECEIVESIZE	RUSIZE used in bind
0	0	0
0	specified	TYPETERM RECEIVESIZE size
specified	0	0
specified	specified	TYPETERM RECEIVESIZE size

Table 19. RECEIVE RUSIZE for defined (nonautoinstalled) terminals (continued)

RECEIVE RUSIZE (VTAM)	TYPETERM RECEIVESIZE	RUSIZE used in bind
Note: The exception to this table is LOGMODE(0). If you specify this in your TYPETERM definition, VTAM values are used, irrespective of what else is specified. If LOGMODE(0), LOGMODECOM(YES) and RECEIVESIZE(0) are all used, CICS uses a RECEIVESIZE of 0 where in earlier releases the CINIT RUSIZE was used. If this causes a problem. the TYPETERM RECEIVESIZE should be made the same as the LOGMODE RUSIZE.		

APPC terminal For an APPC (LUTYPE6.2) single session terminal, 256 would be a suitable value.

Autoinstalled terminal For an autoinstalled terminal, a nonzero value for RECEIVESIZE specifies either the maximum or actual RECEIVE RUSIZE value used in binding a session for a logical unit defined with this TYPETERM.

The effect of RECEIVESIZE depends on whether a RECEIVE RUSIZE is present in the VTAM LOGMODE table. Table 20 shows the RECEIVE RUSIZE used to bind a session for each possible combination of TYPETERM and LOGMODE values.

Table 20. RECEIVE RUSIZE for autoinstalled terminals

RECEIVE RUSIZE (VTAM)	TYPETERM RECEIVESIZE	RUSIZE used in bind
0	0 and BUILDCHAIN(YES)	256
0	0 and BUILDCHAIN(NO)	0
0	specified	TYPETERM RECEIVESIZE size
specified	0	VTAM RECEIVE RUSIZE size
specified less than or equal to TYPETERM RECEIVESIZE	specified	VTAM RECEIVE RUSIZE size
specified greater than TYPETERM RECEIVESIZE	specified	this combination is invalid and results in message DFHZC5963

RECOVNOTIFY({NONE|MESSAGE|TRANSACTION})

This option applies to the recovery of sessions for terminals in a CICS region running with either VTAM persistent sessions or with XRF. It is for use in situations where a terminal user may have to take action, such as sign on again, after a CICS restart. Use RECOVNOTIFY to specify how such a user should be notified.

VTAM persistent sessions: In a CICS region running with persistent session support, this specifies how a terminal end user is notified that their terminal session has been recovered.

XRF: In a CICS region running with XRF support, this specifies how the terminal user is notified that an XRF takeover has occurred.

This option is not applicable to APPC sessions.

NONE
There is no notification that a takeover has occurred.

MESSAGE
A message is displayed on the screen to say that the system has recovered. The message is specified in two BMS maps; DFHXRC1 and DFHXRC2 for XRF; and DFHXRC3 and DFHXRC4 for VTAM persistent sessions. These maps are in map set DFHXMSG. If reduced takeover time is important, use MESSAGE rather than TRANSACTION.

The terminal must be defined with the ATI(YES) option, and must be capable of displaying a BMS map.

TRANSACTION
A transaction is initiated at the terminal. The name of the transaction is specified by the RMTRAN system initialization parameter. (The default transaction for RMTRAN is the one specified in the GMTRAN system initialization parameter: the good-morning transaction.)

For the TRANSACTION option, the terminal must be defined with the ATI(YES) option. If reduced takeover time is important, use MESSAGE rather than TRANSACTION.

RECOVOPTION({SYSDEFAULT|CLEARCONV|RELEASESESS|UNCONDREL|NONE})
This option applies to the recovery of sessions in a CICS region running with VTAM persistent sessions, or with XRF.

VTAM persistent sessions: In a CICS region running with persistent session support, this option specifies how you want CICS to recover the session, and return the terminal to service on system restart within the persistent session delay interval.

XRF: In a CICS region running with XRF support, this option specifies how you want CICS to recover the session, and return the terminal to service after an XRF takeover.

For all recovery options other than NONE, if the action taken is a VTAM UNBIND, the UNBIND is followed by a VTAM SIMLOGON.

SYSDEFAULT
VTAM persistent sessions: In a CICS region running with persistent session support, this specifies that CICS is to select the optimum procedure to recover a session on system restart within the persistent session delay interval, depending on the session activity and on the characteristics of the terminal.

Although sessions are recovered, any transactions in-flight at the time of the failure are abended and not recovered. Transactions are also abended if the recovered session is being used by another CICS region over an APPC connection.

CICS recovers the session with the least possible impact, in one of the following ways:

- If the terminal was not executing a transaction at the time of the CICS failure, no recovery action is required, and CICS takes the appropriate recovery notification action as defined by RECOVNOTIFY.

- If the terminal was busy (that is, executing a transaction) when CICS failed, CICS first tries to recover the session by sending a VTAM end-bracket indicator. If the end-bracket does not recover the session (for example, CICS may be in RECEIVE mode), CICS issues a CLEAR command. If the terminal does not support the CLEAR command, the recovery action taken is a VTAM UNBIND followed by a SIMLOGON.

  See the *CICS Recovery and Restart Guide* for more information about persistent sessions.

XRF: In a CICS region running with XRF support, this specifies that CICS is to select the optimum procedure to recover a busy session at takeover, depending on the session activity and on the characteristics of the terminal.

CLEARCONV

Prevents CICS from sending an end-bracket indicator to close an in-bracket session. Instead CICS sends a CLEAR request, to reset the conversation states. If the session does not support the CLEAR request, CICS sends an UNBIND request. The CLEAR or UNBIND is sent only if the session was busy at the time of system restart (in the case of persistent sessions) or takeover (in the case of XRF).

RELEASESESS

Requires CICS to send an UNBIND request to release the active session. The UNBIND is sent only if the session was busy at the time of system restart (in the case of persistent sessions), or takeover (in the case of XRF). Following the UNBIND, the session is queued for SIMLOGON. If the session is not busy, the requested recovery notification is carried out.

UNCONDREL

Requires CICS to send an UNBIND request to release the active session. The UNBIND is sent whether or not the session was busy at the time of system restart (in the case of persistent session support) or the takeover (in the case of XRF). Following the UNBIND, the session is queued for SIMLOGON.

NONE

VTAM persistent sessions: In a CICS region running with persistent session support, this specifies that the terminal session is not to be recovered at system restart within the persistent session delay interval: in effect, the terminal has no persistent session support. LU6.2 sessions are unbound, but the latest negotiated CNOS value is returned to the CICS system after the restart. After system restart, the terminal is reconnected automatically if you specify AUTOCONNECT(YES), subject to the operation of the AIRDELAY system initialization parameter (AIRDELAY=0 overrides AUTOCONNECT(YES), and the terminal is not reconnected).

| **RECOVOPTION(NONE)** should be used if this terminal or autoinstall model is to be used with persistent sessions (PSDINT = nnn in the SIT) but the terminal may be the subject of an EXEC CICS ISSUE PASS LUNAME() LOGONLOGMODE.

XRF: In a CICS region running with XRF support, this specifies that the logon state is not tracked by the alternate system, and the terminal session is not automatically recovered after a takeover: in effect, the terminal has no XRF support. After takeover, the terminal is reconnected automatically by the alternate system, if you specify AUTOCONNECT(YES).

RELREQ=({NO|YES})
specifies whether CICS is to release the logical unit.

NO CICS is not to release the logical unit upon request by another VTAM application program.

YES CICS is to release the logical unit, if the logical unit is not currently part of a transaction.

ROUTEDMSGS({ALL|NONE|SPECIFIC})
specifies which messages are to be routed to this terminal by an EXEC CICS ROUTE command. The default depends on the value you specify for the DEVICE attribute. See Table 17 on page 335 for details.

ALL BMS routes to this terminal messages that are destined for **all** terminals as well as those specifically destined for **this** terminal.

NONE
BMS does not route any messages to this terminal, whether they are destined for all terminals or for this terminal specifically.

SPECIFIC
BMS routes messages to this terminal when they are destined specifically for this terminal, but not when they are destined for **all** terminals.

SENDSIZE({0|value})
Defined terminal (nonautoinstalled): For a nonautoinstalled terminal, this is the maximum size in bytes of a request unit that can satisfy a VTAM RECEIVE request. The SENDSIZE value is transmitted to the connected logical unit, and must be in the range 0 through 30720. It may be rounded down by CICS, because it must be transmitted in an architected form.

The effect of SENDSIZE depends on whether a RECEIVE RUSIZE is present in the VTAM LOGMODE table. Table 21 shows the RECEIVE RUSIZE used to bind a session for each possible combination of TYPETERM and LOGMODE values.

Table 21. SEND RUSIZE for defined (nonautoinstalled) terminals

SEND RUSIZE (VTAM)	TYPETERM SENDSIZE	RUSIZE used in bind
0	0	0
0	specified	TYPETERM SENDSIZE size
specified	0	This combination is invalid and results in a bind failure with message DFHZC2403
specified	specified	TYPETERM SENDSIZE size
Note: The exception to this table is LOGMODE(0). If you specify this in your TYPETERM definition, VTAM values are used, irrespective of what else is specified.		

APPC terminal: For an APPC (LUTYPE6.2) single session terminal, 256 is a suitable value.

Autoinstalled terminal: For an autoinstalled terminal, a nonzero value for SENDSIZE specifies either the maximum or actual SEND RUSIZE value used in binding a session for a logical unit defined with this TYPETERM.

The effect of SENDSIZE depends on whether a SEND RUSIZE is present in the VTAM LOGMODE table. Table 22 shows the SEND RUSIZE used to bind a session for each possible combination of TYPETERM and LOGMODE values.

Table 22. SEND RUSIZE for autoinstalled terminals

SEND RUSIZE (VTAM)	TYPETERM SENDSIZE	RUSIZE used in bind
0	0	0
0	specified	TYPETERM SENDSIZE size
specified	0	VTAM SEND RUSIZE size
specified less than or equal to TYPETERM SENDSIZE	specified	VTAM SEND RUSIZE size
specified greater than TYPETERM SENDSIZE	specified	this combination is invalid and results in message DFHZC5963

SESSIONTYPE(*type*)
specifies the type of session that can be used for a VTAM SNA logical unit. For details, see Table 17 on page 335.

SHIPPABLE({NO | YES})
specifies whether the definition is allowed to be sent to a remote system if this device tries to initiate a remote transaction. This function may be used for any terminal, whether autoinstalled or with its own TERMINAL definition. The shipping does not work unless the terminal has a definition installed, by one of these methods, in the local system.

Using SHIPPABLE(YES) means that you do not need to ensure that a definition of the terminal exists on the remote system for a locally defined terminal to initiate a transaction in that system. This can be useful when the remote system cannot share the CSD file with the local system.

A definition for the terminal must already be installed in (or already shipped to) the remote system.

For guidance on deciding whether to use SHIPPABLE(YES), see "Terminals for transaction routing" on page 291.

NO This definition cannot be shipped to a remote system.

YES This definition can be shipped to a remote system.

SIGNOFF({YES | NO | LOGOFF})
specifies the actions taken when GNTRAN (CESF or user-defined transaction) is attached and attempts to sign off the terminal. If you are using RACF 1.9 or later, specify the TIMEOUT limit in the RACF segment.

YES When the specified time has elapsed after the last input from the operator, the terminal is automatically signed off from CICS.

NO The terminal is not timed out.

LOGOFF
When the specified time has elapsed after the last input from the operator, the terminal is automatically signed off from CICS and then

logged off from VTAM. (This is not supported for TCAM-attached terminals.) LOGOFF is useful for an autoinstall model, because virtual storage is not wasted on entries for terminals that have been timed out.

If GNTRAN fails to attach because of unprocessed data in the terminal buffer (resulting in a BID failure), the terminal will be signed off and logged off. GNTRAN will not run and will have no effect.

Note: You cannot change the value of this attribute when DEVICE(APPC) is specified. The default value in that case is SIGNOFF(NO).

SOSI({NO|YES})
specifies whether the device supports mixed EBCDIC and double-byte character set (DBCS) fields.

You can use the QUERY structured field to determine whether the device is set up to use mixed EBCDIC and DBCS fields (see page 333).

TERMMODEL({1|2})
specifies the model number of the terminal. If the device is a component of the 3270 Information Display System, this operand must be specified as follows:

1 Specify 1 for the 3270 Model 1 displays and printers (for example, 3277 Model 1) with a default screen or buffer size of 12x40 (480 bytes/characters). TERMMODEL(1) is the default for 3270 Model 1 printers and displays.

Specify 1 for the 3275 Display Station Model 11. The CICS support obtained is identical to that obtained by coding TERMMODEL(1) for 3275 Display Station Model 1.

2 Specify 2 for the 3270 displays and printers (for example, 3278 Model 4) with a default screen or buffer size of 24x80 (1920 bytes/characters). TERMMODEL(2) is the default for the 3286 printer in 3270 compatibility mode.

Specify 2 for the 3275 Display Station Model 12. The CICS support obtained is identical to that obtained by coding TERMMODEL(2) for 3275 Display Station Model 2.

TEXTKYBD({NO|YES})
specifies whether the 3270 device has the text-keyboard feature.

TEXTPRINT({NO|YES})
specifies whether the 3288 printer has the text-print feature.

TTI({YES|NO})
specifies whether transactions can be initiated at the terminal by a user.

YES Transactions can be initiated at the terminal by a user. If you also specify ATI(YES), transactions can also be initiated automatically. In this case, the automatic transaction initiation, either by transient data control or interval control, sets a condition in an appropriate terminal control table terminal entry. If both ATI and TTI are specified as YES, and if there is no transaction at the terminal, terminal control initiates the user-defined task. This task is expected to send messages to the terminal.

For a terminal used in the processing of transactions such as inquiries or order entries, specify TTI(YES) and ATI(NO). This also applies to a display station or hard-copy terminal to which no messages are sent without a terminal request and through which transactions are entered. Note that this is the only specification allowed for 3790 inquiry logical units.

NO Transactions cannot be initiated at the terminal by a user. If you specify NO, specify ATI(YES) to allow transactions to be initiated automatically. An example of this type of terminal is one that is located in a remote location, such as a warehouse, and is unattended but may receive messages.

TYPETERM(*name*)
specifies the name of this extension of a TERMINAL definition. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase except when using the CREATE command.

This name is referred to in all the TERMINAL definitions using this TYPETERM. Note that this TYPETERM definition must be installed before or at the same time as the TERMINAL definitions that reference it.

UCTRAN({NO | YES | TRANID})
specifies whether the input data stream from a terminal is to be translated to uppercase. The input data stream may include a transaction identifier as well as program data. CICS supports transaction identifier definition in mixed case, and the UCTRAN attribute can be used to ensure that the correct transaction is
# located. Uppercase translation is done for both 3270 and non-3270 data
# streams.

NO No uppercase translation is performed.

YES All the data input from the terminal, both the transaction identifier if present and the program data, is translated to uppercase before any processing.

TRANID
When the input data stream includes a transaction identifier, CICS translates it to uppercase before attempting to locate its definition. However, all the input data, both the transaction identifier and the program data, is passed to the program without any translation.

Therefore both the YES and the TRANID options allow transaction identifiers to be defined in uppercase and to be entered from the terminal in either uppercase or lowercase, but the TRANID option causes the transaction identifier and program data to be passed to the program without any translation.

You can also request translation to uppercase at the transaction level on PROFILE definitions (see page 240), but be aware that a TYPETERM UCTRAN(YES) definition overrides a PROFILE UCTRAN(NO) definition. So, if you specify TYPETERM UCTRAN(YES), a PROFILE UCTRAN(NO) has no effect. Translation can be overridden by the application program for all RECEIVE requests except the first, by using the ASIS option.

Table 23 on page 364 shows which portion of the terminal input is translated (transaction id and/or data) according to the setting of the UCTRAN on the PROFILE and TYPETERM resource definitions.

Table 23. The effect of UCTRAN attributes on tranid and data translation

Profile (PROFILE)	Terminal (TYPETERM)		
	UCTRAN (YES)	UCTRAN (NO)	UCTRAN (TRANID)
UCTRAN (YES)	Tranid: Yes Data: yes	Tranid: No Data: Yes	Tranid: Yes Data: Yes
UCTRAN (NO)	Tranid: Yes Data: Yes	Tranid: No Data: No	Tranid: Yes Data: No

USERAREALEN({0 | *number*})
specifies the length in bytes (0 to 255) of the user area for this terminal. It should be made as small as possible. The TCT user area is initialized to zeros at system initialization.

The TCT user area may be located above or below the 16Mb line in virtual storage. Where it is located depends on the value of the TCTUALOC system initialization parameter. Specify this so that it caters for any programs you may have that are not capable of handling 31-bit addressing.

VALIDATION({NO | YES})
For the 8775specifies whether the 8775 device has the extended validation feature, which allows fields to be defined as TRIGGER, MANDATORY FILL, or MANDATORY ENTER.

For the 3290specifies whether the 3290 device has the validation feature, which allows fields to be defined as MANDATORY FILL or MANDATORY ENTER.

This option is not valid for SCS printers. If VALIDATION(YES) is specified for an SCS printer, an error message is raised and the option is ignored.

You can use the QUERY structured field to determine whether the device is set up to use the validation feature (see page 333).

VERTICALFORM({NO | YES})
specifies whether the devices for which you are defining this TYPETERM have the vertical form feature. The devices that can use this feature are batch, batch data interchange, interactive, SCSPRT or LUTYPE4 logical units.

NO	The VTAB option in the BMS map definition is ignored.
YES	BMS uses vertical tabbing when formatting output documents.

XRFSIGNOFF({NOFORCE | FORCE})
specifies the sign-on characteristics of a group of terminals.

If you have a collection of terminals in a security-sensitive area, for example, you might choose to force sign-off of those terminals after a takeover, to prevent the use of the terminal in the absence of the authorized user. (This could happen if the authorized user left the terminal during takeover, and the terminal became active again while it was unattended.)

This option works in conjunction with the XRFSOFF system initialization parameter and the XRFSOFF entry in the CICS RACF segment (if you are running RACF 1.9). The relationship between them is explained on page 299.

Part 6. Macro resource definition

This part gives reference information for the macros needed for each table. The information is divided into a number of sections, one for each table, in alphabetic order as listed below.

Each section describes the function of the table and outlines the macros used to create it. The operands for each macro are given in a syntax box, which is followed by an explanation of each operand, in alphabetic order.

“TCT—terminal control table is further subdivided according to access method and type of device.

Chapter 13. Resource definition with macros

This chapter tells you which CICS system table defines each resource, and how to prepare the tables. You use macros to define:
- Non-VTAM networks
- Non-VTAM terminals
- Non-VSAM files
- Databases
- Queues
- Monitoring resources
- System recovery resources

CICS uses an external security manager for all its security management.

BTAM is not supported in this version of CICS. To gain access to CICS from BTAM terminals, define the BTAM terminals in a version of CICS earlier than 3.2.1, and use transaction routing to gain access to the current release. BTAM terminals must be defined as remote resources in this release, and as local resources in the earlier release. This book contains information about the definition of remote BTAM terminals: information about the definition of local BTAM terminals can be found in the documentation for your earlier release of CICS.

You must use **resource definition online (RDO)** for VSAM files, and to define programs, map sets, partition sets, transactions, and profiles. You must also use RDO to define VTAM terminals, and links and sessions with MRO (multiregion operation) and ISC (intersystem communication) systems. RDO is described in "Chapter 1. Resource definition—an introduction" on page 3.

Introduction to CICS control tables and macros

CICS is configured under your control during system initialization. You select a system initialization table (SIT) and, through it, CICS selects other control tables. Each control table is created separately and may be recreated at any time before system initialization. You prepare the required control tables by coding the appropriate macros. For each table, the macros automatically generate the necessary linkage editor control statements.

This chapter gives you an outline description of the control tables.

You may need to read the *CICS System Definition Guide* for further information about the following areas related to control tables:
- The **system initialization table (SIT)**, required for the system to be operational. Other tables are needed only if you are using the corresponding CICS facilities.
- The **job control language (JCL)** needed for the control tables, and how to link-edit and assemble the macro statements that you specify.
- Whether CICS loads a table above or below the 16MB boundary. The CICS table macros contain linkage editor control statements that determines this. Table 24 on page 368 shows whether specific tables are loaded above or below the boundary.

The control tables that can be defined by macros are shown in Table 24 on page 368.

Table 24. Control tables definable by macros. The third column shows whether the table is loaded above or below the 16MB line.

Control table	Contents	Above the line?	Page reference
Command list table (CLT)	Sets of commands and messages for an XRF takeover. The command list table (CLT) is used for XRF (extended recovery facility). If you are using XRF, you must have a CLT; it is used only by the alternate CICS system. The CLT contains a list of commands that are passed to JES or MVS for execution. It also provides the authorization for canceling the active CICS system.	Yes	372
Data conversion table	A data conversion table may be needed if the CICS system is using ISC to communicate with a member of the CICS family that runs on a hardware platform that does not use EBCDIC (such as CICS OS/2™ or CICS/6000 which use ASCII). The conversion table defines how data is to be changed from ASCII format at the workstation to EBCDIC format at the CICS/ESA host.		The DFHCNV macros used to create the table are described in the *CICS Family: Communicating from CICS on System/390* manual.
Destination control table (DCT)	Contains an entry for each extrapartition, intrapartition, indirect, and remote destination. Extrapartition entries address data sets external to the CICS region. Indirect destination entries redirect data to a destination controlled by another DCT entry. Intrapartition destination entries contain the information required to locate the queue in the intrapartition data set.	Yes	376
DL/I directories (PDIR)	Databases and program specification blocks. If you use CICS-IMS DBCTL (database control) exclusively to manage your CICS system's use of DL/I, you need not define the DL/I directory (PDIR) using CICS. The PDIR is a directory of all the remote program specification blocks (PSBs) that are accessed by the CICS system. If you function-ship requests to a remote database manager (remote DL/I), you need only one directory, the PDIR.	No	391
File control table (FCT)	BDAM file definitions. The file control table (FCT) is retained to allow you to define BDAM files.	Yes	393
Monitoring control table (MCT)	Monitoring actions (data collection) to be taken at each user event monitoring point (EMP). Different actions can be specified for each monitoring class at each EMP.	Yes	406

Table 24. Control tables definable by macros (continued). The third column shows whether the table is loaded above or below the 16MB line.

Control table	Contents	Above the line?	Page reference
Program list table (PLT)	A list of related programs. You may want to generate several PLTs to specify a list of programs that are to be executed in the initialization programs phase of CICS startup; executed during the first or second quiesce stages of controlled shutdown; or both, or enabled or disabled as a group by a CEMT ENABLE or DISABLE command.	Yes	417
Recoverable service table (RST)	Sets of recoverable service elements. The recoverable service table (RST) is used for IBM CICS IMS/ESA® DBCTL (database control) support. If you are using XRF and DBCTL, you must have an RST: it is used by the active CICS system. The RST contains a list of recoverable service elements that define the DBCTL configuration. It defines which DBCTL CICS connects to.	Yes	435
| Resource control table (RCT)	A CICS-DB2 connection. The Resource control table (RCT) is retained for migration purposes only.		421
System initialization table (SIT)	Parameters used by the system initialization process. In particular, the SIT identifies (by suffix characters) the versions of CICS system control programs and CICS tables that you have specified are to be loaded.		See the *CICS System Definition Guide*.
# System recovery table (SRT)	A list of codes for abends that CICS intercepts.		437.
Temporary storage table (TST)	Special processing for temporary storage. Application programs can store data in temporary storage for later retrieval. For the data to be recoverable by CICS if the system terminates abnormally, data identifiers have to be specified in the temporary storage table (TST). A generic data identifier can be coded so that any unique temporary storage identifier (generated dynamically in a program) that begins with the same characters as the generic identifier (in the TST) can automatically acquire the same properties as the TST entries. Resource security level checking can be done on the temporary storage queues.	Yes	484
Terminal control table (TCT)	Retained to define non-VTAM terminal networks.	No	439

Table 24. Control tables definable by macros (continued). The third column shows whether the table is loaded above or below the 16MB line.

Control table	Contents	Above the line?	Page reference
Terminal list table (TLT)	Sets of related terminals. The terminal list table (TLT) allows terminal or operator identifications, or both, to be grouped logically. A TLT is required by the supervisory terminal operation (CEST), to define and limit the effective range of the operation. It can also be used by a supervisory or master terminal operation (CEMT) to apply a function to a predetermined group of terminals. A TLT can be used, singly or in combination with other TLTs, to provide predefined destinations for message switching.	No	481
Transaction list table (XLT)	Sets of logically related transaction identifications. A list of identifications that can be initiated from terminals during the first quiesce stage of system termination, or a group of identifications that can be disabled or enabled through the master terminal.	Yes	491

TYPE=INITIAL (control section)

Most of the tables must start with a TYPE=INITIAL macro. For some tables you can provide information that applies to the whole table, on the TYPE=INITIAL macro.

The TYPE=INITIAL macro establishes the control section (CSECT) for the CICS system table, and produces the necessary linkage editor control statements. CICS automatically generates the address of the entry point of each table through the DFHVM macro that is generated from each TYPE=INITIAL macro. The entry point label of the table is DFHxxxBA. Only the END statement need be specified.

Naming and suffixing the tables

The tables are named as follows:

Table 25. Names of the control tables

Table	Name
Command list table	DFHCLTxx
Destination control table	DFHDCTxx
File control table	DFHFCTxx
Monitoring control table	DFHMCTxx
Program list table	DFHPLTxx
Recoverable service table	DFHRSTxx
Resource control table	DFHRCTxx
System recovery table	DFHSRTxx
Terminal control table	DFHTCTxx
Terminal list table	DFHTLTxx
Temporary storage table	DFHTSTxx
Transaction list table	DFHXLTxx

Table 25. Names of the control tables (continued)

Table	Name

The first six characters of the name of each table are fixed. You can specify the last two characters of the name, using the SUFFIX operand. The SUFFIX operand is specified on the TYPE=INITIAL macro for each table.

Suffixes allow you to have more than one version of a table. A suffix may consist
# of one or two characters. The acceptable characters are: A-Z 0-9 @. (Do not use **NO** or **DY**.) Select the version of the table to be loaded into the system during system initialization, by specifying the suffix in the appropriate system initialization parameter operand.

For example:

```
DFHSIT...,FCT=MY,...
```

Note: The TYPE=INITIAL macros have a STARTER operand that is not listed in the descriptions of the individual macros in the main body of this book.
Coding STARTER=YES enables you to use the $ and # characters in your table suffixes. The default is STARTER=NO. This operand should be used only with starter system modules.

TYPE=FINAL (end of table)

Most of the tables, again with the single exception of the SIT, must end with a TYPE=FINAL macro. The TYPE=FINAL macro creates a dummy entry to signal the end of the table. It must be the last statement before the assembler END statement. The format is always like this:

Table 26. TYPE=FINAL

&rbl;	DFHCLT	TYPE=FINAL

Format of macros

The CICS macros are written in assembler language and, like all assembler language instructions, are written in the following format:

Table 27. Format of macros

Name	Operation	Operand	Comments
Blank or symbol	DFHxxxxx	One or more operands separated by commas	

For information about the syntax, see the “Preface” on page ix.

CLT—command list table

The command list table (CLT) is used by the **extended recovery facility** (XRF). The CLT contains a list of MVS system commands and messages to the operator, to be issued during takeover. Typically, the function of these commands is to tell alternate systems to take over from their active systems in the same MRO-connected configuration.

It also contains the name of the alternate system, with the jobname of the active system that it is allowed to cancel. (See DFHCLT TYPE=LISTSTART FORALT operand.) This provides a security check against the wrong job being canceled, when the alternate system takes over.

In addition, the DFHCLT TYPE=INITIAL macro gives JES routing information, needed to send cancel commands to the appropriate MVS system. If you are using XRF, you **must** have a CLT: it is used only by the alternate CICS system.

For security reasons, link-edit the CLT into a library authorized using APF. For virtual storage constraint relief considerations, link-edit using a MODE control statement specifying AMODE(31),RMODE(ANY). The table should be link-edited as reentrant. The CLT is not loaded into the CICS nucleus.

Your CLT can contain the following statements:
- DFHCLT TYPE=INITIAL
- DFHCLT TYPE=LISTSTART
- DFHCLT TYPE=COMMAND
- DFHCLT TYPE=WTO
- DFHCLT TYPE=LISTEND
- DFHCLT TYPE=FINAL (see page 371)

Note: Although the CLT may be shared by a number of alternate systems, take care that MVS is not given too many redundant commands during takeover. For example, in multiregion operation and using one CLT with commands for several regions, region 1 would send valid commands to other regions, but they would in turn send redundant commands to region 1 and to each other.

Control section—DFHCLT TYPE=INITIAL

The DFHCLT TYPE=INITIAL macro establishes the entry point and the beginning address of the CLT being defined.

Table 28. DFHCLT TYPE=INITIAL

```
label        DFHCLT        TYPE=INITIAL
                           [,JES={JES2|JES3}]  |
                           [,JESCHAR=value]
                           [,JESID=((mvsname1,jesname1,
                                 spoolno1)
                              [,(mvsname2,jesname2,spoolno2),...])]
                           [,SUFFIX=xxx]
```

For general information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see “TYPE=INITIAL (control section)” on page 370.

JES={JES2 | JES3}
: Specifies the version of JES being used. If you have active and alternate CICS systems in different CPCs, you must use the same version of JES on both CPCs.

If you are using JES3, you need release 2.2.0 for full support of all CLT functions in a two-CPC environment. JES2 accepts commands issued by programs. For JES3, this feature was only introduced for release 2.2.0. With an earlier JES3 release, it is possible that a takeover in a two-CPC environment can only proceed after the operator has manually canceled the failing active CICS system. This should occur only when the active system does not realize that it is failing and continues to run.

JESCHAR=value
: Specifies the one-character prefix to be used for commands to be passed to JES. If you omit this keyword:

- JESCHAR=$ is the default for JES=JES2
- JESCHAR=* is the default for JES=JES3

JESID=((mvsname1,jesname1,spoolno1) [,(mvsname2,jesname2,spoolno2),...])
: Specifies the JES routing code that corresponds to the MVS name and JES name of an active CICS system. You must use this option if active and alternate CICS systems are in different CPCs.

mvsname1
: This is the SID, as specified in SYS1.PARMLIB member SMFPRM*xx*, for the CPC on which the active CICS system executes.

jesname1
: This is the JES2 or JES3 subsystem name for the JES under whose control the active system executes. It is defined in the MVS/ESA™ SCHEDULR sysgen macro and also in SYS1.PARMLIB member IEFSSN*xx*.

spoolno1
: For JES2, this is the multiaccess spool member number of the JES2 for the active CICS system. It is defined in the JES2 initialization parameter MASDEF SSID(*n*). For JES3, this is the processor name of the JES3 for the active CICS system. It is defined in the JES3 initialization parameter MAINPROC NAME=*name*. See the *MVS/ESA JES2 Initialization and Tuning* manual or the *MVS/ESA JES3 Initialization and Tuning* manual.

You may extend this, using more groups of *mvsname*, *jesname*, and *spoolno*, so that the CLT can be used to refer to more CPC/JES combinations.

Specifying alternate systems—DFHCLT TYPE=LISTSTART

This macro defines the start of the set of commands and messages that the alternate CICS issues when it takes over from the active CICS. (There may be no commands or messages, but you still need a CLT, so that authorization checks can be made.)

Table 29. DFHCLT TYPE=LISTSTART

```
label          DFHCLT            TYPE=LISTSTART
                                 ,FORALT=((applid1,jobname1)
                                    [,(applid2,jobname2),...])
```

FORALT=((applid1,jobname1)[,(applid2,jobname2),...])
: Specifies pairs of alternate and active CICS systems.

applid1
The name of the alternate CICS that issues the set of commands and messages when it takes over. This name must be the **specific APPLID**, defined in the APPLID system initialization parameter. It is used as an authorization check.

jobname1
The name of the active CICS system from which the alternate system is taking over. This name must be the **MVS JOBNAME** for the active CICS system. It is used as a security check, to ensure that the alternate system does not attempt to cancel any job other than one of that name.

You may extend this, using more pairs of *applid* and *jobname*, so that you can use one CLT for several alternate CICS systems.

Specifying takeover commands—DFHCLT TYPE=COMMAND

This macro allows you to specify the commands to be used by the alternate CICS system during takeover.

Table 30. DFHCLT TYPE=COMMAND

label	DFHCLT	TYPE=COMMAND
		,COMMAND=command-string

COMMAND=command-string
Defines a command that is passed to MVS for execution. CICS does not interpret this command.

The command that is issued in this way most frequently is `CEBT PERFORM TAKEOVER`.

In multiregion operation (MRO), where there is a simple hierarchy of **master** and **dependent** regions, a failing master region can issue this command to each of its dependent regions, if it is necessary that they also move to another CPC.

In a more complex multiregion operation, a failing master region can issue this to its **coordinator** region, and the coordinator can issue the same command to other master and dependent regions in the same hierarchy of regions. Hence, many MRO-connected regions can move together to another CPC, without operator intervention.

Here are some examples:

- A master region without a coordinator sends a command to a dependent region:

```
DFHCLT TYPE=COMMAND,
       COMMAND='MODIFY CICSDEP,CEBT PERFORM
       TAKEOVER'
```

- A master region sends a command to its coordinator region:

```
DFHCLT TYPE=COMMAND,
       COMMAND='MODIFY CICSCRD,CEBT PERFORM
       TAKEOVER'
```

- A coordinator region sends commands to master and dependent regions:

```
DFHCLT TYPE=COMMAND,
       COMMAND='MODIFY CICSMAS,CEBT PERFORM
       TAKEOVER'
DFHCLT TYPE=COMMAND,
       COMMAND='MODIFY CICSDEP,CEBT PERFORM
       TAKEOVER'
```

- You can also issue other commands to any other job running under MVS:

```
DFHCLT TYPE=COMMAND,
        COMMAND='MODIFY jobname,command
        string'
```

Messages to the operator—DFHCLT TYPE=WTO

These two instructions define a message that is written to the system operator.

Table 31. DFHCLT TYPE=WTO

label	DFHCLT	TYPE=WTO ,WTOL=addr
addr	WTO	'message to operator' [,ROUTCDE=(number)] [,DESC=(number)] ,MF=L

WTOL=addr
: Specifies the address of a list format WTO macro that defines the message and any associated route codes and descriptor codes.

The MF (macro format), ROUTCDE (routing code), and DESC (descriptor) operands of the WTO macro are described in the *OS/390 MVS Programming: Authorized Assembler Services Reference ALE-DYN* manual.

An example is to send a request to the operator:

```
          DFHCLT TYPE=WTO,
                WTOL=wtoad
wtoad     WTO   'switch local terminals, please',
                 MF=L
```

Closing the command list—DFHCLT TYPE=LISTEND

This instruction defines the end of the set of commands and messages issued by an alternate system when it takes over from an active system.

Table 32. DFHCLT TYPE=LISTEND

label	DFHCLT	TYPE=LISTEND

DCT—destination control table

Note: Although the DFHDCT macro is supported by CICS Transaction Server for OS/390, you are recommended to use RDO to define your transient data resources. Online definition of TDQUEUE resources supports attributes that are not supported by the DFHDCT macro; for example, the INDOUBT attributes WAIT and WAITACTION.

The DFHDCT TYPE=(INITIAL,MIGRATE) and DFHDCT TYPE=GROUP macros, and extensions to the DFHCSDUP MIGRATE command help you convert your existing DCT macro definitions to RDO definitions. See “RDO GROUP for migration—DFHDCT TYPE=GROUP” and “MIGRATE” on page 102 for more information.

The destination control table (DCT) contains an entry for each **transient data destination**. A destination can be intrapartition, extrapartition, indirect, or remote. You code different DFHDCT macros for each type. The macros specify the symbolic name for each destination, and other information that CICS needs.

CICS uses several destinations for its own purposes. These entries must be included in the generation of the DCT, if the associated functions are being used. See “Required entries in the destination control table” on page 387 for details of the required entries.

DFHDCT macro types

The DFHDCT TYPE=GROUP macro can be used for migrating DCTs to the CSD file. A TYPE=(INITIAL,MIGRATE) macro must be specified in this case, to obtain the correct AMODE and RMODE.

The following macros define transient data destinations:

- DFHDCT TYPE=INITIAL establishes the control section and necessary linkage editor control statements for the DCT.
- DFHDCT TYPE=SDSCI defines the data control block (DCB), for an extrapartition destination.
- DFHDCT TYPE=EXTRA defines an extrapartition destination: a destination that is outside the CICS region.
- DFHDCT TYPE=INDIRECT defines an indirect destination: a logical destination that points to another destination. (This allows several logical destinations to be merged into one physical destination.)
- DFHDCT TYPE=INTRA defines an intrapartition destination: a destination that is within the CICS region.
- DFHDCT TYPE=REMOTE defines a destination that is owned by another CICS system or region.
- DFHDCT TYPE=FINAL concludes the DCT (see page 371).

RDO GROUP for migration—DFHDCT TYPE=GROUP

The DFHDCT TYPE=GROUP macro can be used to migrate a group of transient data resources defined using the DFHDCT macro to the CSD file.

Table 33. DFHDCT TYPE=GROUP

DFHDCT	TYPE=GROUP
	[,GROUP=name]

GROUP=name
: The contents of a DCT is migrated in a group or several groups. Code the GROUP=*name* operand with a group name of up to eight characters in length. If you want your macro definitions to go into the default group, specify GROUP=*DEFAULT.

  For more information about migrating transient data macro definitions to RDO, see “MIGRATE” on page 102.

Control section—DFHDCT TYPE=INITIAL

The DFHDCT TYPE=INITIAL macro establishes the entry point and beginning address for the DCT being defined.

Table 34. DFHDCT TYPE=INITIAL

DFHDCT	TYPE=(INITIAL[,MIGRATE]) [,SUFFIX=xx] [,USERID=name]

For general information about TYPE=INITIAL macros, see “TYPE=INITIAL (control section)” on page 370.

MIGRATE
: specify MIGRATE when assembling your DCTs for migration to the CSD file.

SUFFIX=xx
: Two characters that are concatenated with `DFHDCT` to create the name of the destination control table.

USERID=name
: Code this with the userid that you want CICS to use for security checking for any trigger-level TYPE=INTRA entry that does not specify its own userid.

  If you omit the userid from a trigger-level entry, and also from this TYPE=INITIAL macro, CICS uses the CICS default userid, specified on the DFLTUSER system initialization parameter.

  See the USERID description for the TYPE=INTRA macro for information about surrogate user checking.

Data set control information—DFHDCT TYPE=SDSCI

This macro defines the data control block (DCB) for an extrapartition transient data destination.

You must also code a DFHDCT TYPE=EXTRA macro, to define the destination.

Extrapartition data sets can be blocked or unblocked, of fixed or variable length.

Table 35. DFHDCT TYPE=SDSCI

DFHDCT	TYPE=SDSCI ,DSCNAME=name [,BLKSIZE=length] [,BUFNO={<u>1</u>|number}] [,ERROPT={<u>IGNORE</u>|SKIP}]

Table 35. DFHDCT TYPE=SDSCI (continued)

```
[,RECFORM={FIXUNB|FIXUNBA|
    FIXUNBM|FIXBLK|FIXBLKA|
    FIXBLKM|VARBLK|VARBLKA|
    VARBLKM|VARUNB|VARUNBA|
    VARUNBM}]
[,RECSIZE=length]
[,REWIND={LEAVE|REREAD}]
[,TYPEFLE={INPUT|OUTPUT|
    RDBACK}]
```

TYPE=SDSCI
Indicates that this DCT entry contains data set control information.

BLKSIZE=length
Code this with the length of the block, in bytes.

For V format data sets, each block consists of a block descriptor word followed by one or more logical records. The value coded for BLKSIZE must include 4 bytes for the block descriptor word, and also make allowance for the largest possible logical record (which itself includes 4 bytes of record descriptor word).

If the data set already exists, BLKSIZE can be omitted. However, if BLKSIZE is coded for an input data set it should match the data set BLKSIZE.

BUFNO={1|number}
Code this with the number of buffers to be provided. Any number up to 255 may be coded.

DSCNAME=name
Code this with the 1-to 8-character data set control name. This name must be the same as that coded in the DSCNAME operand of any associated DFHDCT TYPE=EXTRA macro.

The name used for DSCNAME must be used as the ddname on the DD statement, and is also used as the name for the DCB that is created.

The name must not start with the letters “DFH”, which are reserved for use by CICS, unless it is describing one of the standard destinations listed under “Required entries in the destination control table” on page 387. Use of the prefix “DFH” may cause assembly errors and future compatibility problems, because the DSCNAME parameter becomes an externally-specified name.

ERROPT={IGNORE|SKIP}
Code this with the error option to be performed if an I/O error occurs.

IGNORE
The block that caused the error is accepted.

SKIP
The block that caused the error is skipped.

**RECFORM={FIXUNB|FIXUNBA|FIXUNBM|FIXBLK|
FIXBLKA|FIXBLKM|VARBLK|VARBLKA|**

VARBLKM|VARUNB|VARUNBA|VARUNBM}
Code this with the record format of the data set.

FIXUNB
Fixed unblocked records

FIXBLK
Fixed blocked records

VARBLK
Variable blocked records

VARUNB
Variable unblocked records

FIXUNBA,FIXBLKA,VARBLKA,VARUNBA
Fixed unblocked, fixed blocked, variable blocked, or variable unblocked records respectively, **together with** ASA control characters

FIXUNBM,FIXBLKM,VARBLKM,VARUNBM
Fixed unblocked, fixed blocked, variable blocked, or variable unblocked records respectively, **together with** machine control characters.

RECSIZE=length
Code this with the length of the record, in bytes.

For V format data sets, each logical record consists of a record descriptor word followed by a data record. The value coded for RECSIZE must include 4 bytes for the record descriptor word (or LLBB), and also make allowance for the largest possible data record.

RECSIZE=*length* need be coded only for RECFM=FIXBLK.

REWIND={LEAVE|REREAD}
Code this with the disposition of a tape data set.

LEAVE
The current tape is positioned to the logical end of the data set.

REREAD
The current tape is positioned to reprocess the data set.

TYPEFLE={INPUT|OUTPUT|RDBACK}
Code this with the type of data set.

INPUT
An input data set

OUTPUT
An output data set

RDBACK
An input data set that is to be read backward.

Note: This is appropriate only when the data set has been defined on magnetic tape.

An extrapartition SDSCI can be either input or output, but not both.

For more information on the above operands, see the *MVS/ESA Data Administration: Macro Instruction Reference* manual.

Extrapartition destinations—DFHDCT TYPE=EXTRA

Destinations outside the CICS region (but which are allocated to CICS) are specified in the DFHDCT TYPE=EXTRA macro. This macro must be generated once for every extrapartition destination.

Extrapartition destinations are used for:

- Sending data outside the CICS region; for example, data created by a transaction, for processing by a batch program.
- Retrieving data from outside the region; for example, data received from terminals, as input to a transaction.

Extrapartition data is sequential and is managed by QSAM.

Table 36. DFHDCT TYPE=EXTRA

```
DFHDCT           TYPE=EXTRA
                 ,DESTID=name
                 ,DSCNAME=name
                 [,LENGTH=length]
                 [,OPEN={INITIAL|DEFERRED}]
                 [,RMTNAME=name]
                 [,SYSIDNT=name]
```

TYPE=EXTRA
Indicates an extrapartition destination.

DESTID=name
Code this with the symbolic name of the extrapartition destination. The symbolic name is used in the transient data operations to specify the destination.

Any DESTID of more than four characters is truncated on the right.

The DESTID should not start with the letter C, which is reserved for defining the destinations required by some CICS facilities. This applies to DFHDCT TYPE=EXTRA, TYPE=INDIRECT, and TYPE=INTRA. See "Required entries in the destination control table" on page 387 for a list of these destinations.

Do not use special characters, lower case, or mixed case characters in a DESTID name.

DSCNAME=name
Code this with the same file name you used in DFHDCT TYPE=SDSCI.

If two or more extrapartition destinations refer to the same SDSCI, only one destination can be open at the same time.

If OPEN=INITIAL is specified for each of these destinations, the choice of destination to be opened is arbitrary. To avoid this, define one of the destinations as extrapartition and the others as indirect on the first.

LENGTH=length
Specifies, as a decimal value, the record length in bytes of fixed-length records in the queue. The length you specify must correspond to the RECSIZE length on the associated SDSCI entry in the DCT.

A CICS region that references a remote DCT entry requires the length of the record. If you do not specify it on the DCT entry, the application program must specify it on the WRITEQ and READQ requests

If you omit the SYSIDNT parameter, LENGTH is ignored.

OPEN={INITIAL|DEFERRED}
Code this with the initial status of the data set.

INITIAL
The data set is to be opened by system initialization.

DEFERRED
The data set remains closed until you indicate that you want to open it by using the CEMT INQUIRE|SET TDQUEUE command.

RMTNAME=name
code this with the 1- to 4-character name by which the destination is known in the CICS region in which the destination resides (the remote region).

If you omit this parameter, CICS uses the name specified on the DESTID parameter (that is, the local and remote names are the same).

This parameter is meaningful only when you specify the SYSIDNT parameter.

SYSIDNT=name
Identifies the CICS region in which the remote transient data queue resides. The 4-character alphanumeric name specified must match the SYSIDNT system initialization parameter specified on the region that ″owns″ the queue (the region in which the queue is a local resource).

If you omit SYSIDNT, the queue is treated as a local queue.

Indirect destinations—DFHDCT TYPE=INDIRECT

An indirect destination is specified by the DFHDCT TYPE=INDIRECT macro. The indirect destination does not point to an actual data set, but to another destination. This may be extrapartition, intrapartition, or remote. It may even be another indirect destination.

For example, you can give a different symbolic name (DESTID) to each of several different message types. You can then send all these message types either to the same physical destination (INDDEST), or to different physical destinations.

The DFH$TDWT sample program demonstrates how you can use indirect destinations to send different categories of message to the same terminal. For programming information about DFH$TDWT, see the *CICS Customization Guide*.

If you use EXEC CICS INQUIRE TDQUEUE, information is always returned about an indirect queue. (This does not, however, guarantee that the inquiry transaction is able to use EXEC CICS INQUIRE TDQUEUE for the ultimate target queue.)

If the QUEUE operand of an EXEC CICS WRITEQ TD (or READQ or DELETEQ) command specifies an indirect queue, access is determined by the security setting of the ultimate target queue.

Table 37. DFHDCT TYPE=INDIRECT

```
          DFHDCT          TYPE=INDIRECT
                          ,DESTID=name
                          ,INDDEST=name
```

TYPE=INDIRECT
Indicates an indirect destination.

DESTID=name
Code this with the 1-through 4-character symbolic name of the indirect destination. The symbolic name is used when writing to the destination.

You must not use special characters, lower case, or mixed case characters in a DESTID name.

INDDEST=name
Code this with the name (DESTID) of a transient data destination. The destination can be intrapartition, extrapartition, remote, or indirect. If there is no DCT entry for the destination with this name, an assembly error results.

Intrapartition destinations—DFHDCT TYPE=INTRA

This macro specifies a destination for data that is to be stored temporarily.

An intrapartition destination may be a terminal, a file, or another system. A single data set, managed by VSAM, is used to hold the data for all intrapartition destinations. This macro must be coded once for every intrapartition destination.

You can specify a transaction to process the records and a **trigger level** for each intrapartition destination. The trigger level represents a number of records. When this number of records has been accumulated, the specified transaction is initiated.

The intrapartition destination may be defined as logically recoverable, physically recoverable, or not recoverable.

Logically recoverable destinations are restored (after individual transaction failures and after total system failures) to the status they had at the end of the last completed LUW. (A **logical unit of work (LUW)** begins at start of task or at a **synchronization (sync) point**, and ends at end of task or at a syncpoint.)

Physically recoverable destinations are restored (after a total system failure) to the status they had when the system failure occurred.

Table 38. DFHDCT TYPE=INTRA

```
DFHDCT        TYPE=INTRA
              ,DESTID=name
              [,DESTFAC={(TERMINAL[,trmidnt])
              | | FILE[,USERID=name]|
              (SYSTEM,sysidnt)}]
              [,DESTRCV={NO|PH|LG}] |

              [,RMTNAME=name]
              [,SYSIDNT=name]
              [,TRANSID=name]
              [,TRIGLEV={1 number}] |
```

TYPE=INTRA
Indicates an intrapartition destination.

DESTID=name
Code this with the symbolic name of the intrapartition destination. The symbolic name is used to identify the intrapartition queue for I/O operations. It must not be more than four characters in length.

You must not use special characters, lower case, or mixed case characters in a DESTID name.

DESTFAC={(TERMINAL[,trmidnt])| FILE[,USERID=name]|(SYSTEM,sysidnt)}
Code this with the type of destination that the queue represents.

(TERMINAL[,trmidnt])
The transient data destination is to be associated with the terminal identified by *trmidnt*. The terminal must be defined to CICS using either an RDO TERMINAL definition, or a DFHTCT TYPE=TERMINAL macro.

If you do not specify *trmidnt*, it defaults to the value of DESTID. If ATI is used, as specified in the TRANSID and TRIGLEV operands, the transaction that is initiated is associated with the specified terminal, which must be available before the transaction can be initiated.

FILE
The transient data destination is to be used as a file of data records that are not associated with a particular terminal or system. ATI does not require a terminal to be available.

USERID=name
Code this with the userid that you want CICS to use for security checking for the trigger-level transaction specified on the TRANSID operand. USERID is valid only when the destination is defined as DESTFAC=FILE.

The trigger-level transaction runs under the authority of the specified userid, which must be authorized to all the resources used by the transaction.

If you omit the userid from a qualifying trigger-level entry, CICS uses the userid specified on the TYPE=INITIAL macro. If you omit the userid from the TYPE=INITIAL macro also, CICS uses the CICS default userid, specified on the DFLTUSER system initialization parameter. You must ensure that the CICS region userid of any CICS region in which this DCT is installed, is defined as a surrogate for all the userids specified in the DCT. This is because, during initialization, CICS performs a surrogate user security check against the CICS region userid. If this check fails, CICS deactivates automatic transaction initiation by trigger-level for the intrapartition queue for which the surrogate check failed.

For further information about surrogate user security, see the *CICS RACF Security Guide*.

(SYSTEM,sysidnt)
The transient data destination is to be associated with a system identified by sysidnt. The system must be defined to the local CICS system using an RDO CONNECTION definition.

The primary purpose of coding DESTFAC=SYSTEM,*sysidnt* is to initiate a distributed transaction processing (DTP) session. For details of DTP considerations in application programming, see the *CICS Application Programming Guide*.

DESTRCV={NO|PH|LG}
Code this to indicate the recoverability attributes of the destination in the event of an abnormal termination of either CICS or the transaction processing the destination.

NO
This destination is not recoverable, and automatic logging is not to be performed to keep track of accesses to this destination.

Queue records are held on one or more control intervals (CIs); each CI is released as soon as the last record on it has been read.

PH

This destination is physically recoverable, and automatic logging is to be performed to keep track of accesses by application programs. If emergency restart occurs, this destination is to be recovered to its status at the time CICS terminated.

Queue records are held on one or more control intervals (CIs); each CI is released as soon as the last record on it has been read.

LG

This destination is logically recoverable, and automatic logging is to be performed to keep track of accesses by application programs. If a transaction that accessed this destination was in-flight at the time of abnormal termination, in the subsequent emergency restart or dynamic transaction backout this destination is restored to the status it had before the in-flight unit of work modified it.

When this destination is accessed, the task that issued the WRITEQ TD or READQ TD command is enqueued upon the input or output end of the transient data queue, respectively. This enqueue is maintained until the task terminates or issues a syncpoint request to signal the end of a logical unit of work. This is necessary to ensure the integrity of the data being accessed.

Because the enqueues are thus maintained for a longer period of time, an enqueue lockout is possible if an application program that accesses this destination performs what is effectively more than one logical unit of work against it without defining each separate logical unit of work to CICS by issuing the syncpoint request. Furthermore, when a DELETEQ request is issued for a logically recoverable destination, **both** the input and output ends of the queue are enqueued upon. This increases the probability of an enqueue lockout.

Queue records are held on one or more control intervals (CIs); each CI is marked for release as soon as the last record on it has been read. However, the release does not occur until the end of task or until after the next user syncpoint.

RMTNAME=name

code this with the 1- to 4-character name by which the destination is known in the CICS region in which the destination resides (the remote region).

If you omit this parameter, CICS uses the name specified on the DESTID parameter (that is, the local and remote names are the same).

This parameter is meaningful only when you specify the SYSIDNT parameter.

SYSIDNT=name

Identifies the CICS region in which the remote transient data queue resides. The 4-character alphanumeric name specified must match the SYSIDNT system initialization parameter specified on the region that ″owns″ the queue (the region in which the queue is a local resource).

If you omit SYSIDNT, the queue is treated as a local queue.

TRANSID=name
Code this to identify the transaction that is to be automatically initiated when the trigger level is reached. The purpose of transactions that are initiated in such a way is to read records from the destination. If this operand is omitted, or if TRIGLEV=0 is coded, some other means must be used to schedule transactions to read records from the destinations.

This transaction must not reside in a remote CICS system, or be defined as dynamic. If it does, the transaction initiation fails and an attention message is issued to the console.

TRIGLEV={1|number}
Code this with the number of records to be accumulated before a task is automatically initiated to process them. (This number is known as the **trigger level**.) If you code the TRANSID operand, TRIGLEV defaults to 1. The maximum TRIGLEV value is 32767.

If you do not specify a transaction id, the trigger level is ignored.

If you have coded DESTFAC=TERMINAL, the task is not initiated until the terminal is available.

If you have coded DESTFAC=FILE, no terminal is necessary for the task to be initiated. For a non-terminal destination, if a **maximum task**, **short-on-storage** or **no-space** condition exists, the task is not initiated. This is also true during stages 1 and 2 of initialization, and during the final stage of shutdown. It is initiated when the stress condition no longer exists and a subsequent TD WRITE occurs.

If a VTAM terminal is defined, with TRIGLEV=1, as the destination for CSTL on two ISC CICS systems, a performance problem may arise when both systems repeatedly acquire and release the terminal in order to write out the session started and session ended messages.

During CICS operation, the trigger level can be changed using the CEMT transaction. If the trigger level is reduced to a number that is equal to or less than the number of records accumulated so far, the task is initiated when the next record is sent to the destination.

Remote destinations—DFHDCT TYPE=REMOTE

The DFHDCT TYPE=REMOTE macro defines a transient data destination that is owned by another CICS system or region. The destination must also have a complete definition in the system or region in which it resides.

Note: If a transient data request includes the SYSID operand, CICS does not refer to the DCT but identifies the specified destination as remote.

Table 39. DFHDCT TYPE=REMOTE

```
          DFHDCT          TYPE=REMOTE
                          ,DESTID=name
                          ,SYSIDNT=name
                          [,LENGTH=length]
                          [,RMTNAME=name]
```

TYPE=REMOTE
Indicates that this DCT entry identifies a remote transient data destination.

DESTID=name
Code this with a 4-character name by which the destination is known to application programs in the local system or region. For further information, see the RMTNAME operand below.

Do not use special characters, lower case, or mixed case characters in a DESTID name. The DESTID name should not start with a ’C’ unless the name is known to CICS.

SYSIDNT=name
Code this with the 4-character alphanumeric name of the system or region in which the remote transient data destination resides. The name specified must be the same as that given in the CONNECTION name of the RDO definition. (For more guidance information about the CONNECTION option, see “CONNECTION” on page 157.)

LENGTH=length
Code this with the length in bytes of fixed records for a remote destination. The value specified must correspond to that specified for the DCT in the system or region in which the destination resides. If a value is not specified for the LENGTH operand, code the LENGTH parameter in READQ or WRITEQ requests in the application program.

RMTNAME=name
Code this with the 4-character name by which the destination is known in the system or region in which that destination resides. If this operand is omitted (the normal case), the name specified in the DESTID operand is used.

If more than one system or region has a destination with the same name, the DESTID operand allows the definition of an alias that routes a transient data request to a specific system or region. RMTNAME defines the common name, SYSIDNT defines the system or region, and DESTID defines the unique alias. A transient data request using the alias identifies the remote name **and** the system or region to which the request is shipped.

Examples:

1. Destination **A001** is owned by system **A**.

SYSTEM	*DESTID*	*RMTNAME*
A	A001	A001
B	B001	A001
D	D001	A001
E	E001	A001

In system **A**, both RMTNAME and DESTID are **A001**. In systems **B**, **D**, and **E**, RMTNAME is **A001**, but DESTID must be different.

2. Four systems **A**, **B**, **D**, and **E** each own a different destination, but each destination has the same name **X001**.

Each system has a definition for its local destination and each of the three remote destinations.

SYSTEM	*DESTID*	*SYSIDNT*	*RMTNAME*
A	A002	A	X001
A	B002	B	X001
A	D002	D	X001
A	E002	E	X001

Each remote definition in system **A**:
- Has the RMTNAME **X001**
- Defines the remote system with the SYSIDNT parameter **B**, **D**, or **E**
- Uses the DESTID parameter to define a unique alias for use by application programs in the local region

DFHDCT example

Figure 49 shows an example of the coding for a DCT. This DCT includes an extrapartition destination and three intrapartition destinations.

```
DFHDCT TYPE=INITIAL
DFHDCT TYPE=SDSCI,DSCNAME=AAAXTRA,  *
       RECFORM=FIXUNB               *
DFHDCT TYPE=EXTRA,DSCNAME=AAAXTRA,  *
       DESTID=BETA
DFHDCT TYPE=INTRA,DESTID=GAMA
DFHDCT TYPE=INTRA,DESTID=SAMA
DFHDCT TYPE=INTRA,DESTID=DAMA,      *
       TRIGLEV=5,DESTFAC=TERMINAL,  *
       TRANSID=AUTO
DFHDCT TYPE=FINAL
END
```

Figure 49. Extrapartition and intrapartition destinations

Required entries in the destination control table

The following destinations are used by some of the CICS-supplied transactions. The copybook DFH$DCTR in the CICS sample library (CICSTS13.CICS.SDFHSAMP) contains the macros for these destinations. This contains the TYPE=EXTRA entries. The TYPE=SDSCI entries are contained in DFH$DCTD, which should also be used.

- Include these entries in the generation of the DCT, if the associated functions are being used.
- Always enable these destinations. If you define any of them as indirect destinations, its final target destination and any associated transactions (if applicable) must also be enabled.
- Do not code these entries as remote destinations, either directly or by using the DFHDCT TYPE=INDIRECT macro to make the final destination remote.
- Do not code these destinations, either directly or indirectly, as logically recoverable intrapartition destinations.
- You **may** code these destinations as physically recoverable. The advantage of this is that the contents of the transient data queue are not lost across an emergency restart and may provide useful diagnostic information (for example, the CSMT destination). The disadvantage is that emergency restart may take longer because more data has to be read from the log (in extreme cases, CICS may not be able to make an emergency restart).
- Except where noted otherwise, data records written to these destinations have a variable length of up to 120 bytes.

If you define these destinations as extrapartition, the associated SDSCI or DD statement must specify V format records with a minimum BLKSIZE of 128 bytes.

CICS may need certain definitions. The preferred method is to use RDO to define these resources. You can make definitions for these queues available by including DFH$DCTG within a list that is installed as part of an initial or cold start (see "Appendix B. CICS-supplied resource definitions, groups, and lists" on page 503). DFH$DCTR is a sample that allows these definitions to be defined using the macro. See "Required entries in the DCT" on page 281 for information on these definitions and how to install them using DFH$DCTG.

DESTID=CADL (needed to log VTAM resource definitions)
For VTAM resources, this destination keeps a log of each RDO definition installed in the active CICS system. The log records both the installation of entries in the TCT, and the deletion of autoinstalled entries from the TCT. It records definitions installed in each of the following ways:
- By autoinstall
- Using CEDA INSTALL
- At system initialization.

For details about how to define CADL and CSDL, see the *CICS System Definition Guide*.

DESTID=CAIL (needed to log autoinstall terminal model definitions)
The autoinstall terminal model manager (AITM) uses this destination to log all autoinstall terminal model entries installed in, and deleted from, the TCT.

DESTID=CCPI (needed for CPI Communications messages)
The common programming interface for communications (CPI Communications) writes messages to this destination.

DESTID=CCSE (needed for C language support)
Under CICS, C/370 directs the C standard streams to transient data queues. (The queue names are fixed in CICS: thus, the C standard streams cannot be redirected to other queues.) C programs write to the CCSE queue by writing to stderr. You may code this destination as extrapartition, intrapartition, or indirect. If you do not provide a DCT entry for the CCSE queue, writing to stderr in C programs will fail.

DESTID=CCSI (optional, for C language support)
The CCSI queue is reserved for stdin, the C standard stream for input data. Although the CCSI queue name is reserved for stdin, any attempt to read from stdin in CICS results in EOF being returned. This is why this destination has been described as an "optional" entry in the DCT. You may code this destination as extrapartition, intrapartition, or indirect.

DESTID=CCSO (needed for C language support)
CCSO is associated with stdout, the C standard stream for output data. C programs write to the CCSO queue by writing to stdout. You may code this destination as extrapartition, intrapartition, or indirect. If you do not provide a DCT entry for the CCSO queue, writing to stdout in C programs will fail.

The copybook DFH$DCTR has an entry provided for both output queues directed to //COUT. If you use the CICS-supplied sample definitions as the basis for your own DCT, add a DD statement for COUT to your CICS start-up job streams.

DESTID=CDBC (needed for DBCTL DFHDB81xx messages)
CDBC is defined as an indirect queue, which points to the CSML extrapartition

queue. Only DBCTL DFHDB81xx messages use this data log; the remaining messages use either the terminal or the console.

DESTID=CDUL (needed for transaction dump messages)
CDUL is the destination for transaction dump messages. If a transaction dump is requested, for example after a transaction abend, a message is written to this destination to show that a dump has been taken or to give a reason why the dump was suppressed.

DESTID=CESE (needed for run-time output from Language Environment)
For Language Environment, all run-time output is written to this transient data queue. For further information, see the *OS/390 Language Environment Programming Guide*.

DESTID=CMIG (needed for migration log)
CMIG is a **migration log**, that receives messages reporting the use of functions that are no longer supported in CICS (for example, the EXEC CICS ADDRESS CSA command). You can define CMIG as an intrapartition, extrapartition, or indirect destination.

DESTID=CPLI, DESTID=CPLD (both needed for CICS PL/I support)
CPLI is the destination for SYSPRINT output, and CPLD for PLIDUMP output. The minimum logical record size is 137 for CPLI and 125 for CPLD. If these destinations are extrapartition (direct or indirect), they must be V format. See the installation manual for your PL/I compiler for more details.

DESTID=CRDI (needed to log program resource definitions)
This destination provides a log of installed resource definitions for programs, transactions, maps, and mapsets.

DESTID=CSCS (needed for the sign-on transaction)
CSCS receives a message giving details of each sign-on and sign-off. It also receives a message about each rejected attempt at sign-on and each resource authorization failure. This destination can be of any type.

DESTID=CSDL (needed to log RDO commands)
The resource definition online (RDO) transactions write to this destination all commands that result in changes to the CICS system definition (CSD) file or active CICS system.

CSDL is required only if you use RDO and want to keep a log of commands.

The maximum length of data records written to CSDL is 128 bytes. If you define CSDL as extrapartition, the associated SDSCI or DD statement should specify V format records with a minimum BLKSIZE of 136 bytes.

DESTID=CSFL (needed to log file resource definitions)
CSFL is a log of all file resource definitions installed in the active CICS system. Deletions of file resource entries are also logged here.

DESTID=CSKL (needed to log transaction and profile resource definitions)
CSKL is a log of all transaction and profile resource definitions installed in the active CICS system. Deletions are also logged here.

DESTID=CSML (needed for the sign-off transaction)
CICS sign-off writes data to this destination.

DESTID=CSMT (needed for terminal error and abend messages)
The terminal abnormal condition program (DFHTACP) and abnormal condition program (DFHACP) write terminal error and ABEND messages,

respectively, to this destination. You may code this destination as extrapartition, intrapartition, or indirect.

DESTID=CSNE (needed for node error messages)
The node abnormal condition program (DFHZNAC) and the node error program (DFHZNEP) write terminal error messages and data to this destination. You can code this destination as extrapartition, intrapartition, or indirect.

DESTID=CSPL (needed to log program resource definitions)
CSPL is a log of all program resource definitions installed in the active CICS system. Deletions are also logged here.

DESTID=CSRL (needed to log partner resource definitions)
CSRL is a log of all partner resources installed in the active CICS system. Deletions are also recorded here. For more information about partner resources, see "PARTNER" on page 229.

DESTID=CSSL (needed for recovery utility statistics)
The recovery utility program (DFHRUP) writes statistics to this destination. This destination needs a minimum logical record length of 132 bytes and a minimum blocksize of 136 bytes.

DESTID=CSTL (needed for terminal I/O error messages)
The terminal abnormal condition program (DFHTACP) writes terminal I/O error messages to this destination. You may code this destination as extrapartition, intrapartition, or indirect.

DESTID=CSZL (needed for the Front End Programming Interface)
If you have installed the CICS FEPI feature, CSZL is used as the destination for FEPI messages. For information on FEPI transient data destinations, see the *CICS Front End Programming Interface User's Guide*.

DESTID=CSZX (needed for the Front End Programming Interface)
If you have installed the CICS FEPI feature, CSZX is intended for use with a triggered transaction. For information on FEPI transient data destinations, see the *CICS Front End Programming Interface User's Guide*.

PDIR—DL/I directory

The PDIR is a directory of all the remote program specification blocks (PSBs) that are accessed by the CICS system.

To use remote DL/I, you must define the PDIR to CICS. For DBCTL, you must define the DL/I directories to DBCTL using IMS-supplied macros.

To create a program specification block directory (PDIR), code DFHDLPSB TYPE=INITIAL, TYPE=ENTRY, and TYPE=FINAL macros. The TYPE=FINAL (see page 371) must be followed by:

```
END DFSIDIR0
```

Control section—DFHDLPSB TYPE=INITIAL

The DFHDLPSB TYPE=INITIAL macro has the following format and operands:

Table 40. DFHDLPSB TYPE=INITIAL

DFHDLPSB	TYPE=INITIAL

For general information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see “TYPE=INITIAL (control section)” on page 370.

Program specification blocks—DFHDLPSB TYPE=ENTRY

The DFHDLPSB TYPE=ENTRY macro has the following format and operands:

Table 41. DFHDLPSB TYPE=ENTRY

DFHDLPSB	TYPE=ENTRY
	,PSB=psbname
	,MXSSASZ=value
	[,RMTNAME=name]
	,SYSIDNT=name

TYPE=ENTRY
Indicates that an entry is to be generated in the PDIR. The maximum number of entries that can be included in the PDIR is 32760.

PSB=psbname
Specifies the name of the program specification block (PSB) accessed through the remote DL/I. The entry must specify the SYSIDNT and MXSSASZ (and optionally, RMTNAME) operands.

MXSSASZ=value
Specifies the maximum size in bytes of a segment search argument to be used for this PSB.

Note: An excessively large value for MXSSASZ affects performance badly, and may lead to a data stream being shipped which is too large for the connected CICS system.

RMTNAME=name
Indicates the name by which the PSB is known in the remote system or region. The default is the *psbname* specified in the PSB operand.

SYSIDNT=name
Indicates the 4-character alphanumeric name of the remote system or region for which the PSB is applicable. The name specified must be the name of the CONNECTION definition for the remote system.

If the SYSIDNT of the local system is specified, the request is routed to DBCTL and not function-shipped to a remote region. This allows the same PDIR to be used on both sides of the link if this is required. However, it is not necessary to have a PDIR when communicating with DBCTL.

FCT—file control table

The file control table (FCT) describes to CICS the user files that are processed by file management. CICS user files correspond to physical data sets that must have been defined to MVS and allocated to the CICS system before they are used. Because CICS file management processes only VSAM and BDAM data sets, you define any sequential data sets as extrapartition destinations by using the DFHDCT macro.

To define VSAM files, use CEDA or DFHCSDUP. See "FILE" on page 190 for information on defining VSAM files.

DFHFCT macro types

The following macros specify file characteristics, and some of the characteristics of data sets referenced by the files:

- DFHFCT TYPE=INITIAL establishes the beginning of the FCT.
- DFHFCT TYPE=FILE defines the characteristics of a file, such as record characteristics and types of service allowed. (Note that TYPE=DATASET is retained for compatibility with previous releases, and means exactly the same as TYPE=FILE.)
- DFHFCT TYPE=GROUP is used to migrate file definitions to the CSD file.
- DFHFCT TYPE=REMOTE defines a file that is owned by another system or region.
- DFHFCT TYPE=FINAL concludes the FCT. (See page 371.)

Control section—DFHFCT TYPE=INITIAL

The DFHFCT TYPE=INITIAL macro establishes the control sections into which the FCT is assembled, and must be coded as the first statement in the source used to assemble the FCT.

Table 42. DFHFCT TYPE=INITIAL

```
        DFHFCT          TYPE=INITIAL
                        [,SUFFIX=xxx]
                        [,MIGRATE={YES|COMPLETE}]
```

For general information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see "TYPE=INITIAL (control section)" on page 370.

MIGRATE={YES|COMPLETE}
This operand controls the building of FCT entries for VSAM files that are eligible for resource definition online (RDO). The only way RDO-eligible resources may be moved to the CSD file from the macro source is to use DFHCSDUP, as described under "YES" below. For more detailed guidance, see "MIGRATE" on page 102.

YES
Indicates that you want to generate the necessary data to migrate your RDO-eligible resources. The records generated from the macro source are designed to be used as input to the DFHCSDUP utility program. An MNOTE attention message is issued for each RDO-eligible resource. The DFHCSDUP MIGRATE command will convert them into resource definitions on the CSD file. These can then be managed using RDO.

COMPLETE
Use of COMPLETE means that FCT entries are not generated from the macro source for any RDO-eligible files. For each one, the assembly

produces an MNOTE. This means that you can keep your FCT macro source code after you have migrated your definitions.

If you continue to assemble an FCT for resources that are not eligible for RDO, continue to use MIGRATE=COMPLETE.

Local files—DFHFCT TYPE=FILE

The DFHFCT TYPE=FILE macro describes to CICS file control the physical and operational characteristics of a BDAM file. This macro includes operands that provide information about the access method, record characteristics, and types of service allowed for the file. This information is used to generate control # information used by CICS as well as a DCB.

Table 43. DFHFCT TYPE=FILE

```
DFHFCT       TYPE=FILE
             ,ACCMETH=BDAM
             ,FILE=name
             [,DISP={OLD|SHR}]
             [,DSNAME=name]
             [,FILSTAT=({ENABLED|DISABLED|
                 UNENABLED},{OPENED|CLOSED})]
             [,JID={NO|nn}]
             [,JREQ={ALL|(request[,request,...])}]  |
             [,LOG={NO|YES}]
             [,RECFORM=([{UNDEFINED|VARIABLE
                 |FIXED}]
                 [,{BLOCKED|UNBLOCKED}]
                 ,[DCB  format])]
             [,SERVREQ=(request[,request],...)]
             [,BLKKEYL=length]
             [,BLKSIZE=(length[,length])]
             [,KEYLEN=length]
             [,LRECL=length]
             [,RELTYPE={BLK|DEC|HEX}]
             [,RKP=number]
             [,SRCHM=number]
             [,VERIFY=YES]
```

TYPE=FILE
Indicates that this macro describes the characteristics of a file. (Note that TYPE=DATASET is retained for compatibility with previous releases, and means exactly the same as TYPE=FILE.)

ACCMETH=BDAM
specifies the access method of the data set that is allocated to the file.

BDAM
Basic Direct Access Method.

Note: On a cold start of CICS, only BDAM files are installed from DFHFCT; VSAM file definitions are not installed. If you wish to use existing VSAM file definitions, you must migrate your DFHFCT definitions to the CSD file before using CICS Transaction Server for OS/390.

BLKKEYL=length
Code this with a decimal value from 1 through 255, which represents the

length in bytes of the physical key in the BDAM physical record. You must code this operand only for files that reference data sets with physical keys (that is, those with SERVREQ=KEY specified). If a data set contains blocked records, and deblocking is to be performed by using a logical key (that is, a key embedded within each logical record), the logical key length must be specified by using the KEYLEN operand.

If necessary, CICS can place a record under exclusive control by building an ENQ argument by concatenating the data set name, the block reference, and the physical key. An ENQ is issued using a maximum of 255 bytes of this argument. If the argument exceeds 255 bytes in length, the ENQ places a range of keys under exclusive control.

BLKSIZE=(length[,length])

Code this with the length of the block, in bytes. The way you calculate the BLKSIZE depends on the RECFORM. For UNDEFINED or VARIABLE blocks, the length must be the maximum block length. For FIXED length blocks, you calculate the BLKSIZE as follows:

`BLKSIZE = LRECL` for unblocked records.

`BLKSIZE = (LRECL x blocking factor)` for blocked records.

If you wish to have a BLKSIZE value generated in the DCB, you must specify that value in the second parameter of the operand; for example, BLKSIZE=(250,250), where the first 250 relates to the FCT and the second 250 relates to the DCB. If the second parameter is not coded, the DCB is generated without a BLKSIZE value.

Notes:

1. CICS assumes that the block size of the BDAM data set is the size you have specified on the first BLKSIZE parameter. If the value specified is smaller than the actual block size of the data set, you will probably get storage violations or other unpredictable results when using the file.
2. If you specify the second parameter (the DCB value) the value that you code must always be the actual block size. We recommend that you either omit the second parameter, or make it equal to the first parameter.

DISP={OLD|SHR}

Code this to specify the disposition of the data set which will be allocated to this file. If no JCL statement exists for this file when it is opened, the open is preceded by a dynamic allocation of the file using this disposition. If a JCL statement does exist, it will take precedence over this disposition.

OLD The disposition of the data set is set to OLD if dynamic allocation is performed.

SHR The disposition of the data set is set to SHR if dynamic allocation is performed.

Note: You must specify the disposition of the data set, either with the DISP operand, or:
- In a JCL statement
- With CEMT SET
- With EXEC CICS SET

If you specify the disposition in a JCL statement, specify the data set name in the JCL statement as well.

DSNAME=name
Code from 1 to 44 characters to specify the JCL data set name (DSNAME) to be used for this file. If no JCL statement exists for this file when it is opened, the open will be preceded by a dynamic allocation of the file using this DSNAME. If a JCL statement does exist, it will take precedence over this DSNAME.

You must specify the data set name, either with the DSNAME operand, or:
- In a JCL statement
- With CEMT SET
- With EXEC CICS SET

If you specify the data set in a JCL statement, you **must** specify the disposition in the JCL statement as well.

Note: You define the CICS system definition (CSD) file by system initialization parameters, not in the FCT.

FILE=name
Code this with a 1-to 8-character symbolic name by which this FCT entry is to be identified. This name is known as the **file name** and is used by CICS or by CICS application programs to refer to the data set with which this FCT entry has been associated. (Note that the DATASET operand is retained for compatibility with previous releases, and means exactly the same as the FILE operand.)

As well as identifying the FCT entry, this name is also used as the DDNAME when the associated data set is allocated to CICS. The allocation is achieved either by using JCL statements in the start-up job stream, or dynamically, by using the DSNAME and DISP values in the FCT.

Do not use file names that start with the character string DFH for your own files, because CICS reserves the right to use any file name beginning with DFH. In addition, using the character string FCT for a file name prefix can cause assembly errors.

FILSTAT=({ENABLED | DISABLED | UNENABLED}, {OPENED | CLOSED})
Code this to specify the initial status of the file.

The first operand determines the initial enablement state of the file. It is used only during an initial or a cold start. (On a warm or emergency start, the file state is determined by the state at the time of the previous shutdown.)

The second operand specifies whether an attempt is made to open the file at the end of CICS initialization. It applies to initial, cold, warm, and emergency starts.

ENABLED
Normal processing is to be allowed against this file.

DISABLED
Any request against this file from an application program causes the DISABLED condition to be passed to the program.

UNENABLED
This option is valid only with the CLOSED option. It may be used to prevent the file being opened on first reference. An attempt to access the file in this state raises the NOTOPEN condition.

OPENED
The file is opened by an automatically initiated CICS transaction (CSFU) after CICS initialization. (On a warm or emergency start, a file remains UNENABLED, if that was its state at the time of the previous shutdown. CSFU ignores an OPENED option on an UNENABLED file, and leaves the file closed.)

CLOSED
The file is to remain closed until a request is made to open it by the master terminal function, by an EXEC CICS SET command, or by an implicit open.

For each combination of initial states, files are opened as follows:

(ENABLED,CLOSED)
The file is opened on first reference. This is the default.

(ENABLED,OPENED)
The file is opened by the automatically-initiated transaction CSFU after CICS initialization, unless a user application or master terminal function has opened it first.

(DISABLED,CLOSED)
The file is opened only by an explicit OPEN request (for example, from the master terminal transaction).

(DISABLED,OPENED)
The file is opened by the automatically-initiated transaction CSFU after CICS initialization, unless a user application or master terminal function has explicitly opened it first.

(UNENABLED,CLOSED)
The file is opened only by an explicit OPEN request. The file state after it has been opened is (ENABLED,OPENED).

Note: For performance reasons, the default CSD file entry for transaction CSFU is defined with DTIMOUT=10 (seconds). This can cause a transaction timeout abend if there is a delay in opening a file during CICS startup. See page 316 for an explanation of the DTIMOUT value on TRANSACTION definitions.

JID={NO|number}
Code this if automatic journal activity is to take place for this FCT entry, and to identify the journal to be used to record the journaled data. The operations that cause data records to be journaled are specified in the JREQ parameter.

NO
No automatic journaling activity for this file is to take place.

number
The journal identifier to be used for automatic journaling. This may be any number in the range 01 through 99. The number is appended to the letters DFHJ to give a journal name of the form DFHJ*nn*, which maps to an MVS system logger general log stream.

Note: Automatic journaling can be specified if you wish to record file activity for subsequent processing by yourself (for example, user-written data set I/O recovery). It must not be confused with automatic logging (specified with LOG=YES), which is required if CICS is to perform data set backout to remove in-flight task activity during emergency restart or dynamic transaction backout.

JREQ={ALL | (request[,request,...])}

Code this with the file operations that are to be automatically journaled, and whether the journaling operation is to be **synchronous** or **asynchronous** with file activity.

When a synchronous journal operation is executed for a READ request, control is not returned to the program that issued the file control request until the data read is written in the journal data set. When a synchronous journal operation is executed for a WRITE request, the output operation to the data set is not initiated until the data is written in the journal data set.

When an asynchronous journal operation is executed for a READ request, control can be returned as soon as the data read is moved to the journal I/O buffer. When an asynchronous journal operation is executed for a WRITE request, the output operation to the data set can be initiated as soon as the data is moved to the journal I/O buffer.

Synchronization defaults provide asynchronous operation for READs and synchronous operation for WRITEs.

If you have requested automatic journaling, the contents of the journal may not accurately reflect the actual changes to a data set, because the request is journaled before the response from the I/O operation is tested.

If this operand is omitted and JID is coded, JREQ defaults to JREQ=(WU,WN).

Here are the possible values for *request*:

ALL	Journal all file activity with READ asynchronous and WRITE synchronous.
ASY	Asynchronous journal operation for WRITE operations.
RO	Journal READ ONLY operations.
RU	Journal READ UPDATE operations.
SYN	Synchronous journal operation for READ operations.
WN	Journal WRITE NEW operations.
WU	Journal WRITE UPDATE operations.

KEYLEN=length

Code this with the length of the logical key for the deblocking of the BDAM data set to which this file refers.

The logical key for BDAM data sets is embedded and located through the use of the RKP operand. The length of the physical key is coded in the BLKKEYL operand, and can be different from the value specified for KEYLEN.

This operand must always be coded when logical keys are used in blocked BDAM data sets.

LOG={NO | YES}

This operand specifies the recovery attributes of the file. Specify LOG=YES if you want automatic logging. This enables backout (recovery) of incomplete changes to the data set referenced by this file, in the event of an emergency restart or transaction abend. Whenever a change—update, deletion, or addition—is made to the data set, the “before” image is automatically recorded in the CICS system log. (Automatic logging should not be confused with automatic journaling.)

Note: If RLS=YES, the recovery attribute in the ICF catalog overrides the LOG= parameter.

NO
Automatic logging is not to be performed.

YES
Automatic logging is to be performed.

When a request is made to change the contents of the data set referenced by the file, the record being updated, added, or deleted is enqueued upon, using the record identification together with the address of the CICS control block representing the base data set. This enqueue is maintained until the task terminates or the application issues a syncpoint request to signal the end of a logical unit of work. This ensures the integrity of the altered data.

Because the enqueues are thus maintained for a longer period of time, an enqueue lockout can occur if an application program that accesses this data set performs what is effectively more than one logical unit of work against it, without defining each separate logical unit of work to CICS by issuing a syncpoint request. Also, long-running tasks could tie up storage resources.

Always ensure consistency of recovery attributes between files referring to the same base data set cluster or its paths.

To look at the recovery attributes, use the CEMT INQUIRE DSNAME or EXEC CICS INQUIRE DSNAME command on the base cluster to which the file refers. If you use INQUIRE FILE RECOVSTATUS, you do not necessarily find the recovery attribute that CICS has enforced on the base cluster (if there are inconsistencies between files referring to the same base cluster). We explain this in the next paragraph. If all files are consistent, the recovery attribute on the file is the same as on the base cluster.

CICS takes the actions shown in the following list when opening a file for update processing. (That is, if you set SERVREQ=ADD, DELETE, or UPDATE. If you set only SERVREQ=READ and/or BROWSE, CICS does not make these consistency checks.)

- If an FCT entry refers to an alternate index (AIX) path and LOG=YES is specified, the AIX must be in the upgrade set for the base. This means that any changes made to the base data set are also reflected in the AIX. If the AIX is not in the upgrade set, the attempt to open the FCT entry for this AIX path fails.
- If a file is the first to be opened for update against a base cluster after the last initial or cold start, the recovery attribute (LOG=YES or NO) is copied into the base cluster block.
- If a file is not the first to be opened for update against a base cluster after the last initial or cold start, the LOG parameter on the FCT is checked against the LOG parameter held in the base cluster block. These are the possibilities:

- The base cluster has LOG=NO:
  - If the FCT entry is defined with LOG=NO, the open proceeds normally.
  - If the FCT entry is defined with LOG=YES, and other files are currently open against the base cluster, the attempt to open the file fails. If there are no other files open against the base, the open is allowed, and the recovery attribute of the base cluster is updated to LOG=YES. This is the only occasion when the recovery attribute of the base can be safely changed from its original setting.
- The base cluster has LOG=YES:
  - If the FCT entry is defined with LOG=NO, the open proceeds, but the base recovery attribute LOG=YES overrides the setting on the FCT entry.
  - If the FCT entry is defined with LOG=YES, the file open proceeds normally.

CICS also checks whether you have specified **forward** recovery for another file referencing the same data set.

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

The base cluster block, together with its recovery attributes, is preserved even when all the files relating to it are closed, and across warm and emergency restarts.

The file control program decides whether before-and/or after-images are required by inspecting the **data set's** recovery attributes (held in the base cluster block) and not the attributes of the files open against that VSAM data set.

In those cases, listed above, in which the file open proceeds and the data set attributes are used, the file entry is not changed, and the file control program just looks at the base cluster block to obtain the correct recovery settings. The file definitions are not changed because, if they were, the FCT source would not reflect the object code. This could be a problem if the file were later dynamically allocated to another VSAM data set.

LRECL=length
Code this with the maximum length (in bytes) of the logical record. The value specified is also the length of records in a fixed length remote file. See the DFHFCT TYPE=REMOTE macro for further information on remote files.

RECFORM=([{UNDEFINED|VARIABLE|FIXED}],

[{BLOCKED|UNBLOCKED}],[DCB format])
Code this to describe the format of physical records in the data set.

For BDAM data sets, **blocking** refers to CICS blocking, and has no meaning for BDAM. You must specify BLOCKED or UNBLOCKED for all data sets of FIXED or VARIABLE format.

BLOCKED
Specify this option when each physical record is to be viewed by CICS as a block consisting of more than one logical record.

DCB
Code this with the record format to be inserted in the DCB; for example, RECFORM=(FIXED,BLOCKED,FBS).

The DCB format sub-parameter of the RECFORM operand is the only way you can put record format information into the DCB when the FCT is assembled. The first two sub-parameters of the RECFORM operand do not generate information in the DCB.

FIXED
Records are fixed length.

UNBLOCKED
Specify this option when no CICS block structure is to be used. That is, when there is one CICS logical record for each BDAM physical record.

UNDEFINED
Records are of undefined length. (If you specify a data set as UNDEFINED, allow for an additional 8 bytes for the count field, when calculating the BLKSIZE.)

VARIABLE
Records are variable length.

RELTYPE={BLK|DEC|HEX}
Code this if relative addressing is being used in the block reference portion of the record identification field of the BDAM data set referenced by this file. If the RELTYPE operand is omitted, absolute addressing is assumed (that is, MBBCCHHR).

BLK Relative block addressing is being used.

DEC The zoned decimal format is being used.

HEX The hexadecimal relative track and record format is being used.

RKP=number
Code this with the starting position of the key field in the record relative to the
# beginning of the record. With variable-length records, this operand must include space for the 4-byte LLbb field at the beginning of each logical record. This operand must always be coded for data sets that have keys within each logical record, or when browsing.

SERVREQ=(request[,request],...)
Code this to define the types of service request that can be processed against the file. The parameters that can be included are as follows:

ADD
Records can be added to the file.

BROWSE
Records may be sequentially retrieved from the file.

KEY
Records can be retrieved from or added to the file. This parameter is mandatory if the data set referenced by the file is a keyed BDAM data set. It must not be coded for other files.

NOEXCTL
Records are not to be placed under exclusive control when a read for update is requested.

If you do not specify NOEXCTL, BDAM exclusive control is provided by default. This provides integrity in the system. For BDAM, you may specify LOG=YES with SERVREQ=NOEXCTL. This requests only a CICS enqueue and suppress the BDAM exclusive control, thus providing CICS integrity for the update only until a syncpoint.

Note: The CICS enqueue is at the record level within the CICS region, and lasts until a syncpoint, whereas the BDAM exclusive control operates on a physical block, is system-wide, and lasts only until the update is complete.

READ
Records in this file can be read. READ is assumed, if you specify BROWSE or UPDATE.

UPDATE
Records in this file can be changed.

SRCHM=number
Code this if multiple track search for keyed records is to be provided. This operand is applicable only to BDAM keyed data sets.

number
The number of tracks or blocks to be searched. The default is 0.

VERIFY=YES
Code this if you want to check the parity of disk records after they are written. If this operand is omitted, records are not verified after a write request.

Summary table: This section is intended to help you use the DFHFCT TYPE=FILE macro to define your files. Each TYPE=FILE instruction describes the characteristics of the file, and of the data set referenced by the file.

Table 44. DFHFCT TYPE=FILE instructions

	BDAM			
	Blocked		**Unblocked**	
	With key	**Without key**	**With key**	**Without key**
BLKKEYL	R		R	
SRCHM	O		O	
VERIFY	O	O	O	O
RELTYPE	R[1 on page 403]	R[1 on page 403]	R[1 on page 403]	R[1 on page 403]
LRECL	R	R	R	R
BLKSIZE	R[3 on page 403]	R	R[2 on page 403]	R
KEYLEN	R[5 on page 403]			
RKP	R[4 on page 403]		R[4 on page 403]	
RECFORM	O	O	O	O

Table 44. DFHFCT TYPE=FILE instructions (continued)

	BDAM			
	Blocked		**Unblocked**	
	With key	**Without key**	**With key**	**Without key**
FILSTAT	O	O	O	O
SERVREQ	R[6]	O	R[6]	O

Notes:

R Required

O Optional

1. Required if relative type addressing is to be used.
2. If SERVREQ=BROWSE or SERVREQ=ADD, this value must be BLKSIZE + BLKKEYL for unblocked records.
3. If SERVREQ=BROWSE or SERVREQ=ADD, this value must be (LRECL x blocking factor) + BLKKEYL for blocked records.
4. Required if key exists within logical records.
5. Required if deblocking by key for BDAM.
6. SERVREQ=KEY is required.

Migrating FCT definitions—DFHFCT TYPE=GROUP

Use this macro to name the groups into which FCT definitions are put when you migrate to resource definition online. This macro can appear as many times as required and at any point in the macro source. Each time it appears, it defines the CSD file group into which subsequent definitions are put until the next DFHFCT TYPE=GROUP macro occurs.

Table 45. DFHFCT TYPE=GROUP

```
DFHFCT          TYPE=GROUP
                [,GROUP=name]
```

GROUP=name

Code this with the name of the group to which subsequent definitions are migrated. The name can be up to eight alphanumeric characters, but must not begin with DFH. The default name is FCT*xx* where *xx* is the value coded for SUFFIX in the DFHFCT TYPE=INITIAL macro. If an error is found, the existing group name continues.

If a group with the name you specify does not already exist, it is created. If it does exist, subsequent definitions are added to it.

Remote files—DFHFCT TYPE=REMOTE

The DFHFCT TYPE=REMOTE macro defines a file that resides in a remote system or region. You can use function request shipping to access this file from the local system. The remote definition that this macro creates gives less information than is needed for the local definition created by the TYPE=FILE macro. Each file that you define using TYPE=REMOTE also has a TYPE=FILE entry in its own system.

You name the system that owns the file using the SYSIDNT operand. The file may be known by a different name in the system that owns it; in this case, you need to code the RMTNAME operand.

Table 46. DFHFCT TYPE=REMOTE

```
DFHFCT          TYPE=REMOTE
                ,SYSIDNT=name
```

Table 46. DFHFCT TYPE=REMOTE (continued)

```
[,KEYLEN=length]
[,LRECL=length]
[,RMTNAME=name]
```

TYPE=REMOTE
Indicates that this FCT entry identifies a file that resides in a remote system or region.

FILE=name
Code this with a 1-to 8-character file name used by the application programs in the same system as this FCT. (Note that the DATASET operand is retained for compatibility with previous releases, and means exactly the same as the FILE operand.)

KEYLEN=length
Code this with the default key length for a file control request that is sent to a remote system.

For DAM, the value must be equal to the total length of the RIDFLD option required to access the file. In the case of remote blocked files, this length must be the length of the RIDFLD option required to deblock by key. If you do not code this operand, the KEYLENGTH option must be specified in the application program that refers to this file.

LRECL=length
Code this with the default data length in bytes for a READ WRITE or REWRITE request sent to a remote system.

RMTNAME=name
Code this with a 1-to 8-character name by which the file is known to the system or region in which it resides. If this operand is omitted (the normal case), the name coded in the FILE operand is used.

This allows a single file to be known by different names in different systems. For example, suppose A001 is owned by system A. On the definition in system A, both FILE and RMTNAME are A001. In systems B, C, and D the RMTNAME is always A001, but the FILE may be B001, C001, or any other name.

In addition, this allows each system to own a different file with the same name. For example, suppose system A has a local definition for X002, which it owns. System A also has definitions for B002, C002, and D002. These three files, however, are each owned by another system, and the RMTNAME in each of the definitions in system A is X002. In the system that owns it, each of the four files is known as X002, but transactions in each system can distinguish between the four, using a different FILE for each.

SYSIDNT=name
Code this with the 4-character alphanumeric name of the system or region in which the file is resident. The name given must be the name of the CONNECTION definition for the system (or the SYSIDNT operand in the DFHTCT TYPE=SYSTEM macro), or the SYSIDNT in an explicit remote request in an application program.

DFHFCT example

Figure 50 illustrates the coding required to create an FCT entry for a BDAM file.

```
DFHFCT TYPE=FILE,                                *
       FILE=DAM83,                               *
       ACCMETH=BDAM,                             *
       SERVREQ=(READ,BROWSE,KEY),                *
       BLKSIZE=172,                              *
       RECFORM=(FIXED,BLOCKED),                  *
       LRECL=86,                                 *
       RELTYPE=HEX,                              *
       KEYLEN=6,                                 *
       BLKKEYL=6,                                *
       RKP=0,                                    *
       FILSTAT=(ENABLED,OPENED)
```

Figure 50. File control table example—BDAM file

MCT—monitoring control table

The monitoring control table (MCT) defines the user data fields in performance class monitoring records, and describes how they are to be manipulated at the user-coded event monitoring points (EMPs). It also controls which system-defined performance class data fields are recorded. See the *CICS Performance Guide* for details of the MCT and performance.

The MCT is necessary only if you have coded user EMPs in your application programs, or if you need to exclude specific system-defined fields from being recorded. If no MCT is present the following defaults are assumed:

- All monitoring classes are available (exception, performance, and SYSEVENT)
- All CICS system-defined data fields are collected

You code user EMPs in application programs using the EXEC CICS MONITOR command.

Elements of DFHMCT

The MCT consists of the following macro instructions:

- Control section—DFHMCT TYPE=INITIAL
- User event monitoring points—DFHMCT TYPE=EMP
- Control data recording—DFHMCT TYPE=RECORD
- End of monitoring control table—DFHMCT TYPE=FINAL (described on page 371)

Control section—DFHMCT TYPE=INITIAL

The control section name for the MCT is established by the DFHMCT TYPE=INITIAL macro. This macro also creates the necessary linkage editor control statements for subsequent link-editing. The SYSEVENT class of transaction monitoring is always available.

Table 47. DFHMCT TYPE=INITIAL

```
DFHMCT          TYPE=INITIAL
                [,SUFFIX=xx]
```

For the MCT, the suffix shosuld be blank, two valid characters, or NO. For general information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see “TYPE=INITIAL (control section)” on page 370.

User event monitoring points—DFHMCT TYPE=EMP

The DFHMCT TYPE=EMP macro allows you to specify how the user data fields in performance class data records are to be added to or changed at each user event monitoring point. One TYPE=EMP macro must be coded for each user EMP at which user data is required.

The TYPE=EMP macro must be coded between the TYPE=INITIAL macro and the first TYPE=RECORD macro instruction.

Table 48. DFHMCT TYPE=EMP

```
DFHMCT          TYPE=EMP
                ,CLASS=PERFORM
                ,ID={number|(PP,number)
                     |entryname.number}
                [,CLOCK=(number,name1[,name2,...])]
                [,COUNT=(number,name1[,name2,...])]
```

Table 48. DFHMCT TYPE=EMP (continued)

```
                                   [,FIELD=(1,name)]
                                   [,PERFORM=(option[,...])]
```

TYPE=EMP
: Indicates that this macro defines the user data to be collected at a user event monitoring point.

CLASS=PERFORM
: Code this with the monitoring classes for which you want user data to be collected at this user EMP. The value PERFORM must be coded. The corresponding PERFORM operand must also be coded.

ID={number|(PP,number)|entryname.number}
: Code this with the identifier of the user event monitoring point at which the user data defined in this macro is to be collected. Note that if one of the forms *number* or (PP,*number*) is coded, a default entry name, "USER", is provided.

number
: A decimal integer in the range 1 through 255. Identification numbers between 1 and 199 are available for user EMPs. Numbers between 200 and 255 are reserved for IBM program product EMPs and should be coded if you want to collect user data at EMPs defined in the code of IBM program products.

(PP,number)
: An IBM program product EMP identification number. It is equivalent to specifying an ID value of 199 + number. The value of *number* is a decimal integer in the range 1 through 56.

entryname.number
: Allows multiple use of *number*, a decimal integer in the range 1 through 255. Thus 'UNIQUE.3', 'DSN.3', and '3' are three different EMPs. A maximum of 98 entrynames can be specified against any particular number. Furthermore, any count, clock, or byte-offset referred to by one of them is a different object from that referred to by any other.

In the following descriptions, any reference to a constant means a hexadecimal constant of up to eight hexadecimal digits; any shorter string is padded on the left with zeros. For example, to add or subtract decimal 14, the constant would be coded as 0000000E or just E (no quotation marks are required).

Any reference to the fields DATA1 and DATA2 means the two binary fullwords supplied by the user EMP coded in the application program. These are specified by the DATA1 and DATA2 operands of the EXEC CICS MONITOR command for defining user EMPs. Depending on the options coded, the DATA1 and DATA2 fields can be interpreted as numbers, masks for performing logical operations, or pointers to further information.

Any reference to a number means a decimal integer in the range defined in the description of the option.

CLOCK=(number,name1[,name2,...])
: Assigns an informal name to one or more clocks. The informal name of any clock appears in its dictionary entry and is available to a postprocessor for use as, for example, a column heading.

The character string *name1* is assigned to the clock specified by *number* at MCT generation. If specified, *name2* is assigned to the clock *number*+1. Similarly, any subsequent names are assigned to subsequent clocks. Any clock not named by this option receives the entry name value from the ID operand (the default is USER).

Number must be in the range 1 through 256. The names specified must each be a character string up to eight characters long. If any string contains one or more blanks or commas, it must be enclosed in quotes.

COUNT=(number,name1[,name2,...])
Assigns an informal name to one or more count fields. The informal name of any count field appears in its dictionary entry and is available to a postprocessor for use as, for example, a column heading.

The character string *name1* is assigned to the count field specified by number at MCT generation. If specified, *name2* is assigned to the count field *number*+1. Similarly, any subsequent names are assigned to subsequent count fields. Any count fields not named by this option receive the entry name value from the ID operand (the default is USER).

Number must be in the range 1 through 256. The names specified must each be a character string up to eight characters long. If any string contains one or more blanks or commas, it must be enclosed in quotes.

FIELD=(1,name)
Assigns an informal name to the user byte-string field. This appears in its dictionary entry and is available to a postprocessor for use as, for example, a column heading.

Name must be a character string up to 8 characters long. If it contains one or more blanks or commas, it must be enclosed in quotes.

PERFORM=(option[,...])
Code this operand when CLASS=PERFORM is specified. It specifies that information is to be added to or changed in the user fields of the performance class data record at this EMP.

The user fields for each user distinguished by a separate entry name in the ID operand can comprise:

1. Up to 256 counters
2. Up to 256 clocks, each made up of a 4-byte accumulator and 4-byte count
3. A byte string of up to 8192 bytes.

Note: If the combined sizes of the objects (clocks, counts, and fields) implied in the specified options exceed 16384 bytes, assembly-time errors occur. You can avoid this by using fewer objects, either by collecting less data, or by clustering references to clocks and counts to avoid implied, but unused, objects.

Note: When you define user data to be collected at a user event monitoring point, this extends the size of all CICS performance class monitoring records. Each CICS monitoring record is the same size as the largest record; bear this in mind when specifying user data fields.

Actions are performed on the user fields according to the options specified.

PERFORM can be abbreviated to PER. Valid options for the PERFORM operand are:

ADDCNT(number,{constant | DATA1 | DATA2})
The value of the user count field specified by *number* is to be incremented by *constant* or by the value of the field DATA1 or DATA2. *Number* is a decimal integer in the range 1 through 256.

EXCNT(number,{constant | DATA1 | DATA2})
A logical exclusive OR operation is to be performed on the value of the user count field specified by *number*, using *constant* or the value of the field DATA1 or DATA2. *Number* is a decimal integer in the range 1 through 256.

MLTCNT(number1,number2)
A series of adjacent user count fields are to be updated by adding the values contained in adjacent fullwords in an area addressed by the DATA1 field. To use this option, both the DATA1 and DATA2 fields must be passed from the user EMP.

The user count fields that are to be updated start at the field specified by *number1*. The number of user count fields that are updated is the smaller of the values of *number2* and the DATA2 field. If the DATA2 field is zero, the value of *number2* is used. The series of adjacent fullwords used to add into the user count fields starts at the address specified in the DATA1 field. Successive fullwords are added into successive user count fields.

Number1 and *number2* are decimal integers in the range 1 through 256. The number of user counts generated is (*number1* + *number2* - 1). This value must also be in the range 1 through 256.

Note: Only one of the MLTCNT and MOVE options can be used in each DFHMCT TYPE=EMP macro.

MOVE(number3,number4)
A string of data is to be moved into the user byte-string field. To use this option, both the DATA1 and DATA2 fields must be passed from the user EMP.

The user byte-string field is updated starting at the offset specified by *number3*. The data to be moved starts at the address supplied in the DATA1 field. The maximum length of data that can be moved is given by *number4* (in bytes), and the actual length of data that is to be moved is given by the value of the DATA2 field. If the value of DATA2 is zero, the length of the data given by *number4* is moved.

Number3 is a decimal integer in the range 0 to 8191, and *number4* is a decimal integer in the range 1 to 8192. The maximum length of the user character field is (*number3* + *number4*), and must be in the range 1 to 8192.

Note: Only one of the MLTCNT and MOVE options can be used in each DFHMCT TYPE=EMP macro instruction.

NACNT(number,{constant | DATA1 | DATA2})
A logical AND operation is to be performed on the value of the user count field specified by *number*, using *constant* or the value of the field DATA1 or DATA2. *Number* is a decimal integer in the range 1 through 256.

ORCNT(number,{constant | DATA1 | DATA2})
A logical inclusive OR operation is to be performed on the value of the

user count field specified by *number*, using *constant* or the value of the field DATA1 or DATA2. *Number* is a decimal integer in the range 1 through 256.

PCLOCK(number)
The clock specified by number is to be stopped. The 4-byte count in the user clock field is flagged to indicate that the clock is now stopped. The accumulator is set to the sum of its contents before the previous SCLOCK and the elapsed period between that SCLOCK and this PCLOCK. *Number* is a decimal integer in the range 1 through 256.

PCPUCLK(number)
This option performs the same function as PCLOCK, but uses the CPU-time of the CICS main task instead of elapsed time.

SCLOCK(number)
The clock specified by *number* is to be started. The value of the 4-byte count in the user clock field is incremented by 1 and flagged to show its running state. *Number* is a decimal integer in the range 1 through 256.

SCPUCLK(number)
This option performs the same function as SCLOCK, but uses the CPU-time of the CICS main task instead of elapsed time.

SUBCNT(number,{constant|DATA1|DATA2})
The value of the user count field specified by *number* is to be decremented by *constant* or by the value of the field DATA1 or DATA2. *Number* is a decimal integer in the range 1 through 256.

DELIVER
Performance class data accumulated for this task up to this point is delivered to the monitoring buffers. Any running clocks are stopped. The performance class section of the monitoring area for this task is reset to X'00', except for the key fields (transid, termid). Any clocks that were stopped by this option are restarted from zero for the new measurement period. The "high-water-mark" fields are reset to their current values.

Control data recording—DFHMCT TYPE=RECORD

The DFHMCT TYPE=RECORD macro identifies the performance class data fields which have been selected for monitoring.

Table 49. DFHMCT TYPE=RECORD

```
,TYPE=RECORD
,CLASS=PERFORM
[,EXCLUDE={ALL|(n1[,...])}]
[,INCLUDE=(m1[,...])]
```

TYPE=RECORD
Indicates that monitoring data for selected performance class data fields will be recorded.

CLASS=PERFORM
Code this if performance class data fields is to be recorded. You can abbreviate PERFORM to PER.

EXCLUDE={ALL|(n1[,...])}
Code this to prevent one or more CICS fields from being reported by the monitoring facility. By default, all documented performance class fields are reported.

The EXCLUDE operand is always honored before the INCLUDE operand, regardless of the order in which they are coded. (The INCLUDE operand is only relevant when the EXCLUDE operand is coded.)

ALL
This prevents all fields that are eligible for exclusion from being reported. Note that the following fields cannot be excluded:

1, 2, 4, 5, 6, and 89.

You can use the INCLUDE operand at the same time as EXCLUDE=ALL if you want to include some fields but exclude the majority.

Table 50 shows the fields that are eligible for exclusion. Each field has a group name associated with it, which identifies the group of fields to which it belongs. Each field also has its own numeric field identifier.

To exclude a group of fields you code the name of the group (a character string) as *n1*, for example, `EXCLUDE=(DFHTASK)`.

To exclude a single field you code the numeric identifier of the field as *n1*, for example, `EXCLUDE=(98,70)`.

Note: Do not code leading zeros on numeric identifiers. Do not code numeric identifiers of fields that are ineligible for exclusion.

You can code combinations of names and numeric identifiers, for example, `EXCLUDE=(DFHFILE,DFHTERM,112,64)`.

Table 50. This table shows the data groups and fields that can be excluded/included

Group Name	Field Id	Description
DFHCBTS	200	CICS BTS process name
DFHCBTS	201	CICS BTS process type
DFHCBTS	202	CICS BTS process id
DFHCBTS	203	CICS BTS activity id
DFHCBTS	204	CICS BTS activity name
DFHCBTS	205	CICS BTS run process/activity synchronous count
DFHCBTS	206	CICS BTS run process/activity asynchronous count
DFHCBTS	207	CICS BTS link process/activity count
DFHCBTS	208	CICS BTS define process count
DFHCBTS	209	CICS BTS define activity count
DFHCBTS	210	CICS BTS reset process/activity count
DFHCBTS	211	CICS BTS suspend process/activity count
DFHCBTS	212	CICS BTS resume process/activity count
DFHCBTS	213	CICS BTS delete activity or cancel process/activity request count
DFHCBTS	214	CICS BTS acquire process/activity request count
DFHCBTS	215	CICS BTS total process/activity request count
DFHCBTS	216	CICS BTS delete/get/put process container count
DFHCBTS	217	CICS BTS delete/get/put activity container count
DFHCBTS	218	CICS BTS total process/activity container request count
DFHCBTS	219	CICS BTS retrieve reattach request count
DFHCBTS	220	CICS BTS define input event request count
DFHCBTS	221	CICS BTS timer associated event requests count
DFHCBTS	222	CICS BTS total event related request count

Table 50. This table shows the data groups and fields that can be excluded/included (continued)

Group Name	Field Id	Description
DFHCICS	25	CICS OO foundation class request count
DFHCICS	103	Transaction exception wait time
DFHCICS	112	Performance record type
DFHCICS	130	Transaction routing sysid
DFHCICS	131	Performance record count
DFHCICS	167	MVS Workload Manager Service Class name
DFHCICS	168	MVS Workload Manager Report Class name
DFHDATA	179	IMS (DBCTL) request count
DFHDATA	180	DB2 request count
DFHDATA	186	IMS (DBCTL) wait time
DFHDATA	187	DB2 Readyq wait time
DFHDATA	188	DB2 Connection wait time
DFHDATA	189	DB2 wait time
DFHDEST	41	TD get count
DFHDEST	42	TD put count
DFHDEST	43	TD purge count
DFHDEST	91	TD total count
DFHDEST	101	TD I/O wait time
DFHDOCH	226	Document handler Create count
DFHDOCH	227	Document handler Insert count
DFHDOCH	228	Document handler Set count
DFHDOCH	229	Document handler Retrieve count
DFHDOCH	230	Document handler Total count
DFHDOCH	240	Document handler total created document length
DFHFEPI	150	FEPI allocate count
DFHFEPI	151	FEPI receive count
DFHFEPI	152	FEPI send count
DFHFEPI	153	FEPI start count
DFHFEPI	154	FEPI CHARS sent
DFHFEPI	155	FEPI CHARS received
DFHFEPI	156	FEPI suspend time
DFHFEPI	157	FEPI allocate time-out count
DFHFEPI	158	FEPI receive time-out count
DFHFEPI	159	FEPI total count
DFHFILE	36	FC get count
DFHFILE	37	FC put count
DFHFILE	38	FC browse count
DFHFILE	39	FC add count
DFHFILE	40	FC delete count
DFHFILE	63	FC I/O wait time
DFHFILE	70	FC access-method count
DFHFILE	93	FC total count
DFHFILE	174	RLS FC I/O wait time
DFHFILE	175	RLS File request CPU (SRB) time
DFHFILE	176	CFDT I/O wait time

Table 50. This table shows the data groups and fields that can be excluded/included (continued)

Group Name	Field Id	Description
DFHJOUR	10	Journal I/O wait time
DFHJOUR	58	Journal write count
DFHJOUR	172	Log stream write count
DFHMAPP	50	BMS MAP count
DFHMAPP	51	BMS IN count
DFHMAPP	52	BMS OUT count
DFHMAPP	90	BMS total count
DFHPROG	55	Program LINK count
DFHPROG	56	Program XCTL count
DFHPROG	57	Program LOAD count
DFHPROG	71	Program name
DFHPROG	72	Program LINK URM count
DFHPROG	73	Program DPL count
DFHPROG	113	Original abend code
DFHPROG	114	Current abend code
DFHPROG	115	Program load time
DFHSOCK	241	Socket I/O wait time
DFHSOCK	242	Bytes encrypted for secure socket
DFHSOCK	243	Bytes decrypted for secure socket
DFHSOCK	244	Client IP address
DFHSTOR	33	User-storage high-water-mark (UDSA)
DFHSTOR	54	User-storage get-count (UDSA)
DFHSTOR	87	Program-storage high-water-mark - total
DFHSTOR	95	User-storage-occupancy (bytes-ms) (UDSA)
DFHSTOR	105	User-storage get-count–above 16MB line (EUDSA)
DFHSTOR	106	User-storage high-water-mark–above 16MB line (EUDSA)
DFHSTOR	107	User-storage-occupancy (bytes-ms)–above 16MB line (EUDSA)
DFHSTOR	108	Program-storage high-water-mark–below 16MB line
DFHSTOR	116	User-storage high-water-mark–below 16MB line (CDSA)
DFHSTOR	117	User-storage get-count–below 16MB line (CDSA)
DFHSTOR	118	User-storage-occupancy (bytes-ms)–below 16MB line (CDSA)
DFHSTOR	119	User-storage high-water-mark–above 16MB line (ECDSA)
DFHSTOR	120	User-storage get-count–above 16MB line (ECDSA)
DFHSTOR	121	User-storage-occupancy (bytes-ms)–above 16MB line (ECDSA)
DFHSTOR	122	Program-storage high-water-mark (ERDSA)
DFHSTOR	139	Program-storage high-water-mark–above 16MB line
DFHSTOR	142	Program-storage high-water-mark (ECDSA)
DFHSTOR	143	Program-storage high-water-mark (CDSA)
DFHSTOR	144	Shared-storage getmain-count–below 16MB (CDSA and SDSA)
DFHSTOR	145	Shared-storage bytes getmained–below 16MB (CDSA and SDSA)

Table 50. This table shows the data groups and fields that can be excluded/included (continued)

Group Name	Field Id	Description
DFHSTOR	146	Shared-storage freemained–below 16MB (CDSA and SDSA)
DFHSTOR	147	Shared-storage getmain-count–above 16MB (ECDSA and ESDSA)
DFHSTOR	148	Shared-storage bytes getmained–above 16MB (ECDSA and ESDSA)
DFHSTOR	149	Shared-storage bytes freemained–above 16MB (ECDSA and ESDSA)
DFHSTOR	160	Program-storage high-water-mark (SDSA)
DFHSTOR	161	Program-storage high-water-mark (ESDSA)
DFHSTOR	162	Program-storage high-water-mark (RDSA)
DFHSYNC	60	Sync point count
DFHSYNC	173	Sync point elapsed time
DFHSYNC	177	CFDT server syncpoint wait time
DFHSYNC	196	Syncpoint delay time
DFHTASK	7	User-task dispatch time
DFHTASK	8	User-task CPU time
DFHTASK	14	User-task suspend time
DFHTASK	31	Task number
DFHTASK	59	IC put/initiate count
DFHTASK	64	Error flag field
DFHTASK	66	IC total count
DFHTASK	82	Transaction group id
DFHTASK	97	Network name of the originating terminal or system
DFHTASK	98	Unit-of-work id on the originating system
DFHTASK	102	User-task wait-for-dispatch time
DFHTASK	109	Transaction priority
DFHTASK	123	Task global ENQ delay time
DFHTASK	124	3270 Bridge transaction id
DFHTASK	125	First dispatch delay time
DFHTASK	126	First dispatch delay time due to TRANCLASS
DFHTASK	127	First dispatch delay due to MXT
DFHTASK	128	Lock manager delay time
DFHTASK	129	Task ENQ delay time
DFHTASK	132	Recovery manager unit-of-work id
DFHTASK	163	Transaction facility name
DFHTASK	164	Transaction flags
DFHTASK	170	Resource Manager Interface–elapsed time
DFHTASK	171	Resource Manager Interface–suspend time
DFHTASK	181	EXEC CICS WAIT EXTERNAL wait time
DFHTASK	182	EXEC CICS WAITCICS and WAIT EVENT wait time
DFHTASK	183	Interval Control delay time
DFHTASK	184	“Dispatch Wait” wait time
DFHTASK	190	RRMS/MVS unit-of-recovery id (URID)
DFHTASK	191	RRMS/MVS wait time
DFHTASK	195	CICS BTS run process/activity synchronous wait time
DFHTASK	248	CICS TCB change modes
DFHTASK	249	User-task QR TCB wait-for-dispatch time
DFHTASK	250	CICS MAXOPENTCBS delay time
DFHTASK	251	CICS TCB attach count

Table 50. This table shows the data groups and fields that can be excluded/included (continued)

	Group Name	Field Id	Description
#	DFHTASK	253	CICS JVM elapsed time
#	DFHTASK	254	CICS JVM suspend time
|	DFHTASK	255	User-task QR TCB dispatch time
|	DFHTASK	256	User-task QR TCB CPU Time
|	DFHTASK	257	User-task MS TCB dispatch time
|	DFHTASK	258	User-task MS TCB CPU Time
|	DFHTASK	259	User-task L8 TCB CPU Time
|	DFHTASK	260	User-task J8 TCB CPU Time
|	DFHTASK	261	User-task S8 TCB CPU Time
	DFHTEMP	11	TS I/O wait time
	DFHTEMP	44	TS get count
	DFHTEMP	46	TS put auxiliary count
	DFHTEMP	47	TS put main count
	DFHTEMP	92	TS total count
	DFHTEMP	178	Shared TS I/O wait time
	DFHTERM	9	TC I/O wait time
	DFHTERM	34	TC principal facility input messages
	DFHTERM	35	TC principal facility output messages
	DFHTERM	67	TC alternate facility input messages
	DFHTERM	68	TC alternate facility output messages
	DFHTERM	69	TC allocate count
	DFHTERM	83	TC principal facility CHARS input
	DFHTERM	84	TC principal facility CHARS output
	DFHTERM	85	TC alternate facility CHARS input
	DFHTERM	86	TC alternate facility CHARS output
	DFHTERM	100	IR I/O wait time
	DFHTERM	111	VTAM terminal LU name
	DFHTERM	133	TC I/O wait time - LU6.1
	DFHTERM	134	TC I/O wait time - LU6.2
	DFHTERM	135	TC alternate facility input messages - LU6.2
	DFHTERM	136	TC alternate facility output messages - LU6.2
	DFHTERM	137	TC alternate facility CHARS input - LU6.2
	DFHTERM	138	TC alternate facility CHARS output - LU6.2
	DFHTERM	165	Terminal information
	DFHTERM	169	Terminal session connection name
|	DFHWEBB	231	WEB receive request count
|	DFHWEBB	232	WEB characters received
|	DFHWEBB	233	WEB send request count
|	DFHWEBB	234	WEB character sent
|	DFHWEBB	235	WEB total request count
|	DFHWEBB	236	WEB repository read request count
|	DFHWEBB	237	WEB repository write request count

INCLUDE=(m1[,...])

Code this to enable one or more CICS fields to be reported by the monitoring facility. By default, all documented performance class fields are reported, so this operand is relevant only if you code the EXCLUDE operand in the same macro.

The fields that are eligible to be coded for inclusion on this operand are the same as those that are eligible for exclusion. (See the description of the EXCLUDE operand.) Each field has a numeric field identifier associated with it. To include a field, you code *m1* as the numeric identifier of the field. You can code multiple numeric identifiers.

Note: Do not code leading zeros on numeric identifiers. Do not code numeric identifiers of fields that are ineligible for exclusion, and that are therefore included by default

.

The EXCLUDE operand is always honored first. The INCLUDE operand, if coded, then overrides some of its effects. For example, coding:

```
EXCLUDE=DFHFILE,
INCLUDE=(37,93)
```

would secure the collection and reporting of file control PUTs and the total number of file control requests, while file control browse count and other file control fields would be excluded.

If you want to exclude the majority of fields, but include a few, you can code, for example:

```
EXCLUDE=ALL,
INCLUDE=(DFHTERM,97,98)
```

This is more convenient than coding individually all the fields you want to exclude.

CICSTS13.CICS.SDFHSAMP provides the following sample monitoring control tables:

- For terminal-owning region (TOR)—DFHMCTT$
- For application-owning region (AOR)—DFHMCTA$
- For an application-owning region (AOR) with DBCTL—DFHMCTD$
- For file-owning region (FOR)—DFHMCTF$

These samples show how to use the EXCLUDE and INCLUDE operands to reduce the size of the performance class record in order to reduce the volume of data written by CICS to SMF.

DFHMCT example

Figure 51 illustrates the coding to create a monitoring control table (MCT) for two user event monitoring points (EMPs).

```
DFHMCT   TYPE=INITIAL
DFHMCT   TYPE=EMP,                               *
         ID=180,                                 *
         CLASS=PERFORM,                          *
         PERFORM=(SCLOCK(1),ADDCNT(2,1)),        *
         ACCOUNT=ADDCNT(1,1)
DFHMCT   TYPE=EMP,                               *
         ID=181,                                 *
         CLASS=PERFORM,                          *
         PERFORM=PCLOCK(1)
DFHMCT   TYPE=FINAL
         END
```

Figure 51. Monitoring control table—example

PLT—program list table

A program list table (PLT) contains a list of related programs. You may wish to generate several PLTs for one or more of the following reasons:

- To specify a list of programs that you wish to be executed in the second and/or third initialization stages of CICS startup. For more detail about the initialization stages, see the *CICS Recovery and Restart Guide*. For programming information about restrictions on using programs in the initialization stages, see the *CICS Customization Guide*. The selected list should be specified at initialization time by the PLTPI=*xx* system initialization parameter, where *xx* is the suffix of the PLT that contains the required list of programs.

  For convenience, the list of programs selected for execution during initialization is referred to as the 'PLTPI' list.
- To specify a list of programs that you wish to be executed during the first and/or second quiesce stages of controlled shutdown. The selected list should be specified at initialization time by the PLTSD=*xx* system initialization parameter, where *xx* is the suffix of the PLT that contains the required list of programs.

  The PLT specified in the PLTSD system initialization parameter can be overridden at shutdown time by the PLT option in the CEMT PERFORM SHUTDOWN command.

  The shutdown PLT is normally loaded as CICS is being shutdown. However, it is possible to use the same PLT for both initialization and shutdown, and under these circumstances the PLT is loaded during initialization and CICS does not need to reload it during shutdown. If this is the case and the PLT is updated while CICS is operational, a CEMT SET PROGRAM NEWCOPY command must be issued for the PLT to ensure that the updated version is used when CICS is shutdown.

  For convenience, the list of programs selected for execution during shutdown is referred to as the 'PLTSD' list.
- To specify a list of programs that you wish to have enabled or disabled as a group by a master terminal ENABLE or DISABLE command. This use of PLTs means that a master terminal operator can enable or disable a set of programs with just one command, instead of using a separate command for each program.

Any number of PLTs can be generated for the above purposes, provided that:

1. Each PLT has a unique suffix
2. Each program named in a PLT either has a program resource definition entry in the CSD file, or is capable of being autoinstalled (that is, the appropriate
# system initialization parameters have been specified for program autoinstall).

Note: PLTs should not be defined as programs in the CSD.

PLTs must be placed in the DFHRPL library. However, CICS scans the LPA for phase 1 PLTPI programs if they are not already installed.

Elements of DFHPLT

The following macros are available to define the PLT entries:

- Control section—DFHPLT TYPE=INITIAL
- Entries in program list table—DFHPLT TYPE=ENTRY
- End of Program List Table—DFHPLT TYPE=FINAL (Described on page 371)

Control section—DFHPLT TYPE=INITIAL

Table 51. DFHPLT TYPE=INITIAL

DFHPLT	TYPE=INITIAL [,SUFFIX=xx]

For general information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see “TYPE=INITIAL (control section)” on page 370.

TYPE=INITIAL
Generates the PLT control section.

Note that the CSD file must define a program entry for each PLT generated.

SUFFIX=xx
Code this with the suffix character(s) that uniquely identify this particular table.

Note: The PLT suffix is referenced by:
- CEMT {INQUIRE|SET} PROGRAM CLASS(*suffix*)
- CEMT or EXEC CICS PERFORM SHUTDOWN PLT(*suffix*)
- System initialization parameters PLTPI and PLTSD keywords.

Entries in program list table—DFHPLT TYPE=ENTRY

Entries are specified in the PLT as follows:

Table 52. DFHPLT TYPE=ENTRY, PROGRAM=program

DFHPLT	TYPE=ENTRY ,PROGRAM=(program [,program,...])

TYPE=ENTRY
Indicates that one or more program names are to be listed in this table.

Note: As shown below, a TYPE=ENTRY macro is also needed to specify the PROGRAM=DFHDELIM entry.

PROGRAM=program
Code this with a program name of up to eight characters. Each program must either have a definition in the CSD file or must be capable of being autoinstalled (that is, the appropriate system initialization parameters must be specified for program autoinstall). Undefined programs before the DFHDELIM statement are system autoinstalled.

For PLTPI and PLTSD lists, only **initial** programs should be named: other programs that are linked to by initial programs should not be listed (but must be defined or be capable of being autoinstalled). For programming information about restrictions on using PLT programs during initialization, see the *CICS Customization Guide*.

Table 53. DFHPLT TYPE=ENTRY, PROGRAM=DFHDELIM

```
DFHPLT            TYPE=ENTRY
                  ,PROGRAM=DFHDELIM
```

PROGRAM=DFHDELIM

Code this to delimit the programs to run in the first or second passes of PLTPI or PLTSD. The DFHDELIM entry is not a program—it serves as a delimiter only.

Note that:

- Programs listed **before** the PROGRAM=DFHDELIM entry in a PLTPI are executed during the **second** stage of initialization. These are to enable user exit programs needed during recovery. Define the user exit programs in the CSD file, otherwise CICS may not be able to access them after CICS initialization is complete, for example in EXEC CICS DISABLE commands. However, note that the properties defined by RDO have no effect during the second stage of initialization.
- Programs listed after the PROGRAM=DFHDELIM entry in a PLTPI are executed during the third stage of initialization. If these programs are used to enable user exits, the user exit programs must also be defined in the CSD file or must be capable of being autoinstalled.
- Programs listed before the PROGRAM=DFHDELIM entry in a PLTSD are executed during the first quiesce stage of shutdown.
- Programs listed after the PROGRAM=DFHDELIM entry in a PLTSD are executed during the second quiesce stage of shutdown.

Second stage initialization and second stage quiesce PLT programs do not require program resource definitions. If they are not defined, they are system autoinstalled (irrespective of the program autoinstall system initialization parameters). This means that the autoinstall exit is not called to allow the definition to be modified. The programs are defined with the following attributes:

```
LANGUAGE(ASSEMBLER)   STATUS(ENABLED)   CEDF(NO)
DATALOCATION(BELOW)   EXECKEY(CICS)
EXECUTIONSET(FULLAPI)
```

Third stage initialization and first stage quiesce PLT programs can be defined using program autoinstall, depending upon the program autoinstall system initialization parameters. See “Chapter 11. Autoinstall for other resources” on page 145 for further information. If program autoinstall is not used, these programs must have program resource definitions in the CSD file.

DFHPLT example

Figure 52 and Figure 53 illustrate the coding required to generate a PLT.

```
*
* LIST OF PROGRAMS TO BE EXECUTED SEQUENTIALLY DURING SYSTEM
* INITIALIZATION.
* REQUIRED SYSTEM INITIALIZATION PARAMETER: PLTPI=I1
*
  DFHPLT TYPE=INITIAL,SUFFIX=I1
*
*  The following programs are run in the first pass of PLTPI
*
  DFHPLT TYPE=ENTRY,PROGRAM=TRAQA  EXECUTED DURING 2ND INIT. PHASE
  DFHPLT TYPE=ENTRY,PROGRAM=TRAQB  (PROGRAMS SHOULD ALSO BE DEFINED
  DFHPLT TYPE=ENTRY,PROGRAM=TRAQC  BY RDO)
*
  DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
*
*
*  The following programs are run in the second pass of PLTPI
*
  DFHPLT TYPE=ENTRY,PROGRAM=TRASA  EXECUTED DURING 3RD INIT. PHASE
  DFHPLT TYPE=ENTRY,PROGRAM=TRASB  (PROGRAMS MUST ALSO BE DEFINED
  DFHPLT TYPE=ENTRY,PROGRAM=TRASC  BY RDO)
  DFHPLT TYPE=FINAL
*
  END
```

Figure 52. PLTPI program list table—example

```
*
*
* LIST OF PROGRAMS TO BE EXECUTED SEQUENTIALLY DURING SYSTEM
* TERMINATION
* REQUIRED SYSTEM INITIALIZATION PARAMETER: PLTSD=T1
*
  DFHPLT TYPE=INITIAL,SUFFIX=T1
*
*  The following programs are run in the 1st pass of PLTSD
*
*
  DFHPLT TYPE=ENTRY,PROGRAM=TRARA  EXECUTED DURING 1st QUIESCE PHASE
  DFHPLT TYPE=ENTRY,PROGRAM=TRARB  (PROGRAMS MUST ALSO BE DEFINED
  DFHPLT TYPE=ENTRY,PROGRAM=TRARC  BY RDO)
*
  DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
*
*
*  The following programs are run in the 2nd pass of PLTSD
*
  DFHPLT TYPE=ENTRY,PROGRAM=TRAFA  EXECUTED DURING 2nd QUIESCE PHASE
  DFHPLT TYPE=ENTRY,PROGRAM=TRAFB  (PROGRAMS MUST ALSO BE DEFINED
*                                  BY RDO)
  DFHPLT TYPE=FINAL
*
  END
```

Figure 53. PLTSD program list table—example

RCT—CICS DB2 resource control table

The resource control table (RCT) controls the CICS DB2 interface.

| **RCTs must be migrated to online resource definitions in CICS Transaction
| Server for OS/390 Release 3. The macro is only supported for migration
| purposes.**

| The RCT can be defined using the DSNCRCT macro to produce a macro table.
| Your RCT must be assembled using the DSNCRCT macro shipped with this
| release, in the SDFHMAC library.

This chapter describes the syntax of the DSNCRCT macro and guides you through its coding.

Code the RCT as follows before the assembly.
1. The TYPE=INIT macro must be specified first, (unless preceded by TYPE=GROUP).
2. If the TYPE=COMD macro is specified, it must be second.
3. If TYPE=POOL is specified, it must be after TYPE=INIT and TYPE=COMD (if there is a COMD) and before TYPE=ENTRY.
4. The TYPE=ENTRY macro can be specified an indefinite number of times to identify specific transactions.
5. The TYPE=FINAL macro is specified last, causing the RCT to be generated.

For example:

```
DSNCRCT TYPE=INIT,
DSNCRCT TYPE=COMD,[optional]
DSNCRCT TYPE=POOL,[optional]
DSNCRCT TYPE=ENTRY,[optional]
⋮
DSNCRCT TYPE=ENTRY,[optional]
DSNCRCT TYPE=FINAL
END     [assembler end statement required]
```

The RCT must be link-edited into a library that is in the DFHRPL concatenation. (In releases of CICS before CICS Transaction Server for OS/390 Release 3, the RCT had to reside in a library in the STEPLIB concatenation).

Control section—DSNCRCT TYPE=INIT

The TYPE=INIT macro allows you to define the information required for CICS to establish its first connection to DB2 and to specify the default options for other types of the macro.

DSNCRCT

```
>>──DSNCRCT──TYPE=INIT──┬─────────────────────────┬──────────────────────────────>
                        │         ┌─HIGH─┐        │
                        └─DPMODI=─┼─EQ───┼────────┘
                                  └─LOW──┘

>──┬────────────────────────────┬──┬─────────────────────────┬────────────────────>
   └─ERRDEST=(dest1,dest2,dest3)┘  │          ┌─AEY9──┐      │
                                   └─PCTEROP=─┼─N906D─┼──────┘
                                              └─N906──┘

>──┬─────────────────────┬──┬─────────────────────────────┬───────────────────────>
   └─PLANI=plan-name─────┘  └─PLNPGMI=default-exit-name───┘

>──┬──────────────────────────┬──┬────────────────────────┬──┬──────────────────┬──>
   └─PURGEC=minutes, seconds──┘  └─RDONAME=DB2CONN-name───┘  │        ┌─YES─┐   │
                                                             └─ROLBI=─┴─NO──┴───┘

>──┬──────────────────────────┬──┬─────────────────────────────┬──────────────────>
   └─SHDDEST=destination──────┘  └─SIGNID=authorization-id────┘

>──┬──────────────────────────┬──┬─────────────────────┬──┬────────────────┬──────>
   │          ┌─ABEND───┐     │  │         ┌─YES──┐    │  └─SUBID=db2id────┘
   └─STANDBY=─┴─SQLCODE─┴─────┘  └─STRTWT=─┼─NO───┼────┘
                                           └─AUTO─┘

>──┬──────────────────┬──┬──────────────────────┬──┬──────────────────┬──────────>
   └─SUFFIX=suffix-id─┘  └─THRDMAX=integer──────┘  │         ┌─NO──┐  │
                                                   └─TOKENI=─┴─YES─┴──┘

>──┬──────────────────────┬──┬──────────────────────┬────────────────────────────><
   │         ┌─YES─┐      │  │         ┌─YES──┐     │
   └─TXIDSO=─┴─NO──┴──────┘  └─TWAITI=─┼─NO───┼─────┘
                                       └─POOL─┘
```

DPMODI=HIGH|EQ|LOW
Specifies a default for the DPMODE parameter in other TYPEs of this macro.

HIGH
Specifies that subtasks can attain a higher priority than the CICS main task from which the subtask was generated. Use this option for high priority and high volume transactions.

EQ
Specifies that CICS must allow for subtasks to attain equal priority.

LOW
Specifies that subtasks have a lower priority than the CICS main task priority.

ERRDEST=(dest1,dest2,dest3)
Specifies up to three CICS transient data destinations to receive unsolicited messages. For *dest1,dest2,dest3,* substitute up to three valid transient data destinations.

An asterisk can be specified as a destination. The asterisk acts as a place holder and allows later specification of a destination by the DSNC MODIFY DESTINATION command.

PCTEROP=AEY9|N906D|N906
Specifies the type of processing that is to occur following a create thread error.

The error processing occurs after the SQLCA's SQLCODE field has been updated to reflect the reason for the create thread failure. The PCTEROP parameter allows a user to specify whether a transaction dump is taken, and whether the DSNCSQL RMI associated with the transaction is disabled. This parameter can be used to allow a transaction to continue processing if a create thread error occurs. A transaction that continues after a create thread error must take corrective action to allow a new thread to be created. A SYNCPOINT ROLLBACK command must be part of the corrective action taken by the transaction before it can continue to issue SQL requests.

AEY9
The transaction abends with abend code AD2S, AD2T, or AD2U, depending on the type of error. The transaction must be terminated and reinitialized before it is allowed to issue another SQL request.

N906D
Specifies that a transaction dump is to be taken and the DSNCSQL RMI associated with the transaction is *not* to be disabled. The transaction receives a -906 SQLCODE if another SQL is issued, unless the transaction issues SYNCPOINT ROLLBACK. SYNCPOINT without the ROLLBACK option results in an ASP7 abend.

N906
Specifies that a transaction dump is *not* to be taken and the DSNCSQL RMI
associated with the transaction is *not* to be disabled. The transaction
receives a -906 SQLCODE if another SQL request is issued, unless the
transaction issues a SYNCPOINT ROLLBACK. SYNCPOINT without the
ROLLBACK option results in an ASP7 abend.

PLANI=*plan-name*
Specifies the default plan name for any entry in the RCT that does not use dynamic plan selection. The ***plan-name*** can have 1-8 characters. Without the PLANI option, the plan name for an entry in the RCT is:

- The value for PLAN= in the TYPE=ENTRY macro
- The value for TXID= in the TYPE=ENTRY macro if PLAN= is not specified

With the PLANI option, the plan name for an entry in the RCT is:

- The value for PLAN= in the TYPE=ENTRY macro
- The value for PLANI= in the TYPE=INIT macro if PLAN= is not specified
- The value for TXID= in the TYPE=ENTRY macro if neither PLAN= nor PLANI= is specified

The PLANI option has no effect on entries that use dynamic plan selection.

PLNPGMI=*default-exit-name*
Specifies the name of the default dynamic plan exit. If one of the entries has PLNEXIT=YES, but does not supply a value for PLNPGME, this parameter is used as the exit program name for that entry.

Default: DSNCUEXT

PURGEC=*minutes,seconds*
specifies the length of the protected thread purge cycle. The maximum value for PURGEC is (59,59). The minimum is (0,30).

An unprotected thread is terminated as soon as the transaction ends (at SYNCPOINT or EOT). A protected thread is terminated after two purge cycles, which are 30 seconds by default. Normally, a protected thread remains connected for 30-60 seconds after the transaction ends.

You can use PURGEC to modify the 'normal purge cycle'. The purge cycle is 5 minutes long when the attachment starts and then PURGEC for the remaining time that the attachment facility operates. For example, if you specify PURGEC=(0,40), protected threads are normally purged 40-80 seconds after the transaction ends.

RDONAME=*DB2CONN-name*
specifies the name to be used for the DB2CONN when migrating the RCT to the CSD.

ROLBI=YES|NO
Specifies a default for the ROLBE parameter in other TYPEs of the DSNCRCT macro. The specification of this parameter determines the disposition of transaction entries in the event a transaction is selected by DB2 as victim in a deadlock resolution.

YES
Specifies that the attachment facility is to issue a syncpoint rollback before returning control to the application. A SQL return code of -911 is returned to the program. Specifying YES provides compatibility with SQL/DS.

NO
Specifies that the attachment facility is not to initiate a rollback for this transaction. A SQL return code of -913 is returned to the application. It is the responsibility of the application to initiate the rollback.

SHDDEST=*destination*
Specifies a transient data destination to receive the statistical report (the same report that is displayed with the DSNC DISP STAT command) during shutdown of the attachment facility. For *destination*, substitute a valid transient data destination.

It might be useful to direct this transient output data to a destination in another partition that is specified as a JES SYSOUT file.

SIGNID=*authorization ID*
Specifies the authorization ID to be used by the CICS attachment facility when signing on to DB2. For *authorization ID*, substitute a character string of up to eight characters. The name can be up to eight characters in length.

The default is the APPLID of the CICS system. This name is used when indicated when AUTH=SIGNID is set on the TYPE=ENTRY or TYPE=POOL forms of the macro. For a description of the AUTH parameter, see page 428 DSNCRCT TYPE=ENTRY. When it is used, the name specified here must be authorized to the resources being accessed.

STANDBY=SQLCODE|ABEND
Specifies the action to be taken by the attachment facility during the startup process if DB2 is not active.

SQLCODE
Only valid if STRTWT=AUTO or YES is specified. If an application issues a SQL statement while the attachment facility is standing by, SQLCODE -923 is issued instead of abend AEY9.

ABEND
Specifies CICS applications using DB2 fail with abend AEY9 issued by CICS when the attachment is not started.

STRTWT=AUTO | YES | NO
Specifies the action to be taken by the attachment facility during the startup process if DB2 is not active.

AUTO
Specifies automatic restart of the attachment facility if DB2 stops or abends, then restarts. The starting procedures are the same as for YES. If DB2 stops or abends while the attachment facility is up, a message is issued stating the subsystem is not active. The attachment facility goes to standby state and only terminates after the command DSNC STOP is issued or a fatal error is encountered.

YES
Directs the attachment facility to wait for DB2 to start and complete the connection. A CICS task waits to be posted by DB2 when DB2 becomes available. At that time, the initialization of the CICS attach is complete. However, the attachment facility can be terminated by the DSNC STOP command while it is waiting for DB2.

The response messages from the attachment are sent to the transient data destination queue specified in the ERRDEST parameter of the RCT.

NO
Directs the attachment facility to terminate the connection process immediately if DB2 is not already active.

SUBID=*DB2-ID*
Specifies the name of the DB2 subsystem that the attachment facility is to connect with CICS. For *DB2 ID*, substitute a character string of up to four characters.

SUFFIX=*XX*
Specifies the one or two characters that are concatenated with DFHRCT to
| create the name of the resource control table. Acceptable characters are A-Z 0-9
| $ @ and #. Do not use NO, DY or BA. For more information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see "TYPE=INITIAL (control section)" on page 370.

THRDMAX=*integer*
Specifies the maximum number of subtasks (TCBs) that can be identified to DB2. The default is 12, the minimum value is 4. The number controls the total number of threads for each region. For that reason, the recommended value for THRDMAX is the sum of all values on the THRDA parameters (COMD, ENTRY, and POOL threads) + 3. However, the value you specify for THRDMAX can be less than the sum of all values on the THRDA parameter.

TOKENI=NO | YES
Specifies the default TOKENE if TOKENE is not specified on the TYPE=ENTRY statement. For more information about TOKENE, see page 432.

TXIDSO=YES | NO
Specifies whether you want to suppress some sign-ons during thread reuse, and thereby avoid extraneous accounting information. The TXIDSO option affects only pool threads and those RCT entry threads with multiple transaction IDs in one entry (for example, TXID=(XC05,XC07). The attach

checks for thread reuse only within an entry. TXIDSO has no effect on transactions that specify TOKENE=YES. If the plan name changes, the thread is terminated and recreated.

YES
Specifies the following rules for thread reuse:
- A new transaction can reuse an existing thread without a sign-on when:
  – The authorization ID and transaction ID are the same as the last transaction that used the thread, and TOKENE is set to NO.
- A new transaction must sign-on before reusing an existing thread when any of these conditions exist:
  – The authorization ID is different from the authorization ID that last used the thread.
  – TOKENE is YES.
  – The transaction ID has changed.

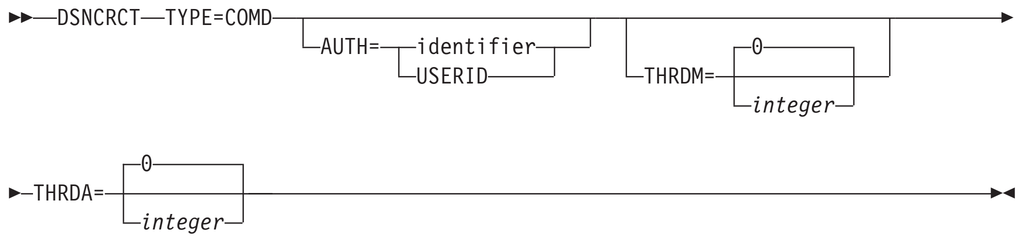
NO
Specifies the following rules for thread reuse:
- A new transaction can reuse an existing thread without a sign-on when:
  – The authorization ID is the same as the last transaction that used the thread and TOKENE is set to NO.
- A new transaction must sign-on before reusing an existing thread when either of these conditions exist:
  – The authorization ID is different from the authorization ID that last used the thread.
  – TOKENE is YES.

TWAITI=YES|NO|POOL
Specifies the default value (YES, NO, or POOL) that is to be created for the TWAIT parameter on other types of the macro.

DB2 commands—DSNCRCT TYPE=COMD

The TYPE=COMD form of the macro is used to describe DB2 command threads used for processing DB2 commands only. Requests for a DB2 command thread are diverted to the pool if a DB2 command thread is not available.

If you code the DSNCRCT TYPE=COMD macro, you must place it immediately following the TYPE=INIT macro and before the TYPE=POOL macro.



Description: The TYPE=COMD form of DSNCRCT is optional; if you do not code it, the RCT automatically generates a TYPE=COMD entry with the following parameters:

```
DSNCRCT TYPE=COMD,
      AUTH=USERID,
      THRDM=1,
      THRDA=1,
```

For a description of the TYPE=COMD parameters, see the same parameters in the description of DSNCRCT TYPE=ENTRY on page 428

One reason you might want to change this generated default setting is to change the AUTH parameter to reflect the need to give users different responsibilities in operating the DB2 subsystem.

Thread sharing—DSNCRCT TYPE=POOL

The TYPE=POOL form of the macro defines threads that can be shared by some or all of the CICS transactions. These threads are allocated to transactions only for a CICS unit of work. They can be considered short-term threads, and they can be used by any RCT entry that is specified to overflow to the pool.

The TYPE=POOL form of the macro must be coded before the TYPE=ENTRY form.

```
►►──DSNCRCT──TYPE=POOL──┬────────────────────────┬─┬─────────────────────┬──────────────────────►
                        └─AUTH=──┬─identifier─┬──┘ │          ┌─HIGH─┐   │
                                 └─USERID─────┘    └─DPMODE=──┼─EQ───┼───┘
                                                              └─LOW──┘

►──┬──────────────────────────────────────────────┬─┬───────────────────┬─┬───────────────────┬──►
   ├─PLAN=plan-name───────────────────────────────┤ │         ┌─YES─┐   │ │         ┌─3───────┐ │
   │          ┌─NO─PLAN=plan-name──────────────┐  │ └─ROLBE=──┴─NO──┴───┘ └─THRDA=──┴─integer─┴─┘
   │          │                  ┌─DSNCUEXT──────────┐ │  │
   └─PLNEXIT=─┴─YES─PLNPGME=─────┴─exit-program-name─┴─┴──┘

►──┬─────────────────────┬─┬────────────────────┬─┬──────────────────┬─────────────────────────►◄
   │        ┌─3───────┐  │ └─TOKENE=──┬─NO──┬───┘ │        ┌─YES─┐   │
   └─THRDM=─┴─integer─┴──┘            └─YES─┘     └─TWAIT=─┴─NO──┴───┘
```

Description:

DSNCRCT TYPE=POOL
: This is the name of the macro. It must be coded exactly as it appears here.

  If the TYPE=POOL form of the macro is not coded, the following default values are assumed:

  ```
  DSNCRCT TYPE=POOL,PLAN=DEFAULT,
        AUTH=USERID
        THRDM=3,
        THRDA=3,
        TWAIT=YES
  ```

  This default assumes there is a plan named DEFAULT.

The rest of the parameters on the TYPE=POOL form of the macro are basically the same as those on the TYPE=ENTRY form of this macro. The only difference for the parameter specifications on TYPE=POOL is:

- The default for PLAN is DEFAULT.

For a description of the parameters, see DSNCRCT TYPE=ENTRY on page 428.

Notes:

1. The pool provides default processing parameters for transactions that do not have an associated RCT entry. Such transactions are processed according to the specifications of the TYPE=POOL macro; transactions specified with the

TYPE=ENTRY form of the macro are processed according to specifications of that macro, even if they overflow to the pool.

2. The THRDM and THRDA parameters must be specified as greater than or equal to 3 for the TYPE=POOL form of the macro.
3. All transactions that do not have an RCT entry must be bound with the plan specified for the pool, unless dynamic plan allocation is specified for the pool.

Entries in resource control table—DSNCRCT TYPE=ENTRY

The TYPE=ENTRY form of the macro defines overrides to values set in the TYPE=POOL definition and provides dedicated threads to a transaction or group of transactions.

The plan name identifies the DB2 resource being accessed by a CICS application. A dynamic plan exit program is a way of allocating plans to CICS dynamically at execution time. For more information about dynamic plan exit programs, see the *CICS DB2 Guide*. In entry threads, all the transactions assigned to the same RCT entry use the plan name or dynamic plan allocation exit program specified for that entry.

Allocation deadlocks might occur if more than one thread is allowed for an RCT entry that uses a plan containing exclusive locking (table space locks). (See the THRDA and TWAIT parameters explanations for more information.)

```
►►──DSNCRCT──TYPE=ENTRY──┬──────────────────────────┬──┬───────────────────────┬──────────────────►
                         │       ┌─USERID─────┐     │  │          ┌─HIGH─┐     │
                         └─AUTH=─┴─identifier─┴─────┘  └─DPMODE=─┼─EQ───┼─────┘
                                                                 └─LOW──┘

►──┬──────────────────────────────────────────────────┬──┬─────────────────┬──┬─────────────────────┬──►
   ├─PLAN=plan-name───────────────────────────────────┤  │        ┌─YES─┐  │  │        ┌─0───────┐  │
   │           ┌─NO─PLAN=plan-name──────────────────┐ │  └─ROLBE=─┴─NO──┴──┘  └─THRDA=─┴─integer─┴──┘
   │           │                  ┌─DSNCUEXT──────────┐ │ │
   ├─PLNEXIT=──┴─YES─PLNPGME=─────┴─exit-program-name─┴─┘ ┤
   └─┬─────────────────────┬──────────────────────────────┘
     └─RDONAME=entryname───┘

►──┬─────────────────────┬──┬─────────────────────┬──┬──────────────────┬──┬──────────────────┬──►
   │        ┌─0───────┐  │  │        ┌─0───────┐  │  └─TOKENE=─┬─NO──┬──┘  │        ┌─YES──┐  │
   └─THRDM=─┴─integer─┴──┘  └─THRDS=─┴─integer─┴──┘            └─YES─┘     └─TWAIT=─┼─NO───┼──┘
                                                                                    └─POOL─┘

►──┬────────────────────────────────┬──────────────────────────────────────────────────────────►◄
   │       ┌─,──────────────┐       │
   │       ▼                │       │
   └─TXID=(───transaction-id─┴───)──┘
```

AUTH=*identifier*
: Specifies an explicit character string (authorization ID) or directs the attachment facility to use names associated with the CICS transaction. Permissible values for *identifier* are:

character-string
: Specifies a character string that is used as a DB2 authorization ID. For *character-string*, substitute a character string of no more than eight characters, except for the reserved words USERID, USER, TERM, TXID, GROUP, and SIGNID.

GROUP
Specifies the RACF sign-on user ID (1 to 8-character USERID) and the connected group name as the authorization ID. The following table shows how these two values are interpreted by DB2.

IDs passed to DB2	How DB2 interprets values
CICS sign-on user ID (USERID)	Represents the primary DB2 authorization ID.
RACF connected group name	If the RACF list of group options is not active, then DB2 uses the connected group name supplied by the CICS attachment facility as the secondary DB2 authorization ID. If the RACF list of group options is active, DB2 ignores the connected group name supplied by the CICS attachment facility, but the value appears in the DB2 list of secondary DB2 authorization IDs.

To use the GROUP option, the CICS system must have SEC=YES specified in the CICS system initialization table (SIT).

If no RACF group ID is available for this USERID, then an 8-character field of blanks is passed to DB2 as the group ID.

SIGNID
Specifies that the value specified for SIGNID on the TYPE=INIT macro is to be used as the authorization ID.

TERM
Specifies the terminal identification (four characters padded to eight) as an authorization ID. An authorization ID cannot be obtained in this manner if a terminal is not connected with the transaction.

If a transaction is started (using a CICS command) and has no terminal associated with it, the AUTH=TERM parameter should not be used.

TXID
Specifies the transaction ID (four characters padded to eight) as the authorization ID.

USER
Specifies the operator identification associated with the userid that is associated with the CICS transaction as the authorization ID (three characters padded to eight).

USERID
Specifies the 8-character USERID associated with the CICS transaction as the authorization ID.

When the sample sign-on exit DSN3@SGN is used with AUTH=USERID, the exit sends the user ID to DB2 as the primary authorization ID and the RACF group ID to DB2 as the secondary ID. When the sample sign-on exit is used, there is no difference between AUTH=USERID and AUTH=GROUP.

Default: USERID

DPMODE=HIGH|EQ|LOW
Specifies the priority of thread subtasks relative to CICS main task priority.

HIGH
Specifies that subtasks can attain a higher priority than the CICS main task from which the subtask was generated. Use this option for high priority and high volume transactions.

EQ
Specifies that CICS must allow for subtasks to attain equal priority.

LOW
Specifies that subtasks have a lower priority than the CICS main task priority.

PLNEXIT=NO|YES
Specifies whether a dynamic plan allocation exit program is invoked. For more information on dynamic plan allocation, see the *CICS DB2 Guide*.

NO
Specifies that this transaction ID entry does not use a plan exit. The CICS attachment facility obtains the plan name using the name specified on the option PLAN=. Specifying NO means you must code the option PLAN=***plan-name***.

YES
Specifies that this transaction ID can dynamically allocate the plan name when the first SQL statement is processed for the application program. This is accomplished by means of the dynamic plan exit specified on the option PLNPGME=. Specifying YES means you must NOT code the option PLAN=.

PLAN=*plan-name*
Specifies the plan ID to be associated with this transaction when it is different from the transaction ID. For *plan-name*, substitute a valid application plan name.

The PLAN ID specified with a TYPE=ENTRY macro is used even if the POOL provides the thread. The PLAN parameter is a required parameter when the TXID parameter is coded as a list of two or more transaction IDs.

If PLAN= is not specified and PLNEXIT=NO, TXID is the default plan if only one transaction is specified on the TXID keyword. See the description of the TXID parameter.

PLNPGME=*exit-program-name*
Specifies the name of the exit program this entry uses. The default is set by the PLNPGMI parameter on the TYPE=INIT statement. For information on how to write your own exit program, see the *CICS DB2 Guide*.

RDONAME=*entryname*
Specifies the name to be used for the DB2ENTRY (and DB2TRAN if one transid is present) when migrating this RCTE to the CSD.

ROLBE=YES|NO
Specifies a disposition for transactions defined by this entry if a transaction is selected by DB2 as the victim in a deadlock resolution. The specification of ROLBE overrides the specification of the ROLBI parameter on the TYPE=INIT macro.

YES
Specifies that the attachment facility is to issue a syncpoint rollback before returning control to the application. A SQL return code of -911 is returned to the program. Specifying YES provides compatibility with SQL/DS™.

NO
Specifies that the attachment facility is not to initiate a rollback for this transaction. A SQL return code of -913 is returned to the application.

THRDA=*integer*
Specifies the maximum number of threads that the attachment facility allows connected for this transaction, group, or pool before requests are either made to wait or are diverted to the pool. (See the description of the TWAIT parameter.)

For *integer*, substitute a value that does not exceed 99 or the THRDM value. The general restriction is (THRDS ≤ THRDA ≤ THRDM ≤ 99). When THRDA=0, TWAIT=YES or TWAIT=POOL causes all threads to be diverted to the pool. Forcing low-use transactions into the pool this way might save MVS ATTACH overhead if pool threads are available.

If a plan specified for an RCT entry has exclusive locking, set the value of the THRDA parameter to 1 and the value of the TWAIT parameter to YES to prevent allocation deadlock.

For information about using threads in conjunction with dynamic plan selection, see the *CICS DB2 Guide*.

Default: 0

THRDM=*integer*
Specifies the maximum number of threads the attachment facility is prepared to connect for this transaction group. For *integer*, substitute an integer value that does not exceed 99. The general restriction is (THRDS ≤ THRDA ≤ THRDM ≤ 99).

You can associate a CICS transaction with a plan name without allocating it to an entry thread by specifying no threads (THRDM=0) and overflow to the pool (TWAIT=POOL). The entry thread uses the plan name associated with its TYPE=ENTRY parameter when it overflows to the pool.

For information about using threads in conjunction with dynamic plan selection, see the *CICS DB2 Guide*.

Default: 0

THRDS=*integer*
specifies the maximum number of protected threads. For *integer*, substitute a value that does not exceed the THRDA value or 99. The general restriction is (THRDS ≤ THRDA ≤ THRDM ≤ 99).

A thread released by a transaction when no other work is queued can be protected. It will not be terminated immediately. A protected thread is terminated only after two complete purge cycles if it has not been reused in the meantime. The purge cycle is specified using the PURGEC parameter of the TYPE=INIT statement.

Threads are protected only while they are inactive. If a transaction reuses a protected thread, the thread becomes active and the current number of protected threads is decremented.

For information about using threads in conjunction with dynamic plan selection, see the *CICS DB2 Guide*.

Default: 0

TOKENE=NO|YES

Specifies whether the CICS attachment facility produces a DB2 accounting record for every CICS transaction, even those transactions that are reusing threads. It also specifies whether the CICS attachment facility passes the CICS LU6.2 (protected) token to DB2 for inclusion in the DB2 accounting trace records. You might receive more than one DB2 accounting record for a CICS transaction that has more than one DB2 unit of recovery, but you can correlate the CICS and DB2 records with the matching LU6.2 tokens.

Because CICS produces accounting records on a transaction basis, and DB2 produces accounting records on a thread basis, it can be difficult to correlate the two. This parameter gives you the option of producing a DB2 accounting record for each CICS transaction, even for transactions that are reusing threads.

If you do not specify YES or NO for TOKENE, then it assumes the value specified in TOKENI on the TYPE=INIT statement.

YES

Specifies that the CICS attachment facility requests that DB2 (using SIGNON) produce an accounting record after each transaction. It also indicates that the attachment facility passes the CICS LU6.2 token to DB2 for inclusion in the DB2 accounting trace records. Specifying YES makes it easier to correlate DB2 and CICS accounting and trace records.

Specifying YES slightly increases the overhead of a SQL request that reuses threads because of additional SIGNON activity. In a thread reuse situation, the transaction rate can degrade by no more than 5%. For additional information on the CICS task scope and DB2 thread scope, see See *Section 5*, in *Volume 2* of the *DB2 Administration Guide*.

NO

Specifies that the CICS attachment facility does not produce a DB2 accounting record during thread reuse. When TOKENE=NO is specified, it is more difficult to correlate DB2 and CICS accounting and trace records.

TWAIT=YES|NO|POOL

Specifying TWAIT overrides the value of the TWAITI parameter on the TYPE=INIT macro.

YES

Specifies that, if all threads are busy, a transaction must wait until one becomes available. A transaction can wait as long as CICS allows it to wait, generally until a thread becomes available.

The number of transactions waiting can be limited by using a TRANCLASS.

If TWAIT=YES is specified with THRDA=0, the attachment facility routes the transaction to the pool. Otherwise, a DB2 transaction could wait indefinitely for a thread.

An alternative to using TRANCLASS is to use TWAIT=POOL. The task picks up a pool thread rather than waiting for an entry thread to become available.

NO
Specifies that, if all threads are busy, a transaction is terminated with an abend. If NO is specified, or if TWAIT=NO has been specified on the TYPE=POOL macro, you should closely coordinate the number of threads specified with the MAXACTIVE number on the TRANCLASS being used. This helps to prevent abends when threads are unavailable.

POOL
Specifies that, if all threads are busy, a transaction must be diverted to use the pool of threads. If the pool is also busy, and NO has been specified for the TWAIT parameter on the TYPE=POOL form of the macro, a transaction is terminated with an abend. See the description of the TWAIT=NO parameter.

TXID=*(transaction-ID)*
Specify the transaction identification, or identifications for this entry. For *transaction ID*, substitute the transaction identifications found in the CSD transaction definition. The way you code this option depends on how many transactions you have and on whether:

- You have different plans for each transaction.
- You want to use dynamic plan allocation.
- You want separate statistics for each transaction.

Here are some examples:

1. If you have several transactions that use the same plan, you can code a list of transaction IDs, to be indexed to the same RCT entry. Code your entry as in this example:

   ```
   DSNCRCT TYPE=ENTRY,TXID=(TXI1,TXI2,TXIn),
           PLAN=(PLNA)
   ```

   You cannot code more than one plan per entry. To specify a different plan for each transaction, you must code a separate DSNCRCT entry for each plan, as follows:

   ```
   DSNCRCT TYPE=ENTRY,TXID=(TXI1),PLAN=(PLNA)
   DSNCRCT TYPE=ENTRY,TXID=(TXI2),PLAN=(PLNB)
   DSNCRCT TYPE=ENTRY,TXID=(TXIn),PLAN=(PLNC)
   ```

2. With dynamic plan selection, DB2 selects a plan based on the DBRM of the first SQL statement, or based on a user-defined exit routine for dynamic plan selection. You can use the PLNEXIT and PLNPGME parameters to specify a user-defined exit routine.

   To use dynamic plan selection for your transactions, code one or more transactions per entry, and optionally add pointers to the PLNEXIT and PLNPGME parameters, as follows:

   ```
   DSNCRCT TYPE=ENTRY,TXID=(TXI1,TXI2,TXIn),
           PLNEXIT=YES
   ```

   ```
   or
   ```

```
DSNCRCT TYPE=ENTRY,TXID=(TXI1),PLNEXIT=YES
DSNCRCT TYPE=ENTRY,TXID=(TXI2),PLNEXIT=YES
DSNCRCT TYPE=ENTRY,TXID=(TXI3),PLNEXIT=YES
```

3. If you want separate CICS attachment facility statistics for each transaction, you must code a separate entry for each transaction, as in this example:

```
DSNCRCT TYPE=ENTRY,TXID=(TXI1)
DSNCRCT TYPE=ENTRY,TXID=(TXI2)
DSNCRCT TYPE=ENTRY,TXID=(TXIn)
```

Migrating RCT definitions—DSNCRCT TYPE=GROUP

Use this macro to name the groups into which the RCT-equivalent DB2 resource definitions are put when you migrate the RCT to the CSD. This macro can appear as many times as required and at any point in the macro source. Each time it appears it defines the CSD group into which subsequent definitions will be put until the next DSNCRCT TYPE=GROUP macro occurs.

GROUP=name
Specifies a name, of 1 to 8-characters, of a group to be used when migrating the RCT to the CSD. All RCT entries following the DSNCRCT TYPE=GROUP statement, until the next TYPE=GROUP statement, can be migrated to the CSD and placed in a group with the specified name. If no TYPE=GROUP statement is used, all RCT entries are migrated to a group called RCTxx, where xx is the one or two character RCT suffix, specified on the TYPE=INIT statement. GROUP=*DEFAULT will be supported, as for other tables, meaning you take the default group RCTxx.

Last code section—DSNCRCT TYPE=FINAL

The TYPE=FINAL form of the macro is coded last and results in the generation of the resource control table.

```
►►──DSNCRCT──TYPE=FINAL──────────────────────────────────────────────────────────►◄
```

Description:

DSNCRCT TYPE=FINAL
This is the name of the macro. It must be coded exactly as it appears here. The TYPE=FINAL form of the macro has no other parameters.

RST—recoverable service table

Note on terminology: DBCTL refers to the CICS—IMS/ESA DBCTL (database control) interface.

The recoverable service table (RST) is used for CICS DBCTL XRF (extended recovery facility) support. It contains a description of the DBCTL configuration. On detection of a DBCTL failure, the **active** CICS system uses the RST together with the MVS subsystem VERIFY to determine the existence of a suitable alternative DBCTL subsystem. The **alternate** CICS system uses the RST to check for the presence of a DBCTL subsystem. For security reasons, the RST should be link-edited into a library authorized using APF. The RST is not loaded as part of the CICS nucleus.

DFHRST macro types

You can code the following macros in a recoverable service table:

- DFHRST TYPE=INITIAL establishes the control section.
- DFHRST TYPE=RSE specifies the start of a recoverable service element (RSE). An RSE consists of a (nonempty) set of identifiers of equivalent DBCTL subsystems, the CICS applids associated with these DBCTL subsystems, and the application identifiers of the CICS systems which use the DBCTL subsystems.
- DFHRST TYPE=SUBSYS specifies one of the DBCTL subsystems in an RSE.
- DFHRST TYPE=FINAL concludes the RST (see page 371).

The RST to be used by the system must be defined in the system initialization table by the following system initialization parameter:

Table 54. Defining an RST in the SIT

DFHSIT	...
	,RST={NO|xx}

Control section—DFHRST TYPE=INITIAL

The DFHRST TYPE=INITIAL macro establishes the recoverable service table control section.

Table 55. DFHRST TYPE=INITIAL

DFHRST	TYPE=INITIAL
	[,SUFFIX=xx]

For general information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see "TYPE=INITIAL (control section)" on page 370.

Recoverable service elements—DFHRST TYPE=RSE

The DFHRST TYPE=RSE macro contains information about a recoverable service element and the CICS systems expected to connect to the DBCTL subsystems within the recoverable service element.

Table 56. DFHRST TYPE=RSE

DFHRST	TYPE=RSE
	[,CTLAPPLS=(applid1,applid2,....)]

TYPE=RSE
Indicates the start of a set of identifiers of equivalent DBCTL systems. Equivalent DBCTL subsystems have the same database (DB) name.

CTLAPPLS=(applid1,applid2,....)
Specifies the application identifiers of the CICS systems that are authorized to restart the DBCTL subsystems in the RSE.

DBCTL subsystems—DFHRST TYPE=SUBSYS

The DFHRST TYPE=SUBSYS macro contains information about a specific DBCTL subsystem within an RSE. For each subsystem in an RSE there must be a DFHRST TYPE=SUBSYS macro.

Table 57. DFHRST TYPE=SUBSYS

DFHRST	TYPE=SUBSYS ,SYBSYSID=subsystem-identifier [,JOBNAME=(jobname1,jobname2,....)]

TYPE=SUBSYS
Defines one of the DBCTL subsystems in an RSE.

SUBSYSID=subsystem-identifier
Code this with the 4-character name of the DBCTL subsystem-identifier. This identifier must be unique within the RST.

JOBNAME=(jobname1,jobname2,....)
Code this with the **MVS JOBNAME(S)** associated with the DBCTL subsystem identified in the SUBSYSID parameter statement. This is a form of security check. If CICS needs to cancel the DBCTL subsystem, it is authorized to do so only if the appropriate MVS jobname associated with the DBCTL subsystem is one of those specified by the **JOBNAME** parameter.

DFHRST example

Figure 54 illustrates the coding required to generate an RST.

```
DFHRST   TYPE=INITIAL
         ,SUFFIX=K1
DFHRST   TYPE=RSE
         ,CTLAPPLS=(applid1,applid2,applid3)
DFHRST   TYPE=SUBSYS
         ,SUBSYSID=CTL1
         ,JOBNAME=(job1,job2,job3,job4)
DFHRST   TYPE=SUBSYS
         ,SUBSYSID=CTL2
         ,JOBNAME=(job5,job6,job7,job8)
DFHRST   TYPE=FINAL
END
```

Figure 54. Recoverable service table—example

SRT—system recovery table

The system recovery table (SRT) contains a list of codes for abends that CICS intercepts. After it intercepts one, CICS attempts to remain operational by causing the offending task to abend.

You can modify the default recovery action by writing your own recovery program. You do this by means of the XSRAB global user exit point within the System Recovery Program (SRP). (See the *CICS Customization Guide* for programming information about the XSRAB exit.)

Note that recovery is attempted only if a user task (**not** a system task) is in control at the time the abend occurs.

DFHSRT macro types

The following macros may be coded in a system recovery table:

- DFHSRT TYPE=INITIAL establishes the control section
- DFHSRT TYPE=SYSTEM | USER specifies the abend codes that are to be handled
- DFHSRT TYPE=FINAL concludes the SRT (see page 371)

Control section—DFHSRT TYPE=INITIAL

The DFHSRT TYPE=INITIAL macro generates the system recovery table control section.

Table 58. DFHSRT TYPE=INITIAL

DFHSRT	TYPE=INITIAL [,SUFFIX=xx]

For general information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see "TYPE=INITIAL (control section)" on page 370.

Abend codes—DFHSRT TYPE=SYSTEM|USER

Table 59. DFHSRT TYPE=SYSTEM|USER

DFHSRT	TYPE={<u>SYSTEM</u> | USER} ,ABCODE=(abend-code,...) [,RECOVER={YES | <u>NO</u>}]

TYPE={SYSTEM | USER}
Indicates the type of abend code to be intercepted.

SYSTEM
The abend code is an operating system abend code corresponding to an MVS S*xxx* abend code.

USER
The abend code is a user (including CICS) abend code corresponding to an MVS U*nnnn* abend code.

ABCODE=(abend-code,...)
Code this with the abend code (or codes) to be intercepted. If you specify a single abend code, parentheses are not required.

If you code TYPE=SYSTEM, this abend code must be three hexadecimal digits (*xxx*) representing the MVS system abend code S*xxx*.

If you code TYPE=USER, this abend code must be a decimal number (*nnnn*) representing the user part of the MVS abend code U*nnnn*. This is usually the same number as the CICS message that is issued before CICS tries to terminate abnormally. (See the *CICS Messages and Codes* manual.)

RECOVER={YES|NO}
specifies whether codes are to be added to or removed from the SRT.

YES
Code this to add the specified codes to the SRT.

NO
Code this to remove the specified codes from the SRT.

Notes:

1. CICS intercepts the following abend codes automatically and tries to recover:

   ```
   001,002,013,020,025,026,030,032,033,034,035,
   036,037,03A,03B,03D,0F3,100,113,137,213,214,
   237,283,285,313,314,337,400,413,437,513,514,
   613,614,637,713,714,737,813,837,913,A13,A14,
   B13,B14,B37,D23,D37,E37
   ```

   Abend code 0F3 covers various machine check conditions. It also covers the Alternate Processor Retry condition that can occur only when running on a multiprocessor. CICS-supplied recovery code attempts to recover from instruction-failure machine checks on the assumption that they are not permanent. It also attempts to recover from Alternate Processor Retry conditions.

   CICS will try to recover from the standard abend codes above if you code the system recovery table (SRT) simply as:

   ```
   DFHSRT  TYPE=INITIAL
   DFHSRT  TYPE=FINAL
   END
   ```

   There is no need to list the standard codes individually.
2. If you want CICS to handle other errors, you can code the SRT as follows:

   ```
   DFHSRT  TYPE=INITIAL
   DFHSRT  TYPE=SYSTEM,or USER,
           ABCODE=(user or system codes),
           RECOVER=YES
   DFHSRT  TYPE=FINAL
   END
   ```
3. If you do not want CICS to try to recover after one or more of the above standard abend codes occurs, specify the code(s) with **RECOVER=NO**, or without the **RECOVER** parameter.
4. CICS tries to recover only if a user task (**not** a system task) is in control at the time the abend occurs.

DFHSRT example

Figure 55 illustrates the coding required to generate a SRT.

```
DFHSRT TYPE=INITIAL,           *
       SUFFIX=K1
DFHSRT TYPE=SYSTEM,            *
       ABCODE=777,             *
       RECOVER=YES
DFHSRT TYPE=USER,
       ABCODE=(888,999),       *
       RECOVER=YES
DFHSRT TYPE=USER,              *
       ABCODE=020
DFHSRT TYPE=FINAL
END
```

Figure 55. System recovery table—example

TCT—terminal control table

This section describes the terminal control table and the macro instructions you use to create it.

All VTAM connected terminals, intersystem communication links, and multiregion operation links must be defined using **resource definition online (RDO)**. See "Devices supported" on page 336 for details of VTAM terminals supported by RDO.

How this section is organized

This section begins with an introduction to the TCT and its macros, followed by the reference information for the TYPE=INITIAL and TYPE=GROUP macros. The section is then divided into topics or sub-topics according to the telecommunications access method.

The topics are as follows:

• Logical device codes...	page 444
• Sequential devices...	page 452
• TCAM DCB interface...	page 457
• Remote terminals for transaction routing...	page 474

Introduction to the TCT

A CICS system can communicate with terminals, sequential devices, logical units, and other systems. The TCT defines each of the devices in the configuration. Each TCT entry defines the optional and variable features of the device to CICS, and specifies the optional and variable features of CICS to be used.

CICS uses a **telecommunication access method** to communicate with terminals. This may be **VTAM** or **TCAM**; you may use one or both of these in your system. Note that logical units supporting logical device codes require DFHTCT macros, even though they are VTAM devices.

A **terminal** can be a telecommunication device; for example, an IBM 3279 Color Display Station, or a subsystem such as an IBM 4700 Finance Communication System. Terminals can be local (channel-attached) or remote (link-attached).

You can use a **sequential device** to simulate a CICS terminal. You can define a card reader or punch, line printer, direct access storage device (disk drive), or magnetic tape drive as a sequential device.

A **logical unit (LU)** is a port through which a user of an SNA network gains access to the network facilities.

A **system** can be, for example, another CICS system, an IBM 8815 Scanmaster, an IBM Displaywriter, or an APPC/PC. Intercommunication with CICS systems can be:

- Between different processors (**intersystem communication** or **ISC**), using the LUTYPE 6.1 or LUTYPE 6.2 protocols, or using an intermediate system as an **indirect link**. (Intercommunication with non-CICS systems also uses ISC.)
- Within the same processor (**multiregion operation** or **MRO**), using interregion communication (IRC). (You can also use LUTYPE 6.2 ISC within the same processor.)

There are three ways in which you can create TCT entries, and install them in the TCT.

1. By coding **DFHTCT macros**, assembling them, and selecting the resulting TCT by using the TCT operand at system initialization. This way, TCT entries can be installed at system initialization only.
2. By using **resource definition online (RDO)** to create resource definitions in the CICS system definition (CSD) file. However, you **must** use RDO for VTAM-connected terminals (local and remote) and for MRO and ISC links and sessions. See "TERMINAL" on page 285 for more information on defining terminals using RDO.
3. By using RDO and **autoinstall** (for VTAM-connected terminals only). See "Chapter 8. Autoinstall for VTAM terminals" on page 121 for information about autoinstall for terminals.

If you have terminals connected by other access methods, you **must** use DFHTCT macros, which are described here.

DFHTCT macro types

The DFHTCT macros you code depend on the device you are defining, and on the access method you are using. You always start with one of these:

`DFHTCT TYPE=INITIAL,...`(See page 441.)

There is a special macro to use when you assemble the TCT to migrate RDO-eligible definitions to the CSD file:

`DFHTCT TYPE=GROUP,...`(See page 443.)

You can define your devices in any order you want. Each device needs one or more macros, and these sometimes have to be in a particular order. You are told when this is the case. The macros you need for each type of device or system are as follows:

Table 60. DFHTCT macro types

VTAM terminals, MVS consoles.	For guidance, see "TERMINAL" on page 285.

Logical device codes.	`DFHTCT TYPE=LDC,...`
	`DFHTCT TYPE=LDCLIST,...`

TCAM terminals, sequential devices.	`DFHTCT TYPE=SDSCI,...`
	`DFHTCT TYPE=LINE,...`

Table 60. DFHTCT macro types (continued)

	`DFHTCT TYPE=TERMINAL,...`

Remote terminals, for transaction routing.	`DFHTCT TYPE=REMOTE,...`
	or:
	`DFHTCT TYPE=REGION,...`
	`DFHTCT TYPE=TERMINAL,...`

At the very end of your macros you code:

```
DFHTCT TYPE=FINAL
END
```
(This macro is described on page 371.)

Note on SYSIDNT and TRMIDNT operands: CICS accepts both uppercase and lowercase characters for SYSIDNT and TRMIDNT, but the lowercase characters are not checked for duplication. Assembling a TCT containing lowercase SYSIDNT or TRMIDNT results in an MNOTE.

If you want duplicate checking, use only uppercase characters for SYSIDNT and TRMIDNT.

Note on assembling the TCT: The assembly and link-edit of a TCT leads to the creation of two separate load modules. Assembly of a suffixed TCT (source name DFHTCT*xx*) produces a single text file. However, when this is link-edited into a load library, two members are created:

1. DFHTCT*xx*, which contains the non-RDO-eligible definitions in control block format
2. DFHRDTxx, which contains the RDO-eligible (VTAM terminal and system) definitions in RDO command format

This happens, **whether or not you intend to use RDO**. You need to be aware of the existence of these two tables if you copy or move assembled TCT tables between load libraries.

If you reassemble the TCT after starting CICS, any changes are picked up at a warm or emergency start.

Control section—DFHTCT TYPE=INITIAL

You code one DFHTCT TYPE=INITIAL macro before all the macros that define your resources. The TYPE=INITIAL macro has two purposes:

1. To establish the area of storage into which the TCT is assembled.
2. To specify information that applies to the whole TCT, or to the individual non-VTAM entries (and any VTAM-LDC definitions).

Table 61. DFHTCT TYPE=INITIAL

DFHTCT	TYPE=INITIAL [,ACCMETH=([VTAM,]NONVTAM]

Table 61. DFHTCT TYPE=INITIAL (continued)

```
[[,ERRATT={NO|([LASTLINE]
      [,INTENSIFY]
      [,{BLUE|RED|PINK|GREEN
         |TURQUOISE|YELLOW|NEUTRAL}]
      [,{BLINK|REVERSE|UNDERLINE}])}]
[,MIGRATE={YES|COMPLETE}
[,SUFFIX=xx]


Intercommunication only:

[,SYSIDNT=name]
```

Note: For general information about TYPE=INITIAL macros, see "TYPE=INITIAL (control section)" on page 370.

ACCMETH=([VTAM,]NONVTAM)
This specifies the access methods required in the running CICS system.

VTAM
Specify this if you are using VTAM as an access method. (You can define VTAM devices only with the CEDA transaction, however.)

NONVTAM
Specify this if you are using telecommunications access methods other than VTAM. These access methods include BSAM (for sequential devices) and TCAM.

Note: The default is to assume **both** VTAM and NONVTAM.

**ERRATT={NO|([LASTLINE][,INTENSIFY]
[,{BLUE|RED|PINK|GREEN|TURQUOISE|YELLOW|
NEUTRAL}][,{BLINK|REVERSE| UNDERLINE}])}**
Indicates the way error messages are displayed on all 3270 display screens. You can either let it default to NO, or specify any combination of LASTLINE, INTENSIFY, one of the colors, and one of the highlights. Specifying INTENSIFY, one of the colors, or one of the highlights forces LASTLINE.

Any attributes that are not valid for a particular device are ignored.

NO
Any error message is displayed at the current cursor position and without any additional attributes.

LASTLINE
Any error message is displayed starting at the beginning of the line nearest the bottom of the screen, such that the message fits on the screen.

Attention: If messages are received in quick succession, they overlay one another. The earlier messages may disappear before the operator has read them.

INTENSIFY
Error messages are intensified, and placed on the last line of the screen.

BLUE|RED|PINK|GREEN|TURQUOISE|YELLOW| NEUTRAL
Error messages are shown in the color specified, and placed on the last line of the screen.

BLINK|REVERSE|UNDERLINE
Error messages are highlighted, and placed on the last line of the screen.

MIGRATE={YES|COMPLETE}
This operand controls the building of TCT entries for VTAM devices that are **eligible** for resource definition online (RDO). The only way RDO-eligible resources may be moved to the CSD file from the macro source is to use DFHCSDUP, as described under YES below.

YES
YES indicates that you want to generate the necessary data to migrate your RDO-eligible resources. The records generated from the macro source are designed to be used as input to the DFHCSDUP utility program. An MNOTE attention message is issued for each RDO-eligible resource. The DFHCSDUP MIGRATE command converts them into resource definitions on the CSD file. These can then be managed using RDO.

COMPLETE
Use of COMPLETE means that TCT entries are not generated from the macro source for any RDO-eligible devices. For each one, the assembly produces an MNOTE. This means that you can keep your TCT macro source code after you have migrated your definitions.

If you continue to assemble a TCT for resources that are not eligible for RDO, continue to use MIGRATE=COMPLETE.

SYSIDNT={CICS|name}
This 1- to 4-character name is a private name that the CICS system uses to identify itself. If you use DFHTCT TYPE=REGION macros to define remote terminals, you must code this operand. It is used to determine whether a remote or a local TCT terminal entry is generated from each TYPE=TERMINAL macro following the TYPE=REGION macro. If the SYSIDNT on the TYPE=REGION macro is the same as the SYSIDNT on the TYPE=INITIAL macro, a local definition is generated; otherwise a remote definition is generated.

The value you code for this operand is used to define the name of the local region during assembly of the TCT only. You must also define the name of the local region, for execution purposes, using the SYSIDNT system initialization parameter.

Migrating TCT definitions—DFHTCT TYPE=GROUP

Use this macro to name the groups into which TCT definitions are put when you migrate to resource definition online. This macro can appear as many times as required and at any point in the macro source. Each time it appears, it defines the CSD file group into which subsequent definitions are put until the next DFHTCT TYPE=GROUP macro occurs.

Table 62. DFHTCT TYPE=GROUP

DFHTCT	TYPE=GROUP
	[,GROUP=name]

GROUP=name
Code this with the name of the group to which subsequent definitions are

migrated. The name can be up to eight alphanumeric characters, but must not begin with DFH. The default name is TCT*xx*, where *xx* is the value coded for SUFFIX in the DFHTCT TYPE=INITIAL macro. If an error is found, the existing group name continues.

If a group with the name you specify does not already exist, it will be created. If it does exist, subsequent definitions are added to it.

For details of how to migrate your TCT to the CSD file, see “Appendix D. Migrating the TCT to the CSD file” on page 545.

DFHTCT logical device codes: VTAM non-3270

Certain types of logical unit may be used to gain access to more than one resource within a subsystem. For example, a card punch device may be attached to a 3770 logical unit: the CICS application program can direct punch output, through BMS, via the 3770 to the card punch device. The facility provided by CICS to permit communication to devices within logical units of this type is the **logical device code** (LDC).

Although these are VTAM units, they require macro definition, unlike other VTAM devices.

The logical units that support LDCs are:
3601 logical unit
3770 batch logical unit
3770 batch data interchange logical unit
3790 batch data interchange logical unit
LUTYPE 4 logical unit

To reference such a device in a CICS application program, or in the CMSG transaction for message switching, you specify an LDC mnemonic which CICS translates into a numeric LDC value. When CICS sends an output data stream to the logical unit, it includes the LDC value in the function management header (FMH). When the logical unit receives the data stream, it uses the LDC value to determine which component is to receive the output, or to perform some standard action.

Each LDC mnemonic to be referenced must be defined in the TCT, optionally with its associated LDC value and certain device characteristics for use by BMS functions. Such LDC information is contained in either the **system LDC table**, or in an **extended local LDC list**. You code the following DFHTCT macros to specify the system LDC table or an extended local LDC list:

- Code DFHTCT TYPE=LDC macro(s) to generate entries in the system LDC table. You may generate certain default LDC entries provided by CICS. For example,

  ```
  DFHTCT TYPE=LDC,LDC=SYSTEM
  ```

  generates the following entries in the system LDC table:

Table 63. System LDC table entries

LDC mnemonic	LDC value	Device	Pagesize (row,column)
DS	1	3604 Keyboard Display	6,40
JP	2	3610 Document Printer	1,80
PB	3	Passbook and Document Printer	1,40
LP	4	3618 Administrative Line Printer	50,80

Table 63. System LDC table entries (continued)

LDC mnemonic	LDC value	Device	Pagesize (row,column)
MS	5	3604 Magnetic Stripe Encoder	1,40
CO	0	Console medium or default print data set group	
R1	32	Card input medium	1,80
H1	32	Card output medium	1,80
P1	48	Print medium or print data set group	50,80
W1	128	Word processing medium 1	50,80
W2	144	Word processing medium 2	50,80
W3	160	Word processing medium 3	50,80
W4	192	Word processing medium 4	50,80

You may also define LDCs specifically to add LDC entries to the system LDC table. For example,

```
DFHTCT TYPE=LDC,
       LDC=XX,
       DVC=BLUPRT,
       PGESIZE=(12,80),
       PGESTAT=PAGE
DFHTCT TYPE=LDC,
       LDC=YY,
       DVC=BLUPCH,
       PGESIZE=(1,80),
       PGESTAT=AUTOPAGE
```

adds the following entries to the system LDC table:

Table 64. System LDC table entries defined by LDCs

LDC mnemonic	LDC value	Device	Pagesize (row,column)	PGESTAT
XX	48	Batch LU printer	12,80	PAGE
YY	32	Batch LU card output	1,80	AUTOPAGE

- Instead of the system LDC table, you may code the following series of DFHTCT TYPE=LDC macros to create an extended local LDC list. Default entries may also be generated. For example,

```
LDC1  DFHTCT TYPE=LDC,LOCAL=INITIAL
*  the next line generates default CO,R1,H1,P1 LDCs
   DFHTCT TYPE=LDC,LDC=BCHLU
   DFHTCT TYPE=LDC,DVC=BLUPRT,LDC=AA,
          PGESIZE=(6,30)
   DFHTCT TYPE=LDC,DVC=BLUPCH,LDC=BB,
          PGESIZE=(1,80)
   DFHTCT TYPE=LDC,DVC=BLUCON,LDC=CC,
          PGESIZE=(1,132)
   DFHTCT TYPE=LDC,LOCAL=FINAL
```

generates an extended local LDC list named LDC1 containing the following entries:

Table 65. Extended local LDC list entries

LDC mnem-onic	LDC value	Device	Pagesize (row,column)
CO	0	Console medium or default printer data set group	
R1	32	Card input medium	1,80
H1	32	Card output medium	1,80
P1	48	Print medium or print data set group	50,80
AA	48	BLUPRT batch LU printer	6,30
BB	32	BLUPCH batch LU card output	1,80
CC	0	BLUCON batch LU console printer	1,132

When you are defining a logical unit in the TCT, you can specify its LDCs in either of two ways:

1. Code a DFHTCT TYPE=LDCLIST macro to define a local list of LDC mnemonics (and optionally their LDC values); for example,

   ```
   LDC2 DFHTCT TYPE=LDCLIST,
               LDC=(DS,JP,PB=5,LP,MS)
   ```

   In the DFHTCT TYPE=TERMINAL macro defining the logical unit, you specify in the LDC operand the name of the local list as defined by the DFHTCT TYPE=LDCLIST macro. For example:

   ```
        DFHTCT TYPE=TERMINAL,
               TRMTYPE=3600,
               LDC=LDC2, ...
   ```

   has associated the LDCs DS, JP, PB, LP, and MS with the 3601 logical unit that you are defining. The LDC values either may be specified in the local list, or are obtained from the system LDC table. If BMS uses these LDC mnemonics, their page size and page status must also be available from the system LDC table.

   Note: A local list defined by a DFHTCT TYPE=LDCLIST macro may be shared by a number of 3601, LUTYPE 4 and batch logical units.

2. In the DFHTCT TYPE=TERMINAL macro defining the logical unit, you specify in the LDC operand the name of an extended local LDC list. For example:

   ```
   LDC1 DFHTCT TYPE=LDC,LOCAL=INITIAL
        DFHTCT TYPE=LDC,LDC=BCHLU
        DFHTCT TYPE=LDC,DVC=BLUPRT,LDC=AA,
               PGESIZE=(6,30)
        DFHTCT TYPE=LDC,DVC=BLUPCH,LDC=BB,
               PGESIZE=(1,80)
        DFHTCT TYPE=LDC,DVC=BLUCON,LDC=CC,
               PGESIZE=(1,132)
        DFHTCT TYPE=LDC,LOCAL=FINAL
        DFHTCT TYPE=TERMINAL,TRMTYPE=BCHLU,
               LDC=LDC1, ...
   ```

   has associated the LDCs CO, R1, H1, P1, AA, BB, and CC with the batch logical unit that you are defining. Their LDC values and device characteristics for BMS functions are described in the extended local LDC list, which is named LDC1.

When CICS requests an output or message switching operation using a particular LDC mnemonic for a logical unit, it tries to resolve the mnemonic from the list (whichever form) specified by the LDC operand of the DFHTCT TYPE=TERMINAL macro. If the LDC is not located in the local list or in the extended local list, the LDC specified is not valid for that terminal entry. In this case, X'00' is inserted in the logical device code portion of the FMH, and no destination name is inserted.

When a BMS function is requested for an LDC, and the LDC mnemonic is successfully resolved, the device characteristics (for example, device name and destination name) are accessed for the BMS function. If the LDC is in an extended local LDC list, these characteristics lie in the located extended local list entry. Otherwise, the system LDC table is searched for the LDC and the associated device characteristics.

Logical device codes—DFHTCT TYPE=LDC macro: The DFHTCT TYPE=LDC macro may only be used with 3600, LUTYPE4, 3770 batch logical unit, and 3770/3790 batch data interchange logical units.

You are responsible for setting up the LDC structure to be used with the terminal.

The expansion of this macro is the same, regardless of where it is specified in the TCT definition.

Table 66. DFHTCT TYPE=LDC

```
[name]      DFHTCT        TYPE=LDC
                          [,DSN=destination-name]
                          [,DVC=(device-type,sub-address)]
                          [,LDC={SYSTEM|LUTYPE4|3600|
                          BCHLU|(aa[=nnn])}
                          [,LOCAL={INITIAL|FINAL}]
                          [,PGESIZE=(row,column)]
                          [,PGESTAT={AUTOPAGE|PAGE}]
```

name
: Code this with the name of the extended local LDC list. It should be the same as that specified in the LDC operand of the DFHTCT TYPE=TERMINAL macro, and is only required if LOCAL=INITIAL is coded.

TYPE=LDC
: Code this if an LDC is being defined to the system LDC table or to the extended local LDC list.

DSN=destination-name
: Code this with the name to be used by BMS for destination selection for the batch data interchange logical unit. See the relevant CICS subsystem guides for further information on destination selection.

DVC=(device-type,sub-address)
: Code this with the device type associated with the LDC to be used for a BMS request. This operand can only be coded in conjunction with the LDC=*aa*[=*nnn*] operand.

device-type
: May be coded as follows:

Table 67. DVC=device-type entries

Device type	Explanation
3604	Keyboard display
3610	Cut-forms document printer or journal printer (including the document/journal printer of a 3612)
3612	Passbook portion of a 3612
3618	Currently selected carriage
3618P	Primary carriage
3618S	Secondary carriage
3618B	Both carriages
BLUCON	Batch logical unit console printer
BLUPRT	Printer component of a batch logical unit
BLURDR	Card input component of a batch logical unit
BLUPCH	Card output component of a batch logical unit
WPMED1	Word processing medium 1
WPMED2	Word processing medium 2
WPMED3	Word processing medium 3
WPMED4	Word processing medium 4

The device types BLUPRT, BLURDR, BLUPCH, and BLUCON are devices attached to a batch, batch data interchange, or LUTYPE4 logical unit.

The WPMED1, 2, 3, and 4 options apply to LUTYPE4 logical units only. The component to which these options apply is defined by the particular type 4 logical unit implementation.

sub-address
Code this with the media sub-address. The range is 0 through 15, with a default of 0. A value of 15 indicates any sub-address. The sub-address differentiates between two units of the same device type (for example, (BLUPRT,0) and (BLUPRT,1)), which could be two print components attached to one logical unit.

LDC={SYSTEM|LUTYPE4|3600|BCHLU|(aa[=nnn])}
Code this with the LDC mnemonic and numeric value to be defined. Only the LDC=*aa*[=*nnn*] option can be used in conjunction with the DVC, PGESIZE, and PGESTAT operands.

SYSTEM
The following system-default LDCs for 3600, batch, and LUTYPE4 logical units are to be established:

Table 68. System default LDCs

LDC mnemonic	LDC value	Device	Pagesize (row, column)
DS	1	3604 Keyboard Display	6,40
JP	2	3610 Document Printer	1,80
PB	3	Passbook and Document Printer	1,40
LP	4	3618 Administrative Line Printer	50,80
MS	5	3604 Magnetic Stripe Encoder	1,40
CO	0	Console medium or default print data set group	

Table 68. System default LDCs (continued)

LDC mnem-onic	LDC value	Device	Pagesize (row, column)
R1	32	Card input medium	1,80
H1	32	Card output medium	1,80
P1	48	Print medium or print data set group	50,80
W1	128	Word processing medium 1	50,80
W2	144	Word processing medium 2	50,80
W3	160	Word processing medium 3	50,80
W4	192	Word processing medium 4	50,80

LUTYPE4
System-default LDC mnemonics are to be established for an LUTYPE4 (word processing) logical unit. These consist of the CO, R1, P1, H1, W1, W2, W3, and W4 mnemonics, the corresponding LDC values, and the appropriate page sizes.

3600
System-default LDC mnemonics for the 3600 are to be established. These consist of the DS, JP, PB, LP, and MS mnemonics, the corresponding LDC values, and the appropriate page-sizes and page-status.

BCHLU
System-default LDC mnemonics for a batch logical unit are to be established. These consist of the CO, R1, P1, and H1 mnemonics, the corresponding LDC values, and the appropriate page-sizes and page-status.

aa The 2-character mnemonic to be used for this LDC.

nnn
The numeric value to be associated with the LDC in the system or extended local LDC list. The value in the system list is used as a default value for this LDC if a value is not found in a local LDC list (that is, not in the extended list) associated with a TCTTE. A value must be specified for a 3600 device. A value need not be specified for a batch, batch data interchange, or LUTYPE4 logical unit but, if one is specified, it must correspond to the LDC value for the device type.

LOCAL={INITIAL|FINAL}
An extended local LDC list is to be generated.

INITIAL
This is the start of an extended local LDC list.

FINAL
This is the end of an extended local LDC list.

Note: LOCAL=INITIAL or FINAL may not be coded in the same DFHTCT TYPE=LDC macro as other operands. All DFHTCT TYPE=LDC entries coded after LOCAL=INITIAL and before LOCAL=FINAL form part of one extended local LDC list; the entries coded outside the structure of this group are added to the system LDC table.

The following is an example of an extended local LDC list:

```
         DFHTCT TYPE=TERMINAL,TRMIDNT=BTCH,      *
         TRMTYPE=BCHLU,ACCMETH=VTAM,LDC=LDCA     *
   LDCA  DFHTCT TYPE=LDC,LOCAL=INITIAL
         DFHTCT TYPE=LDC,DVC=BLUPRT,LDC=AA,      *
         PGESIZE=(6,30)
         DFHTCT TYPE=LDC,DVC=BLUPCH,LDC=BB,      *
         PGESIZE=(1,80)
         DFHTCT TYPE=LDC,DVC=BLUCON,LDC=CC,      *
         PGESIZE=(1,132),PGESTAT=AUTOPAGE
         DFHTCT TYPE=LDC,LOCAL=FINAL
```

PGESIZE=(row,column)
Code this with the logical page size to be used with this LDC when BMS requests are processed.

The product of *row* and *column* must not exceed 32767.

PGESTAT={AUTOPAGE|PAGE}
Indicates whether the device is to use autopaging or not. Autopaging means that BMS multiple page messages are printed continuously, without operator intervention. This is what is normally required for a printer. (Contrast the requirement for multiple page messages, displayed on a 3270-type display, when the operator wants to finish reading a page, before requesting the next page to be delivered.)

Only BMS SEND commands with the PAGING option use autopaging. BMS SEND with TERMINAL or SET, does not use autopaging.

AUTOPAGE
Specify this for printers.

PAGE
Specify this for displays.

If the default PGESIZE or PGESTAT values provided by the LDC operand are to be overridden, code a specific LDC with the mnemonic to be overridden. Code this overriding LDC in the LDC table before coding the LDC operand.

PGESTAT=AUTOPAGE may be used to override the PGESTAT specification in DFHTCT TYPE=TERMINAL.

Logical device codes—DFHTCT TYPE=LDCLIST: The DFHTCT TYPE=LDCLIST macro, which may be used with 3600, LUTYPE4, and batch logical units, allows you to build a common list of logical device codes (LDCs) to be shared by more than one TCTTE.

You are responsible for setting up the LDC structure to be used with the terminal.

The expansion of this macro is the same, regardless of where it is coded in the TCT definition.

Table 69. DFHTCT TYPE=LDCLIST

listname	DFHTCT	TYPE=LDCLIST
		,LDC=(aa [=nnn] [,bb[=nnn] [,cc[=nnn],...)

listname
Is the required name of the LDC list. This name is referenced by TCTTEs through the LDC operand in DFHTCT TYPE=TERMINAL.

TYPE=LDCLIST
An LDC list is being defined.

LDC=(aa[=nnn][,bb[=nnn]][,cc[=nnn]][,...])
Code this with the LDCs (mnemonics and, optionally, the LDC numeric value) in this list.

(aa[=nnn][,bb[=nnn]] [,cc[=nnn]][,...])
Generates the LDCs in the list.

aa,bb,cc...
The 2-character mnemonics of the LDCs in this list.

nnn
A decimal value in the range 1 through 255 to be associated with an LDC. If a value is not specified, the system default value from the table defined by the DFHTCT TYPE=LDC macro, is used for this LDC. This value need not be coded for a batch or LUTYPE4 logical unit, but if it is, it must correspond to the LDC value for the device. LDCs for devices attached to a batch or LUTYPE4 logical unit are listed under the LDC parameter of the DFHTCT TYPE=LDC macro.

DFHTCT examples: LDC

The 3770 System and the 6670 Information Distributor: The *CICS/OS/VS IBM 3767/3770/6670 Guide* provides information for CICS users who intend to install a CICS system that communicates with an IBM 3767 Communication Terminal, a 3770 Communication System, or a 6670 Information Distributor.

LDCs for 3770 batch logical unit TCT example:

```
DFHTCT TYPE=LDC,                          *
       LDC=XX,                            *
       DVC=BLUPRT,                        *
       PGESIZE=(12,80),                   *
       PGESTAT=PAGE
DFHTCT TYPE=LDC,                          *
       LDC=YY,                            *
       DVC=BLUPCH,                        *
       PGESIZE=(1,80),                    *
       PGESTAT=AUTOPAGE
DFHTCT TYPE=LDC,                          *
       LDC=SYSTEM
```

Figure 56. LDCs for 3770 batch logical unit TCT example

6670 LUTYPE 4 TCT example: You must use RDO to define the terminal. For guidance, refer to the LDCLIST keyword of the TYPETERM option on page 351.

DFHTCT TYPE=LDCLIST: VTAM non-3270

```
LDCS   DFHTCT  TYPE=LDC,LDC=SYSTEM
LDC1   DFHTCT  TYPE=LDC,LOCAL=INITIAL
       DFHTCT  TYPE=LDC,DVC=(BLUCON,01),           *
               PROFILE=DEFAULT,LDC=PC,             *
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
       DFHTCT  TYPE=LDC,DVC=(BLUPRT,02),           *
               PROFILE=BASE,LDC=PP,                *
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
       DFHTCT  TYPE=LDC,DVC=(BLUPRT,08),           *
               PROFILE=BASE,LDC=P8,                *
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
       DFHTCT  TYPE=LDC,DVC=(BLUPRT,08),           *
               PROFILE=DEFAULT,LDC=DP,             *
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
       DFHTCT  TYPE=LDC,DVC=(BLUPCH,03),           *
               PROFILE=JOB,LDC=PM,                 *
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
       DFHTCT  TYPE=LDC,DVC=(BLUPCH,03),           *
               PROFILE=DEFAULT,LDC=DM,             *
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
       DFHTCT  TYPE=LDC,DVC=(WPMED1,04),           *
               PROFILE=WPRAW,LDC=P1,               *
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
       DFHTCT  TYPE=LDC,DVC=(WPMED1,04),           *
               PROFILE=DEFAULT,LDC=D1,             *
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
       DFHTCT  TYPE=LDC,DVC=(WPMED2,05),           *
               PROFILE=OII1,LDC=P2,                *
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
       DFHTCT  TYPE=LDC,DVC=(WPMED2,05),           *
               PROFILE=DEFAULT,                    *
               LDC=D2,PGESIZE=(50,80),             *
               PGESTAT=AUTOPAGE
       DFHTCT  TYPE=LDC,DVC=(WPMED3,06),           *
               PROFILE=OII2,                       *
               LDC=P3,PGESIZE=(50,80),             *
               PGESTAT=AUTOPAGE
       DFHTCT  TYPE=LDC,DVC=(WPMED4,07),           *
               PROFILE=OII3,LDC=P4,                *
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
       DFHTCT  TYPE=LDC,LOCAL=FINAL
```

Figure 57. 6670 Lutype 4 TCT example

Sequential devices

CICS uses BSAM to control sequential devices such as card readers, line printers, magnetic tape units, and DASD to simulate terminals. Only unblocked data sets can be used with BSAM.

These "sequential terminals" may be used before actual terminals are available, or during testing of new applications.

To define a sequential device, code the following macro instructions:

```
DFHTCT TYPE=INITIAL,
       ACCMETH=(NONVTAM)    defining the access
                            method
```

(Define the following macro instructions contiguously.)

```
DFHTCT TYPE=SDSCI,
       DSCNAME=isadscn,     defining the input
       DDNAME=indd, ...      data set
DFHTCT TYPE=SDSCI,
       DSCNAME=osadscn,     defining the output
```

```
        DDNAME=outdd, ...     data set
 DFHTCT TYPE=LINE,
        ISADSCN=isadscn,
        OSADSCN=osadscn, ...
 DFHTCT TYPE=TERMINAL,
        TRMIDNT=name, ...
```

The two data sets defined by the DFHTCT TYPE=SDSCI macros simulate a CICS terminal known by the name specified in the TRMIDNT operand of the DFHTCT TYPE=TERMINAL macro. The DSCNAMEs of the input and output data sets must be specified in the ISADSCN and OSADSCN operands of the DFHTCT TYPE=LINE macro respectively.

The end of data indicator (EODI) for sequential devices may be altered using the EODI system initialization parameter.

JCL for sequential devices: The DDNAME operands on the DFHTCT TYPE=SDSCI macros specify the ddname of the DD statements which you must provide in the CICS startup job stream:

```
//indd   DD  ...               input data set
//outdd  DD  ...               output data set
```

where *indd* is the data set containing input from the simulated terminal, and *outdd* is the data set to which output to the simulated terminal is sent.

Sequential devices—DFHTCT TYPE=SDSCI

Table 70. DFHTCT TYPE=SDSCI

DFHTCT	TYPE=SDSCI
	,DEVICE=device
	,DSCNAME=name
	[,BLKSIZE=length]
	[,DDNAME={<u>name-in-DSCNAME</u>|name}]
	[,MACRF=([R][,W])]
	[,RECFM={<u>U</u>|F|V}]

BLKSIZE=length
Code this with the maximum length in bytes of a block.

The default is BLKSIZE=0. If this operand is omitted, the block size can be specified in the data definition (DD) statement associated with the data set. A more detailed explanation of this operand is given in the *MVS/ESA Data Administration: Macro Instruction Reference* manual.

DDNAME={<u>name-in-DSCNAME</u>|name}
Supplies the name of the data definition (DD) statement associated with a particular data set (line group). If this operand is omitted, the DSCNAME becomes the DDNAME.

DEVICE=device
One of the following values may be coded:

- For card readers: **{1442|2501|2520|2540|2560|2596| 3505|3525|5425}**
- For line printers: **{1403|1404|1443|1445|3203|3211|5203}**
- For disk (DASD): **{2314|3330|3340|3350|DASD|DISK}**
- For tapes: **TAPE**.

The TAPE specification generates tape work files for both the input and the output data sets. Note that if an input tape with an expired label is used, the header may be rewritten, causing the first data records to be destroyed.

DSCNAME=name
The name of either the input or the output data set. If you are defining the input data set, ISADSCN on the DFHTCT TYPE=LINE macro must match the name that you specify: if you are defining the output data set, OSADSCN on the DFHTCT TYPE=LINE macro must match it.

MACRF=([R][,W])
Code this with the way in which access to the sequential device is to be gained.

R	Indicates the READ macro.
W	Indicates the WRITE macro.

The default is MACRF=R for a card reader and MACRF=W for a line printer. For other sequential devices, MACRF=R or MACRF=W must be coded.

RECFM={U|F|V}
Code this with the record format for the DCB.

U	Indicates undefined records. Code this option for DEVICE=1403 or 3211, or if you are using DASD for sequential terminal output (that is, if DEVICE=DASD and MACRF=W).
F	Indicates fixed-length records.
V	Indicates variable-length records.

If you omit this operand, you can specify the record format in the data definition (DD) statement associated with the sequential data set.

Sequential devices—DFHTCT TYPE=LINE

Table 71. DFHTCT TYPE=LINE

```
DFHTCT        TYPE=LINE
              ,ACCMETH={SAM|BSAM|
                SEQUENTIAL}
              ,INAREAL=length
              ,ISADSCN=input-name
              ,OSADSCN=output-name
              ,TRMTYPE={U/R|CRLP|DASD|TAPE}
              [,LINSTAT='OUT OF SERVICE']
              [,TCTUAL={0|length}]
```

ACCMETH={SAM|BSAM|SEQUENTIAL}
Specify SAM, BSAM, or SEQUENTIAL — they are equivalent in CICS.

INAREAL=length
Code this with the message input area length. The value should be equal to the length of the longest initial logical record of a transaction that may include multiple physical records.

ISADSCN=name
The name of the input data set. The TYPE=SDSCI DSCNAME operand for the input data set must match this.

LINSTAT='OUT OF SERVICE'
The line is to be initiated with an 'out of service' status.

The default is 'in service'.

OSADSCN=name
The name of the output data set. The TYPE=SDSCI DSCNAME operand for the output data set must match this.

TCTUAL={0|length}
Indicates the length in bytes (0 through 255) of the user area (the process control information field or PCI) for all terminal entries (TCTTEs) associated with this line. Make it as small as possible. The TCT user area is initialized to zeros at system initialization. If you want fields of different (variable) lengths, you can specify the TCTUAL value in one or more TYPE=TERMINAL macro instructions for terminals associated with this line.

TRMTYPE=(U/R|CRLP|DASD|TAPE)
Indicates the sequential device type:

U/R	Any reader or printer
CRLP	A card reader and a line printer
DASD	A direct access storage device
TAPE	A magnetic tape device

Sequential devices—DFHTCT TYPE=TERMINAL

Table 72. DFHTCT TYPE=TERMINAL

```
DFHTCT          TYPE=TERMINAL
                [,LPLEN={120|value}]
                [,PGESIZE=(lines,columns)]
                [,TCTUAL=
                {number-specified-in-TYPE=LINE|
                  number}]
                [,TRANSID=
                transaction-identification-code]
                [,TRMPRTY={0|number}]
                [,TRMSTAT={TRANSACTION|
                (status[,s,...)}]
                [,USERID=userid]
```

LPLEN={120|value}
Controls the length of the print line for SAM output line printers. If no NL symbol is found in a segmented write, the print line length is the LPLEN value. The default is LPLEN=120.

PGESIZE=(lines,columns)
The default page size for a 1403 or CRLP terminal is (12,80). Code PGESIZE if BMS is required for a device that has TRMTYPE=DASD specified, and specify the number of lines and columns you wish to use. These two values multiplied together must equal the value specified for INAREAL. The product of *lines* and *columns* must not exceed 32767.

TCTUAL={number-specified-in-TYPE=LINE |number}
Indicates the length in bytes (0 through 255) of the user area (the process control information field or PCI) for the terminal entry (TCTTE) associated with this terminal. Make it as small as possible. The TCT user area is initialized to zeros at system initialization.

Use the TCTUAL operand of the DFHTCT TYPE=TERMINAL macro if you want fields of different (variable) lengths for terminals associated with this line. In any case, the PCI field is generated for each terminal after the last terminal entry of the last line. The address of the PCI field is located at TCTTECIA; the length is located at TCTTECIL.

TRANSID=transaction-identification-code

Code this with a 1- to 4-character transaction code. This code specifies a transaction that is to be initiated each time input is received from the terminal when there is no active task.

If a TRANSID is not specified in the TCTTE, the TRANSID in a RETURN command from the previous transaction is used. Otherwise, the first one to four characters of the data passed in the TIOA are used as the transaction code. A delimiter is required for transaction identifications of fewer than four characters.

TRMIDNT=name

Code this with a unique 4-character symbolic identification of each terminal. The identification supplied is left-justified and padded with blanks to four characters if less than four characters are supplied.

The value CERR is reserved, as this is the identification generated for the error console.

TRMPRTY={0 | number}

Establishes the terminal priority. This decimal value (0 through 255) is used in establishing the overall transaction processing priority. (Transaction processing priority is equal to the sum of the terminal priority, transaction priority, and operator priority, and must not exceed 255.)

TRMSTAT={TRANSACTION | (status,...)}

Code this with the types of activity that may occur at a given terminal. This terminal status is initially set in the TCTTE and is a combination of the processing status and the service status.

TRANSACTION

A terminal with TRANSACTION status is used in the processing of transactions such as inquiries or order entries. A display station or a hard-copy terminal, to which no messages are sent without a terminal request, and through which transactions are entered, is a TRANSACTION terminal.

INPUT

Indicates a terminal that can send messages to, but cannot receive messages from, CICS.

Note: System messages may be routed to an input terminal under conditions such as invalid transaction identification and ATP batch count. This causes DFHTACP to be scheduled. To handle this situation, code a DFHTEP to perform any action that the user requires.

'OUT OF SERVICE'

Indicates a terminal that can neither receive messages nor transmit input. Such terminals are not polled by CICS. The 'OUT OF SERVICE' parameter can be used in combination with any status setting.

Any terminal except the master terminal can be designated as 'OUT OF SERVICE'. When appropriate, the terminal can be placed in service by the master terminal and polling is resumed.

RECEIVE
Indicates a terminal to which messages are sent but from which no input is allowed. An example of this type of terminal is one that is located in a remote location, such as a warehouse, and is unattended, but may receive messages. Automatic transaction initiation is implemented as for TRANSCEIVE, below.

TRANSCEIVE
A terminal with TRANSCEIVE status is a TRANSACTION terminal to which messages are sent automatically. The automatic transaction initiation, either by transient data control or interval control, sets a condition in an appropriate terminal control table terminal entry. If the terminal status is TRANSCEIVE and if there is no transaction at the terminal, terminal control initiates the user-defined task. This task is expected to send messages to the terminal.

USERID=userid
Code this to specify a user identifier for devices such as printers that are unable to sign on using CESN. (You can also specify USERID for a display device, in which case the display is permanently signed on. Operators are unable to sign on.) You must code this operand if you want to use preset security with this device. All access to protected resources depends on USERID.

The userid is referred to in security error messages, security violation messages, and the audit trail. It must be defined to the security manager.

Userid must be a unique 1- to 8-character user identification. (A-Z 0-9 # $ and @ are acceptable characters.)

TCAM DCB interface

CICS supports the DCB interface (also known as the GET/PUT interface) of ACF/TCAM in an SNA or non-SNA environment. This section describes the DFHTCT macros you code to define the CICS terminals connected to this interface. We use **TCAM** to mean this DCB interface.

Each TCAM communication line is associated with two sequential queues: the input process queue and the output process queue. CICS routes messages for terminals connected using TCAM to the queue named in the DEST option of the SEND and CONVERSE commands. CICS assumes that there is a user-written message control program (MCP) to process messages on the TCAM queues. CICS leaves it to the TCAM MCP to poll and address terminals, translate code, and control lines. Therefore, a number of DFHTCT operands are irrelevant for TCAM. This section describes how to define CICS terminals.

TCAM—DFHTCT TYPE=SDSCI, LINE, TERMINAL: You code one DFHTCT TYPE=SDSCI macro for each input queue, and one for each output queue. The macros generate DCBs, corresponding to TPROCESS blocks. A queue is treated like a communication line by CICS, and each one must be described by a DFHTCT TYPE=LINE macro. The DFHTCT TYPE=LINE macro generates a TCT line entry (TCTLE), of which where is one for each queue.

Each TCAM terminal must be described in a DFHTCT TYPE=TERMINAL macro. This generates a TCT terminal entry (TCTTE), of which there is one for each terminal. To avoid duplication of the TCTTEs for both the input queue and the output queue, all the terminals are described immediately following the DFHTCT TYPE=LINE macro for the output queue. Although attached to the output TCTLE, these TCTTEs are used for both input and output processing. One dummy TCTTE must also be generated for the input TCTLE; this need have only a TRMIDNT operand giving a dummy terminal identification and a LASTTRM operand.

Each input record from TCAM must contain the source terminal identification. Using this identification as a search argument, CICS can find the corresponding TCTTE by comparing against the NETNAME value for each TCTTE.

Note: The usual way of ensuring that the input records contain the source terminal identification is to specify OPTCD=W in the DFHTCT TYPE=SDSCI macro. If you omit this specification, the TCAM user is responsible for ensuring that the record contains a suitable source terminal identification.

Using the POOL feature (by specifying POOL=YES on the DFHTCT TYPE=LINE macro), it is possible to establish a pool of common TCTTEs on the output TCTLE that do not contain terminal identifiers. As required, terminal identifiers are assigned to the TCTTEs or removed from association with the TCTTEs. This POOL feature necessarily imposes a number of restrictions and should be thoroughly understood before it is used.

TCAM devices—DFHTCT TYPE=SDSCI:

Table 73. DFHTCT TYPE=SDSCI

DFHTCT	TYPE=SDSCI
	,BLKSIZE=length
	[,DDNAME={<u>name-in-DSCNAME</u> | name}]
	,DEVICE=TCAM
	,DSCNAME=name
	,MACRF=([R][,W])
	[,RECFM=<u>U</u>]
	[,OPTCD={W | WU | WC | WUC | U | C | UC}]
	[,SYNAD=symbolic-name]

BLKSIZE=length
Code this with the maximum length of a block in bytes. For TCAM queues, the block size value must specify the maximum length that any CICS application program requires to be written in one request. Note that CICS application programs include the master terminal command. A block size of at least 2024 bytes (one screen size plus attribute bytes) should be specified.

DDNAME={<u>name-in-DSCNAME</u> | name}
Supplies the name of the data definition (DD) statement associated with a particular data set (line group). If this operand is omitted, the DSCNAME becomes the DDNAME.

DEVICE=TCAM
This causes CICS to generate the appropriate data set control information to handle the TCAM input or output process queue.

DSCNAME=name
Code this with the symbolic data set control name associated with the data set control information and its related communication line(s).

The same DSCNAME must be specified on the DFHTCT TYPE=SDSCI macro and on its related DFHTCT TYPE=LINE macro(s).

MACRF=([R],[W])
Code this with how access to the TCAM device is to be gained.

R indicates the READ macro.

W
indicates the WRITE macro.

OPTCD={W|WU|WC|WUC|U|C|UC}
Code this with the optional fields for the TCAM work unit.

C Code this if a one-byte field in the work area, called the position field, indicates whether the work unit being handled is the first, an intermediate, or the last segment of the message, and, on input, whether a record delimiter has been detected in the data.

U Code this if the work unit to be handled is either a message or a message segment that is not a record. If U is omitted, the work unit is assumed to be a record.

W For input, code this if TCAM is to place the name of the source of each message in an 8-byte origin field in the work area. If OPTCD=W is omitted, you must ensure that a source name is placed in the origin field.

For output, code this if the name of the destination of the message is placed in an 8-byte destination field in the work area before a TCAM WRITE macro is executed. CICS always inserts a destination name. If OPTCD=W is omitted, you must provide for interpretation of the destination field.

For details of the OPTCD operand, see the *ACF/TCAM Version 3 Application Programming* manual.

RECFM=U
Code this with the record format for the DCB.

U This indicates undefined records. This option must be coded for TCAM queues used by CICS.

SYNAD=symbolic-name
Specifies the address of a subroutine that is to be given control if message processing is used, if the work unit is larger than the work area, or if OPTCD=C is not coded.

For input queues, a user-written SYNAD routine can be specified for which an EXTRN is generated. If SYNAD is not coded, a CICS generated SYNAD routine is provided. If CICS SYNAD is used and the exit occurs:
1. Message DFH4000 is issued.
2. The DCB is closed.
3. The DCB is reopened.
4. Data is truncated to the specified block size and is passed to the CICS application program.

TCAM devices—DFHTCT TYPE=LINE:

Table 74. DFHTCT TYPE=LINE

DFHTCT	TYPE=LINE
	,ACCMETH=TCAM
	,DSCNAME=name
	,INAREAL=length

Table 74. DFHTCT TYPE=LINE (continued)

```
,TRMTYPE=type
[,NPDELAY={0|number}]
[,OUTQ=symbolic-name]
[,POOL=YES]
[,QUEUEID=hexadecimal-number]
[,SIGNOFF={YES|NO}]
[,TCTUAL={0|length}]
[,TRMMODL=number]

TCAM SNA only
 [,TCAMFET=SNA]
```

ACCMETH=TCAM
Indicates that TCAM is to be used.

DSCNAME=name
Code this with the symbolic data set control name associated with the data set control information and its related communication line(s).

The same DSCNAME must be specified on the DFHTCT TYPE=SDSCI macro.

INAREAL=length
Code this with the message input area length.

- For an input line entry, the INAREAL value must be equal to or greater than the corresponding TCAM PCB buffer size.
- For an output line entry, the INAREAL value must be equal to or less than the corresponding TCAM PCB buffer size.

NPDELAY={0|number}
When used with a TCAM line, this parameter specifies the time interval that is to expire before control is passed to DFHTEP when a CICS task is not ready to accept a record from an input process queue.

If the CICS task issues a read before the time interval expires, processing continues normally and DFHTEP is not notified.

OUTQ=symbolic-name
This is required in all TCAM input process queue terminal control table line entries. The symbolic name identifies the corresponding TCAM output process queue TCTLE. Several input process queues may reference the same output process queue.

POOL=YES
Coding POOL=YES on the TCAM output process queue indicates that the TCAM POOL feature is supported for that TCAM line.

QUEUEID=hexadecimal-number
Specifies a unique user ID for the TCAM process queue. The ID is an unframed, 1-byte hexadecimal number (00 to FF) that is placed in the input and output line entry at TCTLEQID to provide queue identification while executing a user exit.

SIGNOFF={YES|NO}
Code this to control whether the terminal should be signed off if the terminal user is inactive for longer than the time limit defined to the external security manager. (If you are using RACF as your security manager, the time limit for each operator is set with the TIMEOUT operand of the ADDUSER command.)

YES
The terminal is signed off if the time limit is exceeded.

NO
The terminal is not signed off if the time limit is exceeded.

TCAMFET=SNA
This must be coded if TCAM SNA devices are to be used on this line. The same DFHTCT TYPE=LINE macro must not include specifications for TCAM SNA and non-SNA devices. Specifying TCAMFET=SNA allows TCAM SNA devices to be used in conjunction with the TRMTYPE/SESTYPE combination of operands in DFHTCT TYPE=TERMINAL to generate logical units. This operand is required if SNA support (for example, FMH) is to be supplied by BMS or by DFHDIP.

TCAMFET=SNA and TRMTYPE=TCAM are mutually exclusive on the same DFHTCT TYPE=LINE macro.

TCTUAL={0|length}
Indicates the length in bytes (0 through 255) of the user area (the process control information field or PCI) for all terminal entries (TCTTEs) associated with this line. Make it as small as possible. The TCT user area is initialized to zeros at system initialization. If you want fields of different (variable) lengths, you can specify the TCTUAL value can be specified in one or more TYPE=TERMINAL macro instructions for terminals associated with this line.

TRMMODL=number
Sets the default terminal model for terminals on this line. See the description of TRMMODL for DFHTCT TYPE=TERMINAL.

TRMTYPE=type
In a non-SNA environment, you can specify TRMTYPE=TCAM to define a TCAM-only terminal associated with the line. This allows terminals supported by TCAM to use the TCAM interface through CICS. If you specify TRMTYPE=TCAM, your message control program must handle device-dependent editing. CICS systems programs insert only NL characters. When you use TRMTYPE=TCAM, code the following parameters on the DFHTCT TYPE=LINE macro: ACCMETH=TCAM, DSCNAME, and INAREAL. QUEUEID, NPDELAY, and OUTQ are optional.

If either 3270 data stream or 2260 support is required under TCAM, specify the appropriate 3270 terminal type (for example, L3270, L3270P) or the appropriate 2260 terminal type (for example, L2260) in the TRMTYPE operand. This enables BMS to generate the correct data stream. TRMTYPE=TCAM should be used for all other terminals that require EBCDIC support. BMS supplies new line editing for those terminals specified in this way.

TRMTYPE=TCAM and TCAMFET=SNA are mutually exclusive on the DFHTCT TYPE=LINE macro.

In an SNA environment, specify TCAMFET=SNA on the DFHTCT TYPE=LINE macro, and code the correct TRMTYPE (see TRMTYPE=type and SESTYPE=session-type on page 471) on the DFHTCT TYPE=LINE macro. Code TRMTYPE=3767 for a TCAM-SNA device type 3767. TRMTYPE is also mandatory on the DFHTCT TYPE=LINE macro when a 3600 TCAM+SNA device is associated with the line.

TCAM devices—DFHTCT TYPE=TERMINAL:

Table 75. DFHTCT TYPE=TERMINAL

```
DFHTCT          TYPE=TERMINAL
                ,TRMIDNT=name
                [,ACCMETH=TCAM]
                [,ALTPGE=(lines,columns)]
                [,ALTSCRN=(lines,columns)]
                [,ALTSFX=number]
                [,BUFFER=buffer-size]
                [,DEFSCRN=(lines,columns)]
                [,ERRATT=
                     {NO|([LASTLINE][,INTENSIFY]
                        [,{BLUE|RED|PINK|GREEN|
                          TURQUOISE|YELLOW|NEUTRAL}]
                        [,{BLINK|REVERSE|
                          UNDERLINE}])}]
                [,FEATURE=(feature[,feature],...)]
                [,FF={NO|YES}]
                [,LASTTRM=LINE|POOL]
                [,NETNAME=
                    {name-specified-in-TRMIDNT|name}]
                [,PGESIZE=(lines,columns)]
                [,PGESTAT={AUTOPAGE|PAGE}]
                [,SIGNOFF={YES|NO}]
                [,TCTUAL=
                      {number-specified-in-TYPE=LINE|
                      number}]
                [,TIOAL={value|(value1,value2)}]
                [,TRANSID=
                      transaction-identification-code]
                [,TRMMODL=
                      {number-specified-in-TYPE=LINE|
                      number}]
                [,TRMPRTY={0|number}]
                [,TRMSTAT=
                      {TRANSACTION|(status[,status])}]
                [,TRMTYPE=
                    {type-specified-in-TYPE=LINE|type}]
                [,USERID=userid]

                TCAM SNA only

                [,BMSFEAT=
                     ([NOROUTE][,NOROUTEALL]
                     [,OBFMT][,OBOPID])]
                [,HF={NO|YES}]
                [,LDC={listname
                      |(aa[=nnn],bb[=nnn],
                         cc[=nnn],...)}]
                [,SESTYPE=session-type]
                [,VF={NO|YES}]
```

ACCMETH=TCAM
Indicates that the access method is TCAM.

ALTPGE=(lines,columns)
Indicates the page size to be used by BMS for this terminal entry when ALTSCRN has been selected as the screen size. The default is the value specified in ALTSCRN.

Unexpected results occur if the columns value of ALTPGE is different from that of ALTSCRN. The lines value of ALTPGE can usefully be less than that of ALTSCRN, perhaps to reserve the bottom line of the screen for error messages (see the ERRATT operand). The product of *lines* and *columns* must not exceed 32767.

ALTSCRN=(lines,columns)
Defines the 3270 screen size to be used for a transaction whose RDO PROFILE definition has the attribute SCRNSIZE(ALTERNATE). The default is the value specified in the DEFSCRN operand. The values that can be specified are:

Device		**Alternate screen size**
3276-1,	3278-1	(12,80)
3276-2,	3278-2	(24,80)
3276-3,	3278-3	(32,80)
3276-4,	3278-4	(43,80)
	3278-5	(27,132)
3279-2A,	3279-2B	(24,80)
3279-3A,	3279-3B	(32,80)

For 3287 and 3289 printers, the value specified must equal the buffer size of the particular device.

Note: No validity checking is performed on the screen size selected, and incorrect sizes may lead to unpredictable results.

For BSC devices, both the alternate and default screen sizes are determined by the device hardware. The alternate screen size is the maximum screen size.

For the 3290 display, both the default and alternate screen sizes are determined by the customer setup procedure. See the *IBM 3290 Information Panel Description and Reference* manual for further information.

If dual screen sizes are used, you can make CICS transactions use the alternate screen size by specifying SCRNSIZE(ALTERNATE) in the RDO PROFILE definition. If an application consists of pseudo conversations containing several linked transactions, provide PROFILE definitions for each of these transactions specifying SCRNSIZE(ALTERNATE) if the application uses alternate screen size.

ALTSFX=number
Code this with a 1-character numeric suffix (specified in the SUFFIX operand of the application program's DFHMSD TYPE={DSECT|MAP} macro). This suffix is appended by BMS to map set names if the screen size being used is the same value as the alternate screen size, that is, if SCRNSIZE(ALTERNATE) is specified in the transaction PROFILE definition (using RDO) or if the default and alternate screen size are the same. In this case, BMS map set selection routines attempt to load the map set with the suffix specified in the ALTSFX operand. If there is no such map set, or if the transaction uses the default screen size, BMS tries to load a map set suffixed with M or L. If this load fails, BMS tries to load an unsuffixed map set version.

BMSFEAT=([NOROUTE][,NOROUTEALL] [,OBFMT][,OBOPID])
Required for TCAM SNA only. Indicates which BMS features are used for this TCTTE.

NOROUTE
BMS is not to produce routed data for this terminal.

NOROUTEALL
BMS is not to include this terminal in the list of terminals to receive data in response to a BMS route request to all devices. This operand is never required, but may be coded for any terminal.

OBFMT
BMS is to support outboard formatting for this terminal. See the *CICS/OS/VS IBM 3650/3680 Guide* for more details.

OBOPID
CICS uses the outboard operator identifiers to support the BMS routing facilities required for this terminal.

BUFFER=buffer-size
Code this with the size of the receive buffer for the indicated terminal. The default value is BUFFER=0.

For 3600 BSC devices, BUFFER represents the maximum data length that the workstation may receive. The value specified must not be greater than the size of the host read buffers of either the 3601 or the workstation.

The unit of transmission from CICS to the BSC 3601 is a segment. If an application program issues a WRITE for a message longer than the value in BUFFER, CICS breaks the message down into as many segments as necessary. The length of each segment except the last is equal to the value of BUFFER. The length of the last or only segment (in the case of a single segment message) is the length of the data still to be sent.

DEFSCRN=(lines,columns)
Defines the 3270 screen size or 3270 printer page size to be used on this device when attached to a transaction or used by BMS for which SCRNSZE(DEFAULT) has been specified in the RDO PROFILE definition. The default is the value associated with the appropriate option in the TRMMODL operand, and is valid for dual screen-size terminals only.

The values that may be specified in the DEFSCRN operand are:

Device		**Screen size**
3278-1		(12,40)
3278-2		(24,80)
3276-3,	3278-3	(24,80)
3276-4,	3278-4	(24,80)
	3278-5	(24,80)
3279-2A,	3279-2B	(24,80)
3279-3A,	3279-3B	(24,80)

For BSC devices, both default and alternate screen sizes are determined by the terminal hardware. The default screen size is (24,80), except for the 3278-1 where it is (12,40).

ERRATT={NO|([LASTLINE][,INTENSIFY] [,{BLUE|RED|PINK|GREEN|TURQUOISE| YELLOW|NEUTRAL}]

[,{BLINK|REVERSE|UNDERLINE}])}
indicates the attributes that are to be associated with error messages displayed on this 3270 screen. This overrides the value of the ERRATT operand coded in a DFHTCT TYPE=INITIAL or TYPE=LINE macro.

For more details, see "Control section—DFHTCT TYPE=INITIAL" on page 441.

FEATURE=(feature[,feature],...)
Code this with the applicable features present in the 3270 terminals.

Further information on the features for 3270 devices can be found in the *CICS 3270 Data Stream Device Guide*.

AUDALARM
Code this for the audible alarm feature for a 3270 terminal.

BTRANS
Code this for the background transparency feature.

COLOR
Code this for the extended color feature on a 3270 terminal. This feature allows a color to be selected for each field or character.

COPY
Code this if the Copy Feature for a 3270 display or printer is included in the 3270 control unit.

EXTDS
Code this for 3270 data stream extensions. This option is implied if any one of the COLOR, HILIGHT, PS, BTRANS, SOSI, OUTLINE, QUERYALL, QUERYCOLD, or VALIDATION (3270 only) options is coded.

HILIGHT
Code this for the 3270 extended highlight facility, which enables fields or characters to be displayed in reverse-video, underline mode, or blink (3270 only).

KATAKANA
Code this for Katakana support. Katakana terminals do not have a lowercase character set, but have a Katakana character set instead. When coded for a Katakana terminal, all lowercase characters directed to the terminal from the following transactions are translated to uppercase:

CBRC	CEDC
CDBC	CEDF
CDBI	CEMT
CEBR	CEOT
CECI	CESN
CEDA	CEST
CEDB	

MSRCNTRL
Code this if the terminal, an 8775 or 3643, has a magnetic slot reader. This option is not valid for SCS printers. If coded for an SCS printer, an error message is raised but the option is not generated.

OUTLINE
Code this for the field outlining feature of the 5550.

PARTNS
Code this if the terminal, a 3290, is to support partitions.

PS
Code this for the 3270 programmed symbol (PS) facility. The facility enables up to six 191-character sets, with customer-defined and program-loaded fonts and codes, to be stored and accessed.

PTRADAPT
Code this for the 3275 printer adapter feature and corresponding 3284 Printer Model 3 on the 3275 Display Station. This feature makes the 3284 eligible for print requests through the program access key from the host 3275. A separate DFHTCT TYPE=TERMINAL macro cannot be coded for the 3284 Printer Model 3, because this printer shares the buffer of the 3275 Display Station.

Note: If output is created on the screen by DFHBMS or DFHMSD macro instructions with CTRL=PRINT, by BMS requests with the NLEOM option, or by the CMSG command, the contents of the screen are automatically copied to a 3270 printer, whether the program access key was hit or not.

QUERYALL
Code this for CICS to determine afresh the features the terminal has, each time the terminal is connected. The features that can be determined are:
BTRANS
COLOR
EXTDS
HILIGHT
MSRCNTRL
OUTLINE
PARTNS
PS
SOSI
VALIDATION

Any features you code explicitly are overridden by the values obtained by the QUERY.

You can find further guidance about using the SNA QUERY function on page 333.

QUERYCOLD
Code this for CICS to determine the features the terminal has when the terminal is connected for the first time after an initial or a cold start. The features that can be determined are:
BTRANS
COLOR
EXTDS
HILIGHT
MSRCNTRL
OUTLINE
PARTNS
PS
SOSI
VALIDATION

Any features you code explicitly are overridden by the values obtained by the QUERY.

You can find further guidance about using the SNA QUERY function on page 333.

SOSI
Code this to indicate that the device supports mixed EBCDIC and DBCS fields.

TRANSPARENCY
Code this if terminal data is not to be translated on a read or write, allowing the sending or receiving of all 256 bit combinations in a byte. This applies to 3270 devices with extended data stream support, and is implied by FEATURE=EXTDS. This is itself implied by setting any extended attribute feature.

UCTRAN
Code this for translation of lowercase data to uppercase in input data
streams. Translation can be overridden in the application program by coding the ASIS option on specific RECEIVE requests.

VALIDATION
Code this for the 3290 device validation feature, which allows fields to be defined as MANDATORY FILL or MANDATORY ENTER.

Information in the TCTTE provided by coding the following features is not used by CICS, but may be of use to applications programs requiring information on the features available on particular devices.

3270E
The device is one of the 3270 range having the alternate screen size facility (3276, 3278, 3279, 3287 or 3289). This option may not be coded for a 3287 printer attached to a 3271 or 3272 control unit.

APLKYBD
The 3270 device has the APL keyboard feature.

APLTEXT
The 3270 device has the APL text feature. This option may not be coded for a 3288 printer (with or without the TEXTPRINT option, below). The APLTEXT feature is used in conjunction with the TEXTKYBD and APLKYBD options.

DCKYBD
Code this for the typewriter keyboard and/or operator console keyboard for a 3270 display. Both uppercase and lowercase data can be transmitted with either of these keyboards.

SELCTPEN
Code this for the selector pen feature for a 3270 display.

TEXTKYBD
The 3270 device has the text-keyboard feature.

TEXTPRINT
A 3288 printer has the text-print feature. This option may be used in conjunction with the 3270E option to indicate that the text-print feature is used on a 3289 printer.

FF={NO|YES}
Indicates whether the terminal supports forms feed (FF). If FF=YES is coded, BMS uses this character when formatting output documents.

For 3270 displays and printers, FF=YES must be used in conjunction with the FORMFEED option in the BMS SEND commands. Use of form feed on display devices provides for a skip to a new page when the screen data is copied to a printer.

HF={NO|YES}
Required for TCAM SNA only. Indicates whether or not the horizontal forms feature is to be supported by the batch, batch data interchange, interactive, or LUTYPE4 logical unit. HF=NO overrides the HTAB=(*tab*,...) parameter in the DFHMSD macro.

LASTTRM={LINE|POOL}
Indicates a “last terminal” condition.

LINE
Applies to TCAM nonswitched line processing.

POOL
Applies to TCAM switched-line processing. It must be coded to identify the last terminal in the pool.

LDC=*listname*
Required for TCAM SNA only. This TCTTE points to a list of logical device codes (LDCs). The list specifies which LDCs are valid for this logical unit and, optionally, which device characteristics are valid for each LDC. The first LDC generated in this list is the default when CICS must choose a default LDC for a logical unit. For more information see “DFHTCT logical device codes: VTAM non-3270” on page 444.

listname
Code this with the name of the local LDC list or extended local LDC list to be associated with this logical unit. (This LDC list is generated by a DFHTCT TYPE=LDCLIST macro or by a series of DFHTCT TYPE=LDC macros for the extended local LDC list.)

NETNAME={name-specified-in-TRMIDNT|name}
The name must be the same as that used in the TCAM TERMINAL macro.

If you code a netname, ensure that it is unique to this terminal.

PGESIZE=(lines,columns)
Indicates the default page size for this terminal.

lines
Indicates the number of lines in the page.

columns
Indicates the number of characters in each line.

The product of *lines* and *columns* must not exceed 32767.

If PGESIZE is not coded, the following defaults are used:

TW33, TW35	(12,80)
3270 display model 1	(12,40)
3270 display model 2	(24,80)
3270 printer	(12,80)

For 3270 devices, PGESIZE defaults to the DEFSCRN value.

PGESTAT={AUTOPAGE|PAGE}
Indicates whether the device is to use autopaging or not. Autopaging means that BMS multiple page messages are printed continuously, without operator intervention. This is what is normally required for a printer. (Contrast the requirement for multiple page messages, displayed on a 3270-type display, when the operator wants to finish reading a page, before requesting the next page to be delivered.)

Only BMS SEND commands with the PAGING option use autopaging. BMS SEND with TERMINAL or SET, does not use autopaging.

AUTOPAGE
Specify this for printers.

PAGE
Specify this for displays.

SESTYPE=session-type
See TRMTYPE=type and SESTYPE=session-type following.

SIGNOFF={YES|NO}
Code this to control whether the terminal should be signed off if the terminal user is inactive for longer than the time limit defined in the external security manager. (If you are using RACF as your security manager, the time limit for each operator is set with the TIMEOUT operand of the ADDUSER command.)

YES
The terminal is signed off if the time limit is exceeded.

NO
The terminal is not signed off if the time limit is exceeded.

TCTUAL={number-specified-in-TYPE=LINE|number}
Indicates the length, in bytes (0 through 255), of the user area (the process control information field or PCI) for the terminal entry (TCTTE) associated with this terminal. Make it as small as possible. The TCT user area is initialized to zero at system initialization.

Use the TCTUAL operand of the DFHTCT TYPE=TERMINAL macro if fields of different (variable) lengths are desired for terminals associated with this line. In any case, the PCI field is generated for each terminal after the last terminal entry of the last line. The address of the PCI field is located at TCTTECIA; the length is located at TCTTECIL.

TIOAL=value
Indicates the terminal input/output area length to be passed to a transaction (that is, the length of the input area of the terminal), and should be set to a value that is slightly larger than the average input message length for the terminal. The default value is the message input area length generated from the INAREAL operand of the associated line entry. You can let it default for local 3270 terminals, but do not omit the TIOAL parameter for remote 3270 terminals.

Note: If automatic transaction initiation is used, the minimum TIOAL that can be specified is 1 byte.

TRMIDNT=name
Code this with a unique 4-character symbolic identification of each terminal. The identification supplied is left-justified and padded with blanks to four characters if less than four characters are supplied.

The name and the destination identification in the destination control table, when applicable to terminal destinations, must be the same.

The value CERR is reserved, as this is the identification generated for the error console.

TRANSID=transaction-identification-code
When using TCAM, TRANSID applies only to TCTLEs associated with the TCAM output queue.

TRMMODL={model-specified-in-TYPE=LINE | number}
When coded on the DFHTCT TYPE=LINE macro, the TRMMODL parameter sets the default model number of the terminals associated with this line.

If models vary on a line, the DFHTCT TYPE=TERMINAL macro can have a TRMMODL parameter associated with it which is different from that specified in the DFHTCT TYPE=LINE macro. This overrides the DFHTCT TYPE=LINE macro instruction for that DFHTCT TYPE=TERMINAL macro.

This operand must be coded on either the DFHTCT TYPE=LINE or TYPE=TERMINAL macro if you are defining a component of the 3270 Information Display System.

1 Code this for the 3270 Model 1 displays and printers with a default screen or buffer size of 12x40 (480 bytes or characters) (for example, 3277 Model 1). TRMMODL=1 is the default for 3270 Model 1 printers and displays. Also code this for 3275 Display Station Model 11 support.

2 Code this for the 3270 displays and printers with a default screen or buffer size of 24x80 (1920 bytes or characters) (for example, 3278 Model 2). TRMMODL=2 is the default for the 3286 printer in 3270 compatibility mode. Also code this for 3275 Display Station Model 12 support.

TRMPRTY={0 | number}
Establishes the terminal priority. This decimal value (0 through 255) is used in establishing the overall transaction processing priority. (Transaction processing priority is equal to the sum of the terminal priority, transaction priority, and operator priority, and must not exceed 255.)

TRMSTAT={TRANSACTION | (status,...)}
Code this with the types of activity that may occur at a given terminal. This terminal status is initially set in the TCTTE and is a combination of the processing status and the service status.

TRANSACTION
A terminal with TRANSACTION status is used in the processing of transactions such as inquiries or order entries. A display station or a hard-copy terminal to which no messages are sent without a terminal request and through which transactions are entered is a TRANSACTION terminal.

INPUT
Indicates a terminal that can send messages to, but cannot receive messages from, CICS.

Note: System messages may be routed to an input terminal under conditions such as invalid transaction identification and ATP batch count. This causes DFHTACP to be scheduled. To handle this situation, code a DFHTEP to perform any action that the user requires.

'OUT OF SERVICE'
Indicates a terminal that can neither receive messages nor transmit input. Such terminals are not polled by CICS. The 'OUT OF SERVICE' parameter can be used in combination with any status setting.

Any terminal except the master terminal can be designated as 'OUT OF SERVICE'. When appropriate, the terminal can be placed in service by the master terminal and polling is resumed.

RECEIVE
Indicates a terminal to which messages are sent but from which no input is allowed. An example of this type of terminal is one that is located in a remote location, such as a warehouse, and is unattended, but may receive messages. Automatic transaction initiation is implemented as for TRANSCEIVE, below.

TRANSCEIVE
A terminal with TRANSCEIVE status is a TRANSACTION terminal to which messages are sent automatically. The automatic transaction initiation, either by transient data control or interval control, sets a condition in an appropriate terminal control table terminal entry. If the terminal status is TRANSCEIVE and if there is no transaction at the terminal, terminal control initiates the user-defined task. This task is expected to send messages to the terminal.

TRMTYPE=type and SESTYPE=session-type
In a non-SNA environment, the TRMTYPE operand can be used to specify the terminal type:

- If the terminal type has not already been specified in the DFHTCT TYPE=LINE macro, or
- To override the type specified in that macro.

In an SNA environment, support for TCAM SNA logical units is generated by coding the appropriate TRMTYPE/SESTYPE combinations. However, TCAM does not support the IBM 3650 Retail Store System or the IBM 3600 Finance Communication System.

USERID=userid
Code this to specify a user identifier for devices such as printers that are unable to sign on using CESN. (You can also specify USERID for a display device, in which case the display is permanently signed on. Operators are unable to sign on.) Code this operand if you want to use preset security with this device. All access to protected resources depends on USERID.

The userid is referred to in security error messages, security violation messages, and the audit trail. It must be defined to the security manager.

Userid must be a unique 1- to 8-character user identification. (A-Z 0-9 # $ and @ are acceptable characters.)

VF={NO|YES}
Required for TCAM SNA only. Indicates whether or not the vertical form feature is to be supported by the batch, batch data interchange, interactive, or LUTYPE4 logical unit. VF=NO overrides the VTAB=(*tab*,...) parameter in the BMS map definition.

TCAM TCT example: This example defines a local line with two 2260 terminals, and a remote line with three 3270 terminals.

DFHTCT example: TCAM

Note: The TCTTEs for the local 2260 terminals and the remote 3270 terminals are used for both input and output processing.

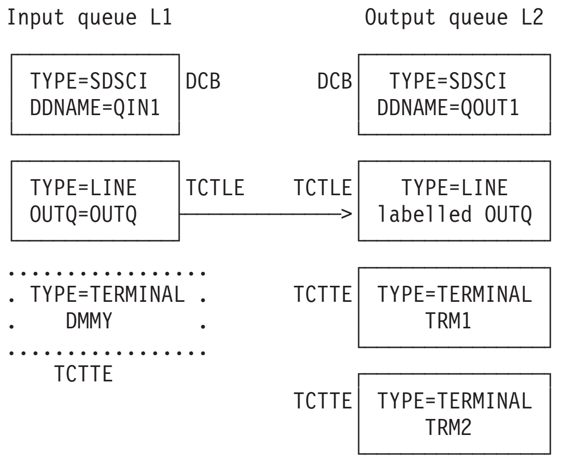


Figure 58. Local line with two 2260 terminals

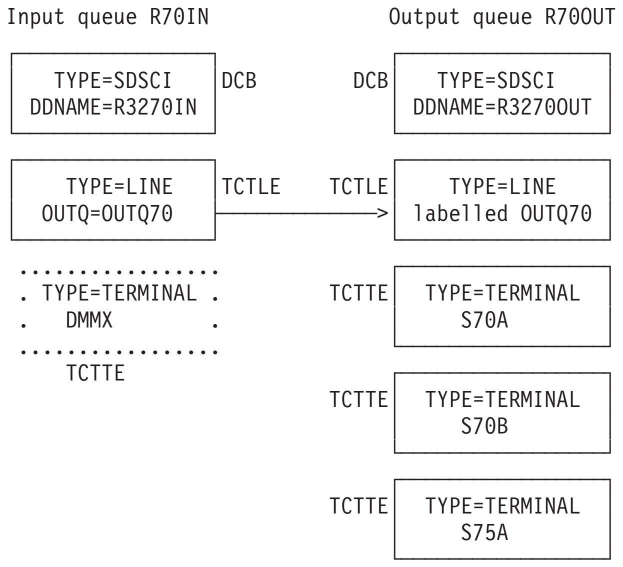


Figure 59. Remote line with three 3270 terminals

Figure 60 on page 473 shows the coding for the configuration shown in Figure 58 and Figure 59.

```
       DFHTCT TYPE=INITIAL,                                    *
              SUFFIX=TV
       DFHTCT TYPE=SDSCI,           DCB for input queue for    *
              DEVICE=TCAM,          local 2260 line            *
              DSCNAME=L1,                                      *
              DDNAME=QIN1,                                     *
              OPTCD=WU,                                        *
              MACRF=R,                                         *
              RECFM=U,                                         *
              BLKSIZE=500
       DFHTCT TYPE=SDSCI,           DCB for output queue for   *
              DEVICE=TCAM,          local 2260 line            *
              DSCNAME=L2,                                      *
              DDNAME=QOUT1,                                    *
              OPTCD=WU,                                        *
              MACRF=W,                                         *
              RECFM=U,                                         *
              BLKSIZE=500
       DFHTCT TYPE=LINE,            Input queue for local 2260 *
              ACCMETH=TCAM,         line                       *
              QUEUEID=F1,                                      *
              INAREAL=500,                                     *
              TRMTYPE=L2260,                                   *
              DSCNAME=L1,                                      *
              OUTQ=OUTQ
       DFHTCT TYPE=TERMINAL,        Dummy                      *
              TRMIDNT=DMMY,                                    *
              TRMPRTY=32,                                      *
              LASTTRM=LINE
OUTQ   DFHTCT TYPE=LINE,            Output queue for local 2260 *
              ACCMETH=TCAM,         line                       *
              QUEUEID=F0,                                      *
              INAREAL=500,                                     *
              TRMTYPE=L2260,                                   *
              DSCNAME=L2
       DFHTCT TYPE=TERMINAL,        First local 2260 terminal  *
              TRMIDNT=TRM1,         on the line                *
              TRMPRTY=32
       DFHTCT TYPE=TERMINAL,        Second and last local 2260 *
              TRMIDNT=TRM2,         terminal on the line       *
              LASTTRM=LINE,                                    *
              TRMPRTY=32
```

Figure 60. Coding for TCAM TCT example (Part 1 of 3)

```
       DFHTCT TYPE=SDSCI,           DCB for input queue for    *
              DEVICE=TCAM,          remote 3270 line           *
              DSCNAME=R70IN,                                   *
              DDNAME=R3270IN,                                  *
              OPTCD=WU,                                        *
              MACRF=R,                                         *
              RECFM=U,                                         *
              BLKSIZE=500
```

Figure 60. Coding for TCAM TCT example (Part 2 of 3)

DFHTCT example: TCAM

```
       DFHTCT TYPE=SDSCI,          DCB for output queue for       *
              DEVICE=TCAM,         remote 3270 line               *
              DSCNAME=R70OUT,                                     *
              DDNAME=R3270OUT,                                    *
              OPTCD=WU,                                           *
              MACRF=W,                                            *
              RECFM=U,                                            *
              BLKSIZE=500
       DFHTCT TYPE=LINE,           Input queue for remote 3270    *
              ACCMETH=TCAM,        line                           *
              INAREAL=500,                                        *
              DSCNAME=R70IN,                                      *
              OUTQ=OUTQ70,                                        *
              TRMTYPE=3277
       DFHTCT TYPE=TERMINAL,       Dummy                          *
              TRMIDNT=DMMX,                                       *
              TRMPRTY=32,                                         *
              LASTTRM=LINE
OUTQ70 DFHTCT TYPE=LINE,           Output queue for remote 3270   *
              ACCMETH=TCAM,        line                           *
              INAREAL=500,                                        *
              DSCNAME=R70OUT,                                     *
              TRMTYPE=3277
       DFHTCT TYPE=TERMINAL,       First remote 3270 terminal     *
              TRMPRTY=32,          on the line                    *
              TRMIDNT=S70A
       DFHTCT TYPE=TERMINAL,       Second remote 3270 terminal    *
              TRMPRTY=32,          on the line                    *
              TRMIDNT=S70B
       DFHTCT TYPE=TERMINAL,       Third and last remote 3270     *
              TRMPRTY=32,          terminal on the line           *
              TRMIDNT=S75A,                                       *
              LASTTRM=LINE
       DFHTCT TYPE=FINAL
       END
```

Figure 60. Coding for TCAM TCT example (Part 3 of 3)

Remote terminals for transaction routing

CICS can communicate with other systems that have similar communication facilities. This kind of communication is known as **CICS intercommunication**. You can read about it in the *CICS Intercommunication Guide*.

When you are concerned with resource definition, the system where the TCT is installed is the **local** system. The system that is being defined in the TCT is the **remote** system.

Transaction routing enables terminals in one CICS system to invoke transactions in another CICS system. You can use transaction routing between systems connected by MRO or by an LUTYPE 6.2 link.

Remote definitions for terminals for transaction routing: There are two possible methods of defining the terminals using macros. (There is another method, only possible using RDO, called ‘shipping terminal definitions’. See “Terminals for transaction routing” on page 291.) The two macro methods are:

- Method 1:

```
DFHTCT TYPE=REGION, ...   (one for each region)

DFHTCT TYPE=SDSCI, ...    (for non-VTAM terminals
                           only; ignored for remote
                           definitions)
```

```
DFHTCT TYPE=LINE, ...     (for non-VTAM terminals
                           only)

DFHTCT TYPE=TERMINAL, ... (for non-VTAM: one for
                           each terminal)
```

- Method 2:

```
DFHTCT TYPE=REMOTE, ...   (one for each terminal)
```

In method 1, you can use copybooks to include the same source code in the TCTs for local and remote systems. The information not needed (that is, the whole of the TYPE=SDSCI macro, and some of the TYPE=LINE and TYPE=TERMINAL macros) is discarded for remote entries.

Method 2 employs a single DFHTCT TYPE=REMOTE macro.

CICS decides whether to create a remote or a local definition on the basis of the SYSIDNT operand, either on the TYPE=REGION, or on the TYPE=REMOTE macro. This is compared with the SYSIDNT operand in DFHTCT TYPE=INITIAL. If they are the same, the definition(s) are local. If they are different, the definition(s) are remote.

These terminals cannot use transaction routing and therefore cannot be defined as remote:
- IBM 7770 or 2260 terminals
- MVS system consoles
- Pooled TCAM terminals
- Pooled 3600 or 3650 Pipeline Logical Units

Note: BTAM is not supported in this release. To gain access to this release of CICS from BTAM terminals, define the BTAM terminals in an earlier release of CICS, and use transaction routing to gain access to the current release. BTAM terminals must be defined as remote resources in this release, and as local resources in the earlier release. More information about the definition of local BTAM terminals can be found in the documentation for earlier releases of CICS.

Remote terminals, method 1—DFHTCT TYPE=REGION: The DFHTCT TYPE=REGION macro introduces information about the named region. The information consists of DFHTCT TYPE=LINE and TYPE=TERMINAL macros. These must follow the DFHTCT TYPE=REGION macro. For a remote region, the DFHTCT TYPE=LINE macro does not generate a TCT line entry (TCTLE). Every terminal that participates in transaction routing must be defined. Only certain DFHTCT macro types and operands are relevant in remote region definitions; all others are ignored. The operands that are relevant are those listed in "Remote terminals, method 2—DFHTCT TYPE=REMOTE" on page 477.

Table 76. DFHTCT TYPE=REGION

DFHTCT	TYPE=REGION
	,SYSIDNT={name | LOCAL}

SYSIDNT={name | LOCAL}
Indicates the 4-character name of the system or region whose information

starts or resumes here. SYSIDNT=LOCAL can be specified to indicate that the TYPE=TERMINAL definitions following it refer to the home region, as do all definitions preceding the first DFHTCT TYPE=REGION macro. The name of the home region (that is, the region in which this terminal control table is used) is the value of the SYSIDNT operand of the DFHTCT TYPE=INITIAL macro. The name can instead be that of a previously defined MRO link or ISC link.

Remote terminals, method 1—DFHTCT TYPE=TERMINAL:

Note: The DFHTCT TYPE=LINE macro and the additional operands of the DFHTCT TYPE=TERMINAL macro are valid, but are ignored if the SYSIDNT operand on the preceding DFHTCT TYPE=REGION macro indicates a remote region. (For details of the DFHTCT TYPE=LINE and DFHTCT TYPE=TERMINAL macros, see "Sequential devices" on page 452 and "TCAM—DFHTCT TYPE=SDSCI, LINE, TERMINAL" on page 457.)

Table 77. DFHTCT TYPE=TERMINAL

```
DFHTCT          TYPE=TERMINAL
                ,ACCMETH=access-method
                ,SYSIDNT=name
                ,TRMIDNT=name
                ,TRMTYPE=terminal-type
                ,[RMTNAME=
                {name-specified-in-TRMIDNT|name}]
```

ACCMETH=access-method
Code this with the access method of the remote terminal.

RMTNAME={name-specified-in-TRMIDNT|name}
Specifies the 1- to 4-character name by which the terminal is known in the system or region that owns the terminal (that is, in the TCT of the **other** system). If this operand is omitted, the name in the TRMIDNT operand is used.

SYSIDNT=name
Indicates the 4-character name of the system or region that owns this terminal. This may be the local system or region (that is, the name defined in the TYPE=INITIAL macro), in which case the TCT entry created is a local definition. It may be the name of a different system or region, in which case the TCT entry created is a remote definition. This SYSIDNT must be the same as the SYSIDNT on the TYPE=REGION macro that precedes this macro.

TRMIDNT=name
Specifies the 1- to 4-character name by which the terminal is known in **this** system (that is, in the local system that owns this TCT, and that owns the transactions).

TRMTYPE=terminal-type
Code this with the terminal type. For details, see "Sequential devices" on page 452 and "TCAM—DFHTCT TYPE=SDSCI, LINE, TERMINAL" on page 457.

Remote terminals, method 2—DFHTCT TYPE=REMOTE

Terminal entries for remote systems or regions can be defined to CICS using the DFHTCT TYPE=REMOTE macro as an alternative to defining them using DFHTCT TYPE=TERMINAL macro instructions in conjunction with a DFHTCT TYPE=REGION macro.

The expansion of the DFHTCT TYPE=REMOTE macro is independent of the region currently referenced.

Note: If the SYSIDNT operand indicates that the **home** region owns the terminal, all the operands of the DFHTCT TYPE=TERMINAL macro become valid on the DFHTCT TYPE=REMOTE macro and have the same meaning as for TYPE=TERMINAL. However, if (as is normally the case) the SYSIDNT operand indicates a remote region, the additional operands of DFHTCT TYPE=TERMINAL are valid on the DFHTCT TYPE=REMOTE macro, but are ignored. (For details of the DFHTCT TYPE=TERMINAL macro, see "Sequential devices" on page 452 and "TCAM—DFHTCT TYPE=SDSCI, LINE, TERMINAL" on page 457.)

Table 78. DFHTCT TYPE=REMOTE

```
          DFHTCT          TYPE=REMOTE
                          ,ACCMETH=access-method
                          ,SYSIDNT=name
                          ,TRMIDNT=name
                          ,TRMTYPE=terminal-type
                          ,[RMTNAME=
                          {name-specified-in-TRMIDNT|name}]
```

ACCMETH=access-method
Code this with the access method of the remote terminal.

RMTNAME={name-specified-in-TRMIDNT|name}
Specifies the 1- to 4-character name by which the terminal is known in the system or region that owns the terminal (that is, in the TCT of the **other** system). If this operand is omitted the name in the TRMIDNT operand is used.

SYSIDNT=name
Specifies the name of the system or region that owns this terminal. The name must be the same as that used in the SYSIDNT operand of a previous TYPE=SYSTEM macro, or the TYPE=INITIAL macro.

TRMIDNT=name
Specifies the 1- to 4-character name by which the terminal is known in **this** system (that is, in the local system that owns this TCT, and that owns the transactions).

TRMTYPE=terminal-type
Code this with the terminal type. For details, see "Sequential devices" on page 452 or "TCAM—DFHTCT TYPE=SDSCI, LINE, TERMINAL" on page 457.

DFHTCT: CICS terminals list

This release of CICS is able to communicate with almost all previously supported terminals, either **directly** or **indirectly**, as described below.

New or current terminals are directly supported by CICS Transaction Server for OS/390 if they conform to the VTAM interface.

DFHTCT TYPE=REGION: remote terminals

Because BTAM and TCAM/ACB are no longer supported, certain device types are not able to connect directly to CICS Transaction Server for OS/390. CICS Transaction Server for OS/390 provides support for such devices indirectly, through transaction routing from an earlier release of CICS.

The BTAM terminals affected are listed in Table 79 and described in more detail in Details of BTAM terminals supported.

If you are a user of TCAM/ACB, you have a choice between:

1. Migrating your TCAM/ACB terminals to ACF/VTAM V3R3 at the same time as migrating to CICS Transaction Server for OS/390.

   Note: All terminals supported by TCAM/ACB are also supported by VTAM.
2. Leaving your TCAM/ACB terminals on a coexisting earlier CICS system, and transaction routing to this release of CICS for the lifetime of the TCAM/ACB terminals.

Table 79 summarizes how terminals are supported in CICS.

Table 79. IBM terminals and system types supported by CICS Transaction Server for OS/390.

Directly supported by CICS Transaction Server for OS/390 using VTAM	**Supported using transaction routing through earlier CICS (BTAM terminals)**
*3101 Display Terminal	1050 Data Communication System
*3230 Printer	2740 Communication Terminal
3268 Printer	2741 Terminal controller
*3270 Information Display System	2770 Data Communication System
3270 PC	2780 Data Transmission Terminal
3270 PC/G	2980 General Banking Terminal System
3270 PC/GX	3660 Supermarket System
3287 Printer	3735 Programmable Buffered Terminal
*3600 Finance Communication System	3740 Data Entry System
3630 Plant Communication System	3780 Data Communications Terminal
3640 Plant Communication System	5100 Portable Computer
3650 Retail Store System	5110 Portable Computer
3680 Programmable Store System	5230 Data Collection System
3730 Distributed Office Communication System	5260 Retail System
*3767 Communication Terminal	Communicating Magnetic Card
*3770 Data Communication System	Selectric Typewriter (CMCST)
3790 Communication System	Office System/6
4300 Processors	Series/1
*4700 Finance Communication System	System/3
*5280 Distributed Data System	System/7
*5520 Administrative System	System/23
5550 Administrative System	
*5937 Rugged Terminal	
*6670 Information Distributor	
8100 Information System	
8775 Display Terminal	
8815 Scanmaster	
*Displaywriter	
*Personal Computer, PS/2, PS/55	
*System/32	
*System/34	
*System/36	
*System/38	

Table 79. IBM terminals and system types supported by CICS Transaction Server for OS/390. (continued)

Directly supported by CICS Transaction Server for OS/390 using VTAM	Supported using transaction routing through earlier CICS (BTAM terminals)
AS/400	
*System/370™ (inc 303x, 308x, and 3090™ processors)	
Teletypewriter Exchange Service (TWX 33/35)	
World Trade Typewriter Terminal (WTTY)	

Note: * These terminals are also supported by BTAM.

Details of BTAM terminals supported: This section provides a detailed account of the BTAM terminals supported by CICS. For information on how to connect them to CICS Transaction Server for OS/390 using transaction routing, see "Remote terminals for transaction routing" on page 474. The following terms or abbreviations are used:

local	channel or adapter attached
s/s	start/stop transmission
SDLC	synchronous data link control
sw	switched
BSC	binary synchronous
nonsw	nonswitched communications

Table 80. IBM BTAM terminals and subsystems supported by transaction routing from an earlier release of CICS

Terminal /System Type	Units	Attachment	Notes
1050	1051, 1052, 1053, 1056	s/s sw or nonsw	
2740		s/s sw or nonsw	
2741		s/s sw or nonsw	
2770	2772, 0545, 1053, 2213, 2265, 2502	BSC sw or nonsw	1 on page 480
2780		BSC sw or nonsw	2 on page 480
2980	2972, 2980	BSC nonsw	3 on page 480
3660	3651, 3661	BSC sw	4 on page 480
3735		BSC sw	
3740	3741	BSC sw or nonsw	6 on page 480
3780		BSC sw or nonsw	7 on page 480
5100		s/s supported as 2741	4 on page 480
5110		BSC attached as 2770	4 on page 480
5230	5231	BSC supported as 3741	
5260	5265	BSC attached as 3741	
CMCST		Supported as 2741	
Office System/6	6640, 6670	Supported as 2770	
Series/1		Attached as System/3; supported as 3650 Pipeline (VTAM) or 3790 (full function LU)	4 on page 480
System/3	5406, 5408, 5410, 5412, 5415	BSC sw or nonsw	4 on page 480

Table 80. IBM BTAM terminals and subsystems supported by transaction routing from an earlier release of CICS (continued)

Terminal /System Type	Units	Attachment	Notes
System/7	5010	s/s or BSC, sw or nonsw	44, 5
System/23		Attached as 3741 (BSC)	

Notes:

1. CICS 2770 support includes optional 2772 features #3650 (EBCDIC Transparency), #9936 (WACK response), #1490 (Buffer Expansion), #1910 (Conversational Mode), #1340 (Automatic Answering), #4610 (Identification), #6310 (Security Identification), and #5010 (Multipoint).
2. 6-bit transcode is not supported. Support includes optional features #8030 (EBCDIC transparency) and #1340 (Automatic Answering), or #5020 (Multipoint).
3. 2980 support is for 2972 Model 8 (RPQ 858160) or Model 11 (RPQ 8582311) with 2980 Model 1 (RPQ 835504), Model 2 (RPQ 835505) or Model 4 (RPQ 858147), including options RPQ 858188 (Auditor key for Model 2) and RPQ 858165 (Buffer Expansion).
4. Devices and features supported by a system or programmable controller are generally transparent to CICS. In some cases CICS provides specific device support, in which case the units are listed.
5. Nonswitched as a multipoint device: System/7 Remote IPL is supported. Switched and as a Point-to-Point Device: Remote IPL is not supported.
6. 3740 support includes optional features #7850 (Terminal Identification), #1685 (Multipoint), #5450 (Operator ID Card Reader), and #1680 (Expanded Communications).
7. 3780 support includes features #3601 (EBCDIC Transparency), #9936 (WACK), #5010 (Multipoint), or #7651 (Switched), #1601 (Component Selection).

The terminals listed above are all BTAM and are not supported by VTAM. Other terminals are supported by both BTAM and VTAM, and can be used for transaction routing. For a detailed list of all VTAM terminals, see the “Devices supported” on page 336. Some BTAM terminals cannot be used for transaction routing from an earlier release of CICS to CICS Transaction Server for OS/390. The terminals are:

2260
2265
7770

VTAM terminals: For a detailed list of VTAM-supported terminals, and how to define them to CICS, see “Devices supported” on page 336.

TLT—terminal list table

A terminal list table (TLT) generated by the DFHTLT macro instruction allows terminal and operator identifications to be grouped logically. A TLT:

- Is **mandatory** for use by CEST (the supervisor terminal transaction), to define and limit the effective range of the operation. For example:

  ```
  CEST SET TERMINAL(*) SUPRID(CG) OUTSERVICE
  ```

  sets all terminals defined in DFHTLTCG out of service.
- May be used by CEST or CEMT (the master terminal transaction) to apply an operation to a predetermined group of terminals. (For a CEST operation, this TLT must define a subset of the TLT specified by SUPRID.) For example, each of the following commands:

  ```
  CEST SET TERMINAL(*) SUPRID(CG) CLASS(EM) INSERVICE
  CEMT SET TERMINAL(*) CLASS(EM) INSERVICE
  ```

  sets all terminals defined in DFHTLTEM in service.
- May be used singly or in combination with other TLTs to provide predefined destinations for message switching. For example:

  ```
  CMSG ROUTE=PG,'PRODUCTION MEETING AT 11.00 IN
             ROOM 2124',SEND
  ```

  sends a message to all terminals or operators defined in DFHTLTPG.

The same TLT can be used for message switching and for supervisory or master terminal functions. For example, a TLT defining the terminals that are under control of a supervisory terminal could also be used as a destination list for sending messages to those terminals.

For some logical units, logical device code (LDC) mnemonics (that may be associated with each table entry), are used for message switching and are ignored for master and supervisory terminal operations.

In an intercommunication network, all the terminals in a terminal list table must be owned by the system on which the table is used.

Elements of DFHTLT

The following macros define the TLT entries:

- Control section—DFHTLT TYPE=INITIAL
- Entries in terminal list table—DFHTLT TYPE=ENTRY
- End of terminal list table—DFHTLT TYPE=FINAL (see page 371)

Control section—DFHTLT TYPE=INITIAL

The DFHTLT TYPE=INITIAL macro establishes the entry point and the address of the start of the terminal list table being defined.

Table 81. DFHTLT TYPE=INITIAL

DFHTLT	TYPE=INITIAL
	[,LDC=aa]
	[,SUFFIX=xx]

Note: For general information about TYPE=INITIAL macros, see "TYPE=INITIAL (control section)" on page 370.

LDC=aa
Code this with a 2-character logical device code (LDC) mnemonic. This is associated with every logical unit identification, except for those for which an LDC mnemonic is specified on a DFHTLT TYPE=ENTRY macro.

SUFFIX=xx
The module name of the TLT is DFHTLT*xx*, where *xx* is a 1-or 2-character suffix. This provides unique identification for each TLT used. Because the names TLTBA, TLTBB, TLTBC, and TLTEA are used within the TLT, suffixes BA, BB, BC, and EA must not be used.

A TLT must have a suffix to be used by the message switching transaction, CMSG.

Entries in terminal list table—DFHTLT TYPE=ENTRY

Entries are coded in the TLT as follows:

Table 82. DFHTLT TYPE=ENTRY

```
DFHTLT        TYPE=ENTRY
              ,TRMIDNT=([termid-1[*ldc-1]][/opid-1]
                    [,termid-2[*ldc-2][/opid-2],...])
```

TYPE=ENTRY
Code this if one or more entries are to be generated in this table, up to a maximum of 1000 entries.

TRMIDNT=([termid-1[*ldc-1]] [/opid-1][, termid-2[*ldc-2][/opid-2],...])
Code this with a list of start-stop and BSC terminal, logical unit, and operator identifications. A logical unit identification can be qualified by an LDC mnemonic.

termid
Indicates a 1- to 4-character start-stop or BSC terminal or logical unit identification.

Note: A 3614 attached to a communications controller may be used in master or supervisory terminal operations but should not be used in message switching operations. (A 3614 is not valid for a message destination.)

ldc
Indicates a 2-character LDC mnemonic, which must be preceded by an asterisk (*) and is only used following the 'termid' parameter to which it is appended.

opid
Indicates a 1- to 3-character operator identification that must be preceded by a slash (/).

Any terminal or operator identification specified should also be specified in the TRMIDNT operand of the DFHTCT macro and in your external security manager, respectively. (If you employ RACF, you use the OPIDENT operand of the ADDUSER command to record the identification for each operator.) Any LDC mnemonic specified should also be specified in the LDC operand of the DFHTCT TYPE=LDC and DFHTCT TYPE=TERMINAL macros.

Supervisory and master terminal functions use the terminal and logical unit identifications included in the TLT, but ignore all references to LDC mnemonics and operator identifications.

DFHTLT example

Figure 61 shows how to create a terminal list table.

```
Example 1

DFHTLT TYPE=INITIAL,                               *
       SUFFIX=AA
DFHTLT TYPE=ENTRY,                                 *
       TRMIDNT=(NYC,CHI,LA,WDC)
DFHTLT TYPE=ENTRY,                                 *
       TRMIDNT=SF
DFHTLT TYPE=ENTRY,                                 *
       TRMIDNT=(BSTN/OP1,ATL/OP5,/OP9,DNVR)
DFHTLT TYPE=ENTRY,                                 *
       TRMIDNT=/OP6
DFHTLT TYPE=FINAL
END

Example 2

DFHTLT TYPE=INITIAL,                               *
       SUFFIX=XX
DFHTLT TYPE=ENTRY,                                 *
       TRMIDNT=(NYC,T361*LP,T362*LP/OP1)
DFHTLT TYPE=ENTRY,                                 *
       TRMIDNT=(T363/OP2,T364/OP5,T365)
DFHTLT TYPE=FINAL
END
```

Figure 61. Terminal list table—example

TST—temporary storage table

| **Note:** Although the DFHTST macro is supported by CICS Transaction Server for
| OS/390, it is recommended that you use resource definition online (RDO) to
| define the equivalent TSMODEL definitions. The TSMODEL resource
| definition supports attributes that are not supported by the DFHTST macro;
| for example, the LOCATION(MAIN/AUXILIARY) attribute.

| The temporary storage table (TST) is a list of generic names (or prefixes) used to identify sets of temporary storage queues. Generic names are formed from the leading characters of the appropriate queue names, and can be up to seven characters long.

- The generic names coded on a DFHTST TYPE=RECOVERY macro identify queues for which CICS provides backout of changes in the event of transaction failure or protection against system failure.
- The generic name coded on a DFHTST TYPE=REMOTE macro identifies queues for which CICS routes the temporary storage request to a remote CICS region or TS server, unless the remote system name (SYSIDNT) is the same as that of the local CICS. If SYSIDNT is the same name as the local CICS, the queues specified by the DATAID option are treated by CICS as local queues.
- The generic name coded on a DFHTST TYPE=LOCAL macro identifies queues as local queues that reside in the CICS region in which the TST is installed.
- The generic name coded on a DFHTST TYPE=SECURITY macro identifies queues for which resource security checking is required.

Notes:

1. DATAIDs using all eight characters define unique temporary storage queue names.
2. # You cannot specify a generic name on a TYPE=SHARED entry, which does not
# have a DATAID parameter.

Choose a naming convention for queue names that enables you to define many queues with only a few generic names. This reduces considerably the task of TST definition.

Note: CICS searches the TST for the first prefix that satisfies the particular search criteria. For example, if CICS searches for temporary storage queue ABCDEFGH, and the TST contains prefix A followed by prefix AB, A is selected. To avoid this, define the less-generic entries to the TST before any more-generic entries, so that the first to be found is the least generic of all possible matches.

Note that when CICS is looking for DATAIDs to match against a TS queue name, it searches only the types of entry in which it is interested for that particular search. CICS searches:

- Local *and* remote entries when determining whether a queue is remote. Thus, local and remote entries are regarded as one search category when CICS is matching a queue name against generic names.
- Recovery and remote entries when determining whether a queue is recoverable. However, if the leading characters of a queue name match **both** TYPE=RECOVERY and TYPE=REMOTE generic names, TYPE=REMOTE takes precedence, and the recovery option must be redefined in the local region in which the queue resides. (Queues in a shared TS pool cannot be recoverable.)

- Security entries only when determining whether a queue is subject to security.

When a task modifies temporary storage data designated as recoverable, the data is protected from modification by a concurrent task by enqueuing on the queue name. The queue name is not dequeued until the task terminates or issues a task syncpoint request to designate the end of a logical unit of work. At this time a log record is written to the system log data set to provide external information sufficient to recover the data if the system subsequently terminates abnormally.

Elements of DFHTST

You can use these macros to define the TST entries:

- Control section—DFHTST TYPE=INITIAL
- Recoverable temporary storage—DFHTST TYPE=RECOVERY
- Local temporary storage—DFHTST TYPE=LOCAL
- Remote temporary storage—DFHTST TYPE=REMOTE
- Temporary storage security checking— DFHTST TYPE=SECURITY
- | Temporary storage data sharing—DFHTST TYPE=SHARED
- End of temporary storage table—DFHTST TYPE=FINAL (see page 371)

Control section—DFHTST TYPE=INITIAL

The entry point and the beginning address for the temporary storage table being defined are established by the DFHTST TYPE=INITIAL macro.

Table 83. DFHTST TYPE=INITIAL

DFHTST	TYPE=(INITIAL[,MIGRATE]) [,SUFFIX=xx] [,TSAGE={0|number}]

For general information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see "TYPE=INITIAL (control section)" on page 370.

| **MIGRATE**
| Specify MIGRATE when assembling your TST for migration to the CSD file.

TSAGE={0|number}
Defines the aging limit of temporary storage data used by the temporary storage domain during emergency restart of CICS. Data that has not been referenced for the specified interval is not recovered. The value is specified in days with a maximum value of 512. A value of zero indicates that no data is to be purged on this basis.

Recoverable temporary storage—DFHTST TYPE=RECOVERY

The DFHTST TYPE=RECOVERY macro specifies the generic names used for temporary storage queues for which recovery is applicable.

Table 84. DFHTST TYPE=RECOVERY

DFHTST	TYPE=RECOVERY ,DATAID=(*character-string* *[,character-string,...]*)|()

TYPE=RECOVERY
Code this to identify the temporary storage queue names that are recoverable. If a temporary storage queue name is such that it is defined by both a remote

and a recovery DATAID, it is considered to be remote. Recoverability can only be specified in the CICS region in which the queue is local.

Note: TYPE=ENTRY is retained for compatibility with previous releases, and means exactly the same as TYPE=RECOVERY.

DATAID=(*character-string[,character-string,...]***)|()**
Code this with one or more alphanumeric TS queue names that you want to be recoverable, where each name can be up to 8-characters in length. (See page 484 for information about generic names and matching criteria.)

character-string
Each character string can represent a generic queue name, or a unique TS queue name. Generic names are specified using 1 to 7 leading characters of TS queue names. DATAIDs that use all 8 characters define unique queues names.

Multiple names must be enclosed in parentheses, and separated by a comma. You can omit the parentheses if you specify only one name. Some CICS-generated TS queue names that you should consider for recovery are:

- **DF** refers to temporary storage queues used by CICS interval control for START commands with data, but which do not specify a REQID.
- ****** refers to temporary storage queues used by the BMS ROUTE command, and to those commands that use the PAGING operand.
- **$$** refers to temporary storage queues used by the BMS CMSG transaction when the PROTECT=YES option is specified on a START TRANSID command.

() This special (null) operand, without any value between the parentheses, is interpreted by CICS as specifying any queue that is not more explicitly specified by other DATAIDs.

Notes:

1. If a TST is generated with no TYPE=RECOVERY entries, no recovery processing is performed. If an EXEC CICS START command is issued with any of the FROM, RTRANSID, RTERMID, or QUEUE parameters specified, and a REQID is not specified, CICS generates request identifications starting with the prefix “DF”. If recovery is required for these requests, the TST should be generated with the corresponding generic name.
2. All temporary storage queues used by restartable transactions (those defined with RESTART(YES) in the transaction resource definition) should be made recoverable (including those with the default DF prefix).
3. Only data on auxiliary storage can be made recoverable. Data written to main storage is not recoverable, regardless of any recovery options that you may specify.

Local temporary storage—DFHTST TYPE=LOCAL

The DFHTST TYPE=LOCAL macro defines temporary storage queue names that reside in the local CICS region in which the TST is installed. This macro enables you to define local queues without knowing the SYSIDNT (see the SYSIDNT option on the DFHTST TYPE=REMOTE macro for more information).

Used in conjunction with the all-generic DATAID specified on the TYPE=REMOTE macro for remote and shared queues, this macro can help you to simplify greatly the task of defining local and remote queues.

Table 85. DFHTST TYPE=LOCAL

DFHTST	TYPE=LOCAL ,DATAID=(*character-string* *[,character-string,...]*)|()

TYPE=LOCAL
Indicates that this TST entry defines a set of local temporary storage queues.

DATAID=(*character-string[,character-string,...]***)|()**
Code this with one or more alphanumeric TS queue names, where each name can be up to 8-characters in length.

character-string
Each character string can represent a generic queue name, or a unique TS queue name. Typically, generic names are specified using 1 to 7 leading characters of TS queue names. DATAIDs that use all 8 characters define unique queue names.

Multiple names must be enclosed in parentheses, and separated by a comma. You can omit the parentheses if you specify only one name.

() This special (null) operand, without any value between the parentheses, is interpreted by CICS as specifying any queue that is not more explicitly specified by other DATAIDs. You can use this as a catch-all in the following way:

- If certain queues, which reside either in another region or in a shared TS pool, are specified on a TYPE=REMOTE macro with suitable generic DATAIDs, you can define all other queues as local by specifying DATAID=() on the TYPE=LOCAL macro.

This null option on the TYPE=LOCAL macro is mutually exclusive with DATAID=() on the TYPE=REMOTE macro, and the TST macro returns an assembly error if it is specified on both local and remote entries. Thus, if you specify DATAID=() on local TS queue entries, the TYPE=LOCAL macros must follow all TYPE=REMOTE macros.

Remote temporary storage—DFHTST TYPE=REMOTE

The DFHTST TYPE=REMOTE macro defines temporary storage queue names that reside in remote CICS regions when CICS intercommunication facilities are being used.

| Use this macro also to define queues residing in a shared queue pool, which is
| treated like a remote region except that the name of the remote system matches the
| system name on a DFHTST TYPE=SHARED macro.

Table 86. DFHTST TYPE=REMOTE

DFHTST	TYPE=REMOTE ,DATAID=(*character-string* *[,character-string,...]*)|() ,SYSIDNT=name [,RMTNAME=character-string]

TYPE=REMOTE
Indicates that this TST entry defines a set of remote temporary storage queues, which can reside either in a remote CICS region or in a shared TS pool in a coupling facility.

DATAID=(*character-string[,character-string,...]***)|()**
Code this with one or more alphanumeric TS queue names, where each name can be up to 8-characters in length. Use 1 to 7 leading characters from the leading characters of queue names to form generic names of those queues for which requests are to be routed to a remote region or to a TS server. (See page 484 for information about generic names and matching criteria.)

Note: You cannot use the list form of the DATAID operand when RMTNAME is specified. If you specify the RMTNAME parameter, the syntax for DATAID is DATAID=*character-string*.

character-string
Each character string can represent a generic queue name, or a unique TS queue name. Typically, generic names are specified using 1 to 7 leading characters of TS queue names. The generic names are those used by application programs in the region in which this TST is installed.

Multiple names must be enclosed in parentheses, and separated by a comma. You can omit the parentheses if you specify only one name.

() This special operand, without any value between the parentheses, is interpreted by CICS as specifying any queue that is not more explicitly specified by other DATAIDs. You can use this as a catch-all in the following way:

- If the queues with names beginning with letters L, M, and N are local, and these are specified on a TYPE=LOCAL macro with suitable generic DATAIDs, you can define all other queues as remote by specifying DATAID=() on the TYPE=REMOTE macro, as follows:

```
         DFHTST TYPE=LOCAL,       *
               DATAID=(L,M,N)
*
         DFHTST TYPE=REMOTE,      *
               DATAID=()
```

The DATAID=() option on the TYPE=REMOTE macro is mutually exclusive with DATAID=() on the TYPE=LOCAL macro, and the TST macro returns an assembly error if it is specified on both local and remote entries.

DATAID=() must be the last entry in a set of local and remote entries. Thus, if you use DATAID=() on remote TS queue entries, the TYPE=REMOTE macros must follow any TYPE=LOCAL macros.

SYSIDNT=name
Identifies the region or server in which the remote or shared temporary storage queues reside. For a remote queue owned by another CICS region, the 4-character alphanumeric name specified must be the same as a REMOTENAME specified in the CONNECTION definition, or the SYSIDNT name specified on a DFHTST TYPE=SHARED entry.

You can use this parameter to specify the name of the local region in which the TST is installed. When the SYSIDNT operand matches the SYSIDNT specified on the system initialization parameter, the TS queues that match the DATAIDs are treated as local queues.

RMTNAME=character-string
Code this with the 1- to 8-character prefix that is to be used by CICS to replace that specified in the DATAID operand when a reference to the temporary storage queue is transmitted to a remote system or region. This operand defaults to the character string specified in the DATAID operand. The length of the character string specified in this operand must be the same as that in the DATAID operand. This mechanism allows access to a temporary storage queue in the remote system with the same name as one in the local system.

Temporary storage security checking—DFHTST TYPE=SECURITY

The DFHTST TYPE=SECURITY macro indicates that security checking is required for the temporary storage queues specified in the TST.

Table 87. DFHTST TYPE=SECURITY

DFHTST	TYPE=SECURITY ,DATAID=(*character-string* *[,character-string,...]*)|()

TYPE=SECURITY
Indicates that this TST entry defines a set of temporary storage queues that require security checking. You are recommended to code this operand for each temporary storage queue that can be browsed by the CEBR transaction.

DATAID=(*character-string[,character-string,...]***)|()**
Code this with one or more alphanumeric TS queue names, where each name can be up to 8-characters in length. Use 1 to 7 leading characters from the leading characters of queue names to form generic names of those queues that are subject to security checking. (See page 484 for information about generic names and matching criteria.)

Notes:

1. When this macro is used, a suitable profile (see the *CICS RACF Security Guide* for information about profiles) must be defined to the external security manager to control access to the TSQ. Otherwise, the macro will not have the intended effect.
2. The full TSQ name, rather than the DATAID, is passed to the security manager. (CICS/ESA 3.3 and earlier releases passed the DATAID.)

character-string
Each character string can represent a generic queue name, or a unique TS queue name. Typically, generic names are specified using 1 to 7 leading characters of TS queue names. The generic names are those used by application programs in the region in which this TST is installed.

Multiple names must be enclosed in parentheses, and separated by a comma. You can omit the parentheses if you specify only one name.

() This null operand, without any value between the parentheses, is interpreted by CICS as specifying any queue that is not more explicitly specified by other DATAIDs.

Temporary storage data sharing—DFHTST TYPE=SHARED

The DFHTST TYPE=SHARED macro specifies the remote system name by which CICS identifies a temporary storage pool in the coupling facility.

Table 88. DFHTST TYPE=SHARED

DFHTST	TYPE=SHARED

Table 88. DFHTST TYPE=SHARED (continued)

```
,SYSIDNT=system_name
,POOL=pool_name
```

TYPE=SHARED
indicates that this TST entry defines the name of a pool of TS data sharing queues. Note that TYPE=SHARED entries do not have a DATAID parameter, and are located by their SYSIDNT in one of the following ways:

- # From a matching SYSIDNT specified on a TYPE=REMOTE entry
- # From a matching SYSID specified explicitly on an EXEC CICS temporary storage command, or supplied by a global user exit program.

Note: If you specify the SYSID explicitly on API commands to refer to shared TS queues that reside in a data sharing pool, you cannot migrate these TYPE=SHARED entries to TSMODELs in the CSD and must continue to use a TST.

SYSIDNT=*system_name*
specifies the 1- to 4-character system name that corresponds to a TS pool name.

CICS uses this SYSIDNT to map remote queues (defined by a TYPE=REMOTE entry, or an explicit SYSID on an API command) to a TS server, as follows:

- If an API temporary storage command specifies a SYSID option (explicitly on the command, or inserted by an XTSEREQ global user exit program), CICS maps the SYSID to a matching SYSIDNT on a TYPE=SHARED entry:
  - If a matching SYSIDNT is found, CICS uses the corresponding POOL name to identify the TS server that manages the shared TS queue.
  - If the SYSID does not match any TYPE=SHARED entry, the request is function shipped to the remote queue-owning region (QOR) named by SYSID.
- If an API temporary storage command references a remote queue identified by a TYPE=REMOTE entry, CICS checks for a matching SYSIDNT in the TYPE=SHARED entries:
  - If a TYPE=SHARED entry with a matching SYSIDNT is found, CICS uses the corresponding POOL name to identify the TS server that manages the shared TS queue.
  - If a TYPE=SHARED entry is not found, the queue is a remote queue and the request is function shipped to the QOR.

You can create multiple TYPE=SHARED entries, with different SYSIDNT names, that refer to the same POOL name. In this case, references to the same queue name refer to the same queue name regardless of which SYSID is used (on the API).

POOL=*pool_name*
specifies the 1- to 8-character name of the pool of TS queues that is to be used for TS requests that specify, implicitly or explicitly, the corresponding system name. The pool-name must match the name specified on the POOL parameter of the TS server that manages the TS pool.

DFHTST example

Figure 62 illustrates an example of the coding necessary to create a CICS TST.

```
     DFHTST TYPE=INITIAL,              LIST OF GENERIC NAMES OF QUEUES *
            SUFFIX=01                  THAT ARE RECOVERABLE, REMOTE,
*                                      SHARED, LOCAL, OR REQUIRE
*                                      SECURITY CHECKING.
*
*  The following macro specifies that all LOCAL queues with
*  names beginning with the letter 'R' are RECOVERABLE:
*
     DFHTST TYPE=RECOVERY,                                             *
            DATAID=R
*
*  The following macro specifies that queues with names
*  beginning with C,D,E, and X are local queues:
*
     DFHTST TYPE=LOCAL,                                                *
            DATAID=(C,D,E,X)
*
*  The following macro specifies that queues with names
*  beginning with AB,L,M,N are remote queues on system RSYS:
*
     DFHTST TYPE=REMOTE,                                               *
            DATAID=(AB,L,M,N),                                         *
            SYSIDNT=RSYS,              Queue names on remote system    *
            RMTNAME=LC                 begin with letters LC
*
*  The next macro specifies that all queues not local as defined
*  above, or remote in system RSYS as defined above, are remote
*  queues that reside in a shared TS pool TYPE=SHARE macro.
*
     DFHTST TYPE=REMOTE,                                               *
            DATAID=(),                                                 *
            SYSIDNT=SHR1
*
*  The next macro specifies that remote queues with SYSIDNT=SHR1
*  are mapped to shared TS pool named TSQSHR1.
*
     DFHTST TYPE=SHARED,                                               *
            SYSIDNT=SHR1,                                              *
            POOL=TSQSHR1
*
*  The following macro specifies that queues with names
*  beginning with SAQ require security checking.
*    Note that the full TS queue name is passed to the ESM.
*
     DFHTST TYPE=SECURITY,                                             *
            DATAID=SAQ
*
     DFHTST TYPE=FINAL
     END
```

Figure 62. Temporary storage table—example

XLT—transaction list table

The transaction list table (XLT), generated by the DFHXLT macro instruction, is a list of logically related transaction identifications. The XLT can be used to define:

- A list of transaction identifications that can be initiated from terminals during the first quiesce stage of system termination. If there are no PLT programs to execute, the first quiesce time can be short, thus giving little time to enter any XLT program before going into the second quiesce stage. You specify the suffix of the table to be used by means of the XLT system initialization parameter. The master terminal operator can change the suffix at system termination, using the XLT option of the CEMT PERFORM SHUTDOWN command. In addition to the transactions listed in the XLT, the CEMT and CESF CICS-supplied transactions can be initiated from terminals during the first quiesce stage, as can any transactions defined with SHUTDOWN(ENABLED).

Note: You can also define the XLT as a PROGRAM if you would rather use RDO than macros. See "PROGRAM" on page 242 for information on defining programs. Defining it as a program also means that it can be autoinstalled; see "Chapter 11. Autoinstall for other resources" on page 145 for information on autoinstall for programs.

- A group of transaction identifications to be disabled or enabled through the master terminal. The master terminal operator specifies the suffix of the table to be used, using the CLASS option of the CEMT SET TRANSACTION command. For details of the CEMT commands, see the *CICS Supplied Transactions* manual.

Figure 63 on page 493 illustrates the coding to create a XLT.

Elements of DFHXLT

The following macros are available to define the XLT entries:

- Control section—DFHXLT TYPE=INITIAL
- Entries in transaction list table—DFHXLT TYPE=ENTRY
- End of transaction list table—DFHXLT TYPE=FINAL (see page 371)

Control section—DFHXLT TYPE=INITIAL

The DFHXLT TYPE=INITIAL macro establishes the entry point and start address of the XLT being defined:

Table 89. DFHXLT TYPE=INITIAL

```
DFHXLT          TYPE=INITIAL
                [,SUFFIX=xx]
```

For general information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see "TYPE=INITIAL (control section)" on page 370.

Entries in transaction list table—DFHXLT TYPE=ENTRY

Table 90. DFHXLT TYPE=ENTRY

```
DFHXLT          TYPE=ENTRY
                [,TASKREQ=(kkkk[,kkkk],...) |
                ,TRANSID=(xxxx[,xxxx],...)]
```

TYPE=ENTRY
Code this if one or more entries are to be generated in the XLT.

TASKREQ=(kkkk[,kkkk],...)
kkkk can be one of the following:

- PA1 through PA3, and PF1 through PF24 indicates one of the special 3270 keys that can be used to initiate a task.
- LPA (light pen attention) indicates that a transaction is to be initiated when a light pen detectable field is selected.
- OPID (operator identification card reader) indicates that a transaction is initiated when the appropriate operator's identity badge has been read in.
- MSRE indicates that transactions are initiated when the 10/63 character magnetic slot reader is used.

Define each TASKREQ on the CSD file, and install it in the running system. (For further information, see the description of TASKREQ on page 324.)

TRANSID=(xxxx[,xxxx],...)
Represents a 1- to 4-character transaction code. Define each TRANSID on the CSD file, and install it in the running system. (For further information, see the description of TRANSACTION on page 326.)

If the TRANSID contains a special character (for example, a comma), the TYPE=ENTRY instruction must contain only one TRANSID with quotation marks as delimiters.

Note: TASKREQ and TRANSID are mutually exclusive parameters.

DFHXLT example

```
   DFHXLT TYPE=INITIAL,                   LIST OF TRANSACTIONS     *
          SUFFIX=IN                       THAT ARE ACCEPTED
*                                         DURING THE FIRST QUIESCE
*                                         PHASE OF SYSTEM
*                                         TERMINATION.
   DFHXLT TYPE=ENTRY,TASKREQ=PF5          (TASKREQ MUST ALSO BE
*                                         DEFINED IN THE CSD AND
*                                         INSTALLED IN THE RUNNING
*                                         CICS SYSTEM. AN ENTRY FOR
*                                         THE XLT MUST BE MADE IN
   DFHXLT TYPE=ENTRY,TRANSID=(USR1,USR2)  THE CSD.)
   DFHXLT TYPE=ENTRY,TRANSID='AA,1'
   DFHXLT TYPE=ENTRY,TRANSID='AA,2'
   DFHXLT TYPE=FINAL
   END

   DFHXLT TYPE=INITIAL,                   LIST OF LOGICALLY RELATED*
          SUFFIX=G1                       TRANSIDS TO BE ENABLED OR
*                                         DISABLED BY MASTER
*                                         TERMINAL.
   DFHXLT TYPE=ENTRY,TRANSID=(TSSA,TSRA)  (TRANSIDS MUST ALSO BE
   DFHXLT TYPE=ENTRY,TRANSID=(TDSA,TDRA)  DEFINED IN THE CSD AND
   DFHXLT TYPE=ENTRY,TRANSID=ICSA         INSTALLED IN THE RUNNING
   DFHXLT TYPE=FINAL                      CICS SYSTEM.)
   END
```

Figure 63. Transaction list table—example

Part 7. Appendixes

Appendix A. Obsolete attributes retained for compatibility

The attributes described here are not valid in CICS Transaction Server for OS/390, but are supported to provide CSD compatibility for earlier releases of CICS where they are still valid. See "Compatibility mode (CSD file sharing)" on page 12 for more information on compatibility mode.

Table 91 on page 501 shows which resource or resources each attribute is associated with, and which release or releases it was supported in.

BINDPASSWORD(password) (APPC only)
A password of up to 16 hexadecimal digits (0-9, A-F). A password of fewer than 16 digits is padded on the right with hexadecimal zeros.

CICS masks the password you supply to avoid unauthorized access. You should therefore find a safe way of recording the password.

If you supply a password, an identical password must be supplied in the remote system to ensure bind-time security, allowing a connection to be established.

EXTSEC({NO|YES})
specifies whether an external security manager (for example, RACF) is to be used for transaction security or resource security checking.

NO Only the security facilities provided by CICS are used by this transaction.

YES An external security manager may be used by this transaction.

INDOUBT({BACKOUT|COMMIT|WAIT})
specifies the action required if the transaction is using intercommunication, and abends at a critical time during syncpoint or abend processing. For guidance on using the INDOUBT option, see the *CICS Intercommunication Guide*.

BACKOUT
The effects of the transaction are backed-out. This must be specified for recoverable files.

COMMIT
The effects of the transaction are committed. Use INDOUBT(COMMIT) if you do not want dynamic transaction backout.

WAIT Changes to recoverable temporary storage are locked until the session is recovered. The resources are then committed or backed out in step with the remote system.

INSERVICE({YES|NO})
specifies whether the session(s) can be used for communication. This attribute applies only to LUTYPE 6.1 ISC sessions. It is invalid for LUTYPE 6.2, and is ignored for MRO sessions. For MRO the status (in service or out of service) is determined by the status of the corresponding MRO CONNECTION.

YES Transactions may be initiated and messages may automatically be sent across the session(s).

NO The session(s) can neither receive messages nor transmit input.

LOGMODECOM({NO|YES})
LOGMODECOM indicates LOGMODE compatibility. It shows whether CICS is to make LOGMODE work the way it does in releases earlier than CICS/ESA 4.1. This parameter is not available in releases later than CICS/ESA 4.1.

LOGMODECOM(NO)
is the default and causes LOGMODE(0|name) to work as described under LOGMODE.

LOGMODECOM(YES)
causes LOGMODE(0|name) to work as it did in releases before CICS/ESA 4.1 for non *XRF-capable terminals. LOGMODECOM(YES) is ignored for XRF-capable terminals. Use this parameter only in exceptional circumstances - see the *CICS/ESA 4.1 Migration Guide* and the *CICS/ESA 4.1 Resource Definition Guide* for a fuller explanation.

OPERID(*code*)
specifies the 3-character operator identifier associated with the sessions. Use OPERID if you are not specifying SECURITYNAME on the CONNECTION definition. Specifying OPERID is the **only** way of having an operator identifier if you have preset security (by specifying OPERRSL and OPERSECURITY).

OPERPRIORITY({0|number})
specifies the operator priority code to be used to determine the task processing priority for each transaction attached to the sessions. The code may be any value from 0 through 255. Use OPERPRIORITY if you are not specifying SECURITYNAME on the CONNECTION definition. Specifying OPERPRIORITY is the **only** way of having an operator priority code if you have preset security (by specifying OPERRSL and OPERSECURITY).

OPERRSL({0|number[,...]})
specifies the resource security key for these sessions.

number[,...]
Code the preset resource security keys for these sessions. The OPERRSL keys are checked to see that they include the resource RSL value, by transactions that request RSL checking (RSLC(YES)). They are referenced for function shipping and distributed transaction processing requests. The OPERRSL keys comprise one or more decimal values from 1 through 24. You can specify more than one value as an inclusive range, using a dash (for example: `5-12`), or as a series of numbers separated by commas, for example: `5,6,7,8,9,10,11,12.` These two examples are equivalent. You can use dashes and commas in the same specification if you need to.

Specify OPERRSL keys if you are not specifying SECURITYNAME on the CONNECTION definition. However, if you specify OPERRSL keys for the sessions, you cannot have a sign-on, using SECURITYNAME, when the link is established. (For more information, see the *CICS Intercommunication Guide*). Note that the OPERRSL keys give access only to resources with the RSL values actually specified in the OPERRSL keys, not to resources with lower RSL values.

0 The sessions have no OPERRSL keys specified and do not have access to any resources through transactions with RSLC(YES), except resources with RSL(PUBLIC).

OPERSECURITY({1|number[,...]})
specifies the preset transaction security keys for the device. The transaction

security keys are checked to see that they include the security value (TRANSEC) for a transaction about to be attached. They are referenced for function shipping and distributed transaction processing requests.

The security keys comprise one or more decimal values from 1 through 64. You can specify these values in the same way as for OPERRSL, above. In addition to the values you specify, a value of 1 is also assumed. The default value of 1 gives access to all unsecured transactions, because the default TRANSEC value is 1. For example: `5-10,12` is translated into: `1,5,6,7,8,9,10,12.`

Use OPERSECURITY if you are not specifying SECURITYNAME on the CONNECTION definition. However, if you specify OPERSECURITY keys for the sessions, you cannot have a sign-on, using SECURITYNAME, when the link is established. (For more information, see the *CICS Intercommunication Guide*.)

PRIMEDSIZE({0|value})
specifies the primed storage allocation size in bytes.

0 CICS takes care of the storage for the control blocks.

Note: Leave PRIMEDSIZE as 0 if this TRANSACTION definition has been migrated with ANTICPG=YES.

value This value must not exceed 65520 bytes and, if specified at all, must include an allowance of 2800 bytes for CICS control blocks, and an allowance for the size of the TWA.

Storage acquired by a GETMAIN within the primed storage area is never freed (that is, the corresponding FREEMAIN is ignored).

Note that storage accounting areas within the primed storage allocation are doubleword-aligned, instead of the normal double-doubleword-aligned.

PROTECT({NO|YES}) (SNA LUs only)
specifies whether output messages can be recovered (see the MSGINTEG option on page 237), and whether message logging is to take place.

NO Neither message integrity nor message logging is to take place.

YES Provides recovery for output messages. CICS also records the contents of deferred write requests that are pending at a syncpoint, and records the receipt of the definite response (associated with the deferred write) on the system log for message recovery and resynchronization purposes. Journaling support is required during generation of the CICS system.

If you specify PROTECT(YES):

- Specify MSGINTEG(YES). This ensures that the integrity response is received.
- Ensure that definitions for the transaction CSLG and program DFHZRLG are available.

RECOVNOTIFY({NONE|MESSAGE|TRANSACTION})
specifies whether, and how, the terminal user is notified that an XRF takeover has occurred, in case the user needs to take some action such as signing on again.

NONE
The user is not notified.

MESSAGE
: The user receives a message on the screen that the system has recovered. There are two BMS maps, DFHXRC1 and DFHXRC2, in map set DFHXMSG for the message. MESSAGE, rather than TRANSACTION, minimizes the takeover time.

  The terminal must be defined with the ATI(YES) option, and must be capable of displaying a BMS map.

TRANSACTION
: CICS initiates a transaction at the terminal. The name of the transaction is specified by the RMTRAN system initialization parameter. (The default transaction for this is the one specified in the GMTRAN system initialization parameter: the good-morning transaction.) TRANSACTION is more versatile than MESSAGE.

  The terminal must be defined with ATI(YES).

RESSECNUM({0 | value | PUBLIC})
specifies the resource security value to be associated with this file. This attribute is used when an EXEC command is executed within a transaction that has been defined with RESSEC(YES), and the command is attempting to reference this file.

0 A transaction defined with RESSEC(YES) is not allowed access to the file.

value The resource security value, in the range 1 through 24. When a transaction defined with RESSEC(YES) attempts to reference this file, *value* is checked against the keys derived from RESSECKEYS either in the sign-on table, or from the TERMINAL definition. If one of these keys matches *value*, the transaction is allowed access to the file.

PUBLIC
Any transaction is allowed access to the file, regardless of whether security checking is specified or not.

| **RPG**
| RPG was a permitted value for the LANGUAGE option of a PROGRAM
| resource until CICS Transaction Server for OS/390.

RSL(0 | *value* | PUBLIC)
specifies the resource security value to be associated with this resource. This operand is used when an EXEC command is executed within a transaction that has been defined with RSLC(YES), and the command is attempting to reference the partition set.

0 A transaction defined with RSLC(YES) is not allowed access to the partition set.

value The resource security value, in the range 1 through 24. When a transaction defined with RSLC(YES) attempts to reference this partition set, *value* is checked against the keys derived either from the RSLKEY in the sign-on table, or from the OPERRSL on the TERMINAL definition. If one of these keys matches *value*, the transaction is allowed access to the partition set.

PUBLIC
Any transaction is allowed access to the partition set, regardless of whether no security checking or RSL checking is specified. However, if

an external security manager is in force, it checks access authorities no matter what RSL value (including PUBLIC) has been defined for the resource.

TCLASS({NO | value})
specifies the class associated with the task.

NO No class is assigned to the task.

value The decimal value (from 1 to 10) of the class associated with a task.

Note: Do not specify a TCLASS for a CICS-supplied transaction, because it may not be able to start if the class threshold is reached.

TRANSACTION(*name*)
allows only the specified transaction to be initiated from this device.

The name can be up to four characters in length. The acceptable characters are: A-Z a-z 0-9 $ @ # . / - _ % & ¢ ? ! : | ″ = ¬ , ; < >.

If you code this operand for a 3270 display, the only CICS functions the operator is able to invoke—other than this transaction—are paging commands and print requests.

TRANSEC({1 | *value*})
specifies the transaction security value, in the range 1 through 64. When a user attempts to initiate the transaction, or when it is automatically initiated (through transient data or interval control), *value* is matched against the user's security keys defined in the DFHSNT SCTYKEY operand or, if the user is not signed on, the security keys defined in OPERSECURITY on the TERMINAL definition. If *value* is present in the security keys, the transaction is initiated.

Because all users and terminals have a security key of 1, any transaction with the default TRANSEC value of 1 is an unsecured transaction, and as such, it can be initiated by any user on the CICS system, whether they are signed on or not.

Obsolete attributes—when were they supported?

Table 91 shows which resource or resources each attribute is associated with, and which release or releases it was supported in.

Table 91. Obsolete attributes and their valid releases of CICS

Attribute	Resource type	Supported in
BINDPASSWORD	CONNECTION TERMINAL	CICS/ESA 4.1 CICS/ESA 3.3 CICS/ESA 3.2.1 CICS/ESA 3.1.1 CICS/MVS 2.1
EXTSEC	TRANSACTION	CICS/MVS 2.1
INDOUBT	TRANSACTION	CICS/ESA 4.1 CICS/ESA 3.3 CICS/ESA 3.1.1 CICS/MVS 2.1
INSERVICE	SESSION	CICS/ESA 3.2.1 CICS/ESA 3.1.1 CICS/MVS 2.1.2
LOGMODECOM	TYPETERM	CICS/ESA 4.1
OPERID	SESSION TERMINAL	CICS/ESA 3.1.1 CICS/MVS 2.1.2

Table 91. Obsolete attributes and their valid releases of CICS (continued)

Attribute	Resource type	Supported in
OPERPRIORITY	SESSION TERMINAL	CICS/ESA 3.1.1 CICS/MVS 2.1
OPERRSL	SESSION TERMINAL	CICS/MVS 2.1
OPERSECURITY	SESSION TERMINAL	CICS/MVS 2.1
PROTECT	PROFILE	CICS/ESA 4.1 CICS/ESA 3.3 CICS/ESA 3.1.1 CICS/MVS 2.1
PRIMEDSIZE	TRANSACTION	CICS/MVS 2.1
RECOVNOTIFY	SESSION	CICS/ESA 3.3 CICS/ESA 3.2.1 CICS/ESA 3.1.1 CICS/MVS 2.1
RESSECNUM	FILE	CICS/ESA 3.3 CICS/ESA 3.2.1 CICS/ESA 3.1.1
RPG (value of LANGUAGE attribute)	PROGRAM	CICS/ESA 4.1
RSL	MAPSET PARTITIONSET PROGRAM TRANSACTION	CICS/MVS 2.1
TCLASS	TRANSACTION	CICS/ESA 3.3 CICS/ESA 3.2.1 CICS/ESA 3.1.1 CICS/MVS 2.1
TRANSACTION	SESSION	CICS/MVS 2.1
TRANSEC	TRANSACTION	CICS/ESA 3.1.1 CICS/MVS 2.1

Appendix B. CICS-supplied resource definitions, groups, and lists

This chapter lists the resource definitions, groups, and lists supplied by IBM.

You initialize the CSD file by using the DFHCSDUP INITIALIZE command. (This command has no operand.) Following initialization, the CSD file contains two categories of resource definition groups:

1. Groups named in DFHLIST, essential for using RDO and other CICS-supplied transactions
2. Groups of definitions for the sample application programs, which are described in the *CICS 4.1 Sample Applications Guide*.

This chapter also lists the CICS transactions, the TYPETERM definitions, model TERMINAL definitions, and PROFILE definitions that are supplied by IBM. These definitions are in four groups:

- DFHTYPE—TYPETERM definitions
- DFHTERM—model TERMINAL definitions for automatic installation
- DFHISC—PROFILE definitions for intersystem communication sessions
- DFHSTAND—PROFILE definitions

DFHLIST definitions

Table 92. DFHLIST resource definitions

Group Name	Description	Programs (map sets, profiles, terminals, tranclasses journalmodels or typeterms as indicated)	Transactions
DFHAUGRP	Affinities Utility application programs	CAUCAFBE CAUCAFB1 CAUCAFB2 CAUCAFFE CAUCAFF1 CAUCAFF2 CAUCAFF3 CAUCAFF4 CAUCAFF5 CAUCAFF6 CAUCAFF7 CAUCAFP CAUMSGCS CAUTABM CAUXITI1 CAUXITO1 CAUXITM1 CAUXDUMM **Map sets**: CAUMAP1 CAUMAP2 CAUMAP3 CAUMAP4	CAFB CAFF
DFHBMS	Basic mapping support	DFHTPQ DFHTPR DFHTPS	CSPG CSPQ CSPS
| **DFHCBTS** [5]	Local Request Queue file for BTS	**File**: DFHLRQ	
| **DFHCFC**	Programs needed for CICS C++ foundation classes	ICCFCDLL	-
| **DFHCLNT**	CICS Client CTIN	DFHZCT1 DFHZCN1 **Tranclass**: DFHCOMCL	CTIN CCIN
DFHCONS	Write to CPU console	DFHCWTO	CWTO

Table 92. DFHLIST resource definitions (continued)

Group Name	Description	Programs (map sets, profiles, terminals, tranclasses journalmodels or typeterms as indicated)	Transactions
DFHDBCTL	DBCTL transactions	DFHDBAT DFHDBCON DFHDBCT DFHDBDI DFHDBDSC DFHDBIQ DFHDBME DFHDBUEX **Map sets:**DFHDBIE DFHDBNE	CDBC CDBD CDBI CDBN CDBO CDBT
DFHDB2	DB2 support	DFHD2CM0 DFHD2CM1 DFHD2CM2 DFHD2CM3 DFHD2EDF DFHD2EX1 DFHD2EX2 DFHD2INI DSNCUEXT DSNTIAC DSNTIA1	CDBF CDBQ CEX2 DSNC
DFHDCTG [5]	TD queues for basic CICS facilities	-	CADL CAFF CAIL CCPI CCSE CCSO CDBC CDUL CESE CESO CMIG CPLD CPLI CRDI CRPO CSCC CSCS CSDL CSFL CSKL CSML CSMT CSNE CSPL CSQL CSRL CSSL CSTL CSZL CSZX CWBO
DFHDOC [5]	Document handler template reader	DFHDHEI	-
DFHEDF	Execution diagnostic facility	DFHDBMS DFHDBTI DFHEDFBR DFHEDFD DFHEDFP DFHEDFR DFHEDFX DFHEIGDS DFHEITAB **Map set**: DFHEDFM	CEBR CEDF
DFHEDP	EXEC DLI user exit	DFHEDP	-
DFHFE	FE terminal test facility	DFHFEP DFHTRAP	CSFE
DFHFEPI	Front End Programming Interface	DFHEITSZ DFHSZRMP	CSZI
DFHHARDC	3270 Printer - VTAM	DFHP3270	CSPP
DFHIIOP	Programs for DFHIIOP/CORBA group	DFHIIOP DFHIIOPA DFHXOPUS DFJIIOP DFJCORB DFJIISRV **Request Model**: DFHGFACT	CIOD CIOF CIOR
DFHINDT	In-doubt test tool	DFHINDAP DFHINDT DFHINTRU **Tranclass**: DFHTCIND	CIND
DFHINQUI	Command definition	DFHEITBS	-
DFHINTER	Command interpreter	DFHECID DFHECIP DFHECSP	CECI CECS

Table 92. DFHLIST resource definitions (continued)

Group Name	Description	Programs (map sets, profiles, terminals, tranclasses journalmodels or typeterms as indicated)	Transactions
DFHISC	Intersystem communication	DFHCCNV DFHCHS DFHCLS3 DFHCLS4 DFHCLS5 DFHCNV DFHCRNP DFHCRQ DFHCRR DFHCRS DFHCRSP DFHCRT DFHDFST DFHDSRP DFHDYP DFHLUP DFHMIRS DFHMXP DFHRTC DFHRTE DFHSHRRP DFHSHRSP DFHUCNV DFHZLS1 **Profiles**: DFHCICSF DFHCICSR DFHCICSS	CDFS CEHP CEHS CLQ2 CLR2 CLS1 CLS2 CLS3 CLS4 CMPX CPMI CQPI CQPO CRSQ CRSR CRTE CRTX CSHR CSMI CSM1 CSM2 CSM3 CSM5 CSNC CSSF CVMI CXRT
DFHJAVA	Programs needed for CICS java support via VisualAge for Java, Enterprise ToolKit for OS/390	DFHDLLOD DFJCICS DFJCICSB DFJCZWRP DFJCZDTC DFHJVCVT	-
DFHLGMOD	CICS log manager	**Journalmodels**DFHLOG DFHSHUNT DFHLGLOG	-
DFHLGQC	CICS log manager quiesce	DFHLGQC	CSQC
DFHMISC	Miscellaneous programs	DFHNEP DFHPEP DFHRTY	-
DFHMSWIT	Message switching program	DFHMSP	CMSG
DFHOPCLS	Dynamic open/close program	DFHFCU	CSFU
DFHOPER	Operator programs	DFHCETRA DFHCETRB DFHCETRC DFHCETRD DFHEITMT DFHEITOT DFHEITST DFHEMTA DFHEMTD DFHEMTP DFHEOTP DFHESTP **Map sets**: DFHCTRH DFHCTRM	CEMT CEOT CEST CETR
DFHPGAIP	Autoinstall for programs	DFHPGADX DFHPGAHX DFHPGALX DFHPGAOX DFHPGAPG **Map set**: DFHPGAMP **Partition set**: DFHPGAPT	-
Note: Group DFHPLI is withdrawn. For PL/I Version 2 or later, see the PL/I installation manual (for example, *OS PL/I Version 2 Installation and Customization under MVS Guide*) for a complete list of the required program entries.			
DFHRMI	Resource manager interface	DFHRMSY	CRSY
DFHRPC	Remote procedure call	DFHRPAL DFHRPAS DFHRPC00 DFHRPMS DFHRPRP DFHRPTRU **Map sets**: DFHRP0	CRPA CRPC CRPM
DFHRSEND	VTAM resend program	DFHZRSP	CSRS
DFHSDAP	SHUTDOWN ASSIST	DFHCESD	CESD
DFHSIGN	Sign-on/sign-off programs and table	DFHSFP DFHSNP DFHSNT **Map sets**: DFHSNLE DFHSNSE	CESF CESN

Table 92. DFHLIST resource definitions (continued)

Group Name	Description	Programs (map sets, profiles, terminals, tranclasses journalmodels or typeterms as indicated)	Transactions
DFHSPI	Resource definition online	DFHAMP DFHDMP DFHEDAD DFHEDAP DFHEITSP DFHPUP DFHTBS DFHTOR DFHZATA DFHZATD DFHZATDX DFHZATR DFHZATS DFHZCQ DFHZCTDX DFHZDTDX DFHZPTDX	CATA CATD CATR CATS CDTS CEDA CEDA CEDB CEDC CFTS CITS CMTS
DFHSTAND	Standard CICS application programs	DFHACP DFHCXCU DFHPSIP DFHQRY DFHSTP DFHTACP DFHTEP DFHTEPT DFHTFP DFHZXCU DFHZXRE DFHZXST **Profiles**: DFHCICSA DFHCICSE DFHCICSP DFHCICST DFHCICSV DFHPPF01 DFHPPF02 **Map set**: DFHXMSG	CQRY CSAC CSTE CXCU CXRE
DFHTCL	Compatibility TRANCLASS definitions	**Tranclasses:**DFHTCL00 DFHTCL01 DFHTCL03 DFHTCL04 DFHTCL05 DFHTCL06 DFHTCL07 DFHTCL08 DFHTCL09 DFHTCL10 DFHTCLQ2 DFHTCLSX	-
# **DFHTERM**	Model TERMINAL definitions	**Terminals**: LU2 LU3 APPC SCSP 3270 3284 L0E2 L0M2 L0M3 L0M4 L0M5 L2E2 L2M2 L2E3 L2M3 L2E4 L2M4 L2M5	-
# **DFHTYPE**	TYPETERM definitions	**Typeterms**: DFHCONS DFHLU2 DFHLU3 DFHLU62T DFHSCSP DFH3270 DFH3270P DFHLU0E2 DFHLU0M2 DFHLU0M3 DFHLU0M4 DFHLU0M5 DFHLU2E2 DFHLU2M2 DFHLU2E3 DFHLU2M3 DFHLU2E4 DFHLU2M4 DFHLU2M5	-
DFHVTAM	VTAM programs	DFHGMM DFHZNAC DFHZNEP	CSGM CSNE
DFHVTAMP	VTAM terminal control print key function	DFHCPY DFHEXI DFHPRK DFHRKB	CSCY CSPK CSRK

Table 92. DFHLIST resource definitions (continued)

Group Name	Description	Programs (map sets, profiles, terminals, tranclasses journalmodels or typeterms as indicated)	Transactions
DFHWEB	WEB interface definitions	DFH$WB1A DFHWBA DFHWBADX DFHWBAHX DFHWBALX DFHWBAOX DFHWBA1 DFHWBC00 DFHWBENV DFHWBEP DFHWBGB DFHWBIMG DFHWBIP DFHWBLT DFHWBPA DFHWBRA DFHWBST DFHWBTC DFHWBTL DFHWBTRU DFHWBTTA DFHWBUN DFHWBXN **TSModels**: DFHWEB	CWBA CWBC CWBG CWBM CWXN
Note: The DFHTYPE typeterm definitions match the VTAM-supplied LOGMODE definitions. If you are not using the supplied VTAM LOGMODES, you may need to modify the DFHTYPE TYPETERM definitions. For programming information on VTAM LOGMODE definitions, see the *CICS Customization Guide*.			

CICS-supplied groups not in DFHLIST

Table 93. Resource definitions not in DFHLIST

Group Name	Description	Programs (map set or file as indicated)	Transactions
DFHCMAC	CICS online messages and codes	DFHCMAC **Map set**: DFHCMCM **File**: DFHCMACD	CMAC (alias CHLP)
Note: DFHCMAC is not included in DFHLIST for reasons of compatibility with earlier releases. If you want to use the online messages and codes transaction, you must add DFHCMAC to your start-up group list.			
DFHMISC3	Miscellaneous group	DFHNET	-
Note: DFHMISC3 is not included in DFHLIST for reasons of compatibility with earlier releases. If you require it in your system, add it to your start-up group list.			

5. This group is not protected by a lock. The definitions it contains can be modified if required.

CICS-supplied compatibility groups

Table 94. Group definitions supplied for compatibility with earlier releases

Group Name	Description	Programs	Transactions
DFHCOMP1	Required for compatibility with release 2.1.2	DFHCCMF DFHCMON DFHDMP DFHEMA DFHEMB DFHEMC DFHEMD DFHEME DFHEMF DFHEMG DFHEMH DFHEMI DFHFCS DFHMTPA DFHMTPB DFHMTPC DFHMTPD DFHMTPE DFHMTPF DFHMTPG DFHRTY DFHSTKC DFHSTLK DFHSTPD DFHSTSP DFHSTTD DFHSTTR DFHTAJP DFHTRNSM DFHTRNSN DFHVAP DFHZCQ DFSHLPI	CAUT CCMF CDTS CECI CECS CEDA CEDB CEDC CEOT CEST CFTS CITS CMSG CMTS CQRY CRSR CSCY CSFE CSIR CSMT CSNC CSOT CSPK CSPP CSRK CSSC CSSF CSSN CSST CSTA CSTT CVST CWTO CXCU 8888 9999
DFHCOMP2	Required for compatibility with release 2.1.2	DFHECID DFHECIP DFHECSP DFHEDCP DFHEIPRT DFHEIPSE DFHEIPSH DFHEIQDN DFHEIQDS DFHEIQDU DFHEIQIR DFHEIQMS DFHEIQMT DFHEIQSA DFHEIQSC DFHEIQSJ DFHEIQSK DFHEIQSM DFHEIQSP DFHEIQSQ DFHEIQST DFHEIQSV DFHEIQSX DFHEIQTR DFHEIQVT DFHEOP DFHESE DFHESN DFHEITAB DFHEITBS DFHETRX DFHMATOC DFHOCP DFHUAKP IBMBCCLA IBMBCCRA IBMBEOCA IBMBETAA IBMBETBA IBMBETCA IBMBETIA IBMBETOA IBMBETPA IBMBETQA IBMBETTA IBMFEFCA IBMFESMA IBMFESNA IBMFKCSA IBMFKMRA IBMFKPTA IBMFKTBA IBMFKTCA IBMFKTRA IBMFPGDA IBMFPMRA IBMFSTVA	CSNC

Table 94. Group definitions supplied for compatibility with earlier releases (continued)

Group Name	Description	Programs	Transactions
DFHCOMP3	Required for compatibility with releases 2.1.2 and 3.2.1	DFHAKP DFHBRCP DFHCHS DFHDBAT DFHDBP2$ DFHDLG DFHDLS DFHEBRCT DFHFCU DFHJCC DFHJCI DFHJCIOE DFHJCKOJ DFHJCO DFHMIR DFHMIRVM DFHNET DFHRTY DFHSTP DFHTACP	CATS CEMT CEOT CEST CPMI CRSR CSFE CSJC CSMI CSM1 CSM2 CSM3 CSM5 CSNE CSPP CSPQ CSPS CSRS CVMI
DFHCOMP4	Required for compatibility with releases 2.1.2, 3.2.1, and 3.3	DFHCRP DFHCSSC DFHNEP DFHRTY DFHSNT	CEDF CLS1 CLS3 CSPG CSSC
DFHCOMP5	Required for compatibility with releases 2.1.2, 3.2.1, 3.3, and 4.1	DFHAKP DFHBRCP DFHDBP1$ DFHDBP2$ DFHDLG DFHDLS DFHEBRCT DFHJCBSP DFHJCC DFHJCEOV DFHJCI DFHJCIOE DFHJCKOJ DFHJCO DFHJCSDJ DFHZRLG	CBRC CSGX CSKP CSLG CSSX CSZI
DFHCOMP6	Required for compatibility with releases 2.1.2, 3.2.1, 3.3, 4.1 and CICS TS Release 1	DSN2COM0 DSN2COM1 DSN2COM2 DSN2EDF1 DSN2EXT1 DSN2EXT2 DSN2MSG0 DSN2STOP DSN2STRT DSNCUEXT	DSNC
DFHCOMP7	Required for compatibility with releases 2.1.2, 3.2.1, 3.3, 4.1, CICS TS Release 1 and CICS TS Release 2	DFHWBM DFHWBWB DFHWBTRU	BRG1 BRG2 BRG3 BRG4 BRG5 BRG6 BRG7 BRG8 CWB1 CWB2 CWB3 CWB4 CWB5 CWB6 CWB7 CWB8 CWBM

Sharing the CSD between different releases of CICS

If, after upgrading a CSD file to CICS Transaction Server for OS/390, you plan to share the CSD file with earlier releases of CICS, you must include the appropriate DFHCOMPx compatibility groups in your startup group list.

The *CICS Transaction Server for OS/390 Migration Guide* has information about sharing the CSD between different releases of CICS. It shows you which DFHCOMPx groups you need to include for the earlier releases. Do not attempt to share a CSD file with a CICS region running at a higher level than the CSD file.

The sample application program groups

These resource definitions are needed to run the sample application programs supplied with CICS. The groups are not named in DFHLIST.

Sample programs

Table 95. CICS sample applications - resource definitions

Group Name	Language	Description	Resources Defined
DFH$ACCT	COBOL	*CICS Application Programming Primer (VS COBOL II)* samples	**Map set**: ACCTSET **Programs**: ACCT00 ACCT01 ACCT02 ACCT03 ACCT04 **Transactions**: ACCT ACEL ACLG AC01 AC02 AC03 AC05 AC06
DFH$AFFY		Transaction Affinity Utility sample	**Files**: CAUAFF1 CAUAFF2 CAUAFF3 CAUCNTL
DFH$AFLA	Assembler	FILEA sample applications	**Map sets**: DFH$AGA DFH$AGB DFH$AGC DFH$AGD DFH$AGK DFH$AGL **Programs**: DFH$AALL DFH$ABRW DFH$ACOM DFH$AMNU DFH$AREN DFH$AREP **Transactions**: AADD ABRW AINQ AMNU AORD AORQ AREP AUPD
DFH$BMSP	COBOL & PL/I	BMS partition support applications	**Partitionset**: DFH0PS **Map sets**: DFH0CGP DFH$PGP **Programs**: DFH0CPKO DFH0CPLA DFH$PPKO DFH$PPLA **Transactions**: PPKO PPLA XPKO XPLA
DFH$CBTS		CICS BTS sample application	**Map sets**: DFH0PAS DFH0SAS **Programs**: DFH0DEL1 DFH0INV1 DFH0PAY0 DFH0PAY1 DFH0RED1 DFH0REM1 DFH0SAL0 DFH0SAL1 DFH0SAL2 DFH0STOC **Transactions**: DEL1 INV1 MENU PAYM PAY1 RED1 REM1 SALE SAL1 STOC **Process type**: ORDER
DFH$CFCS	LE370	CICS foundation classes sample application	**Programs**: ICC$BUF ICC$CLK ICC$CON ICC$DAT ICC$EXC1 ICC$EXC2 ICC$EXC3 ICC$FIL ICC$HEL ICC$JRN ICC$PRG1 ICC$PRG2 ICC$PRG3 ICC$RS1 ICC$RES2 ICC$SEM ICC$SES1 ICC$SES2 ICC$SRQ1 ICC$SRQ2 ICC$SYS ICC$TMP ICC$TRM ICC$TSK ICC$USR**Transactions**: IBUF ICLK ICON IDAT IXC1 IXC2 IXC3 IFIL IHEL I JRN IPG1 IPG2 IPG3 IRS1 IRS2 ISEM ISE1 ISE2 ISR1 ISR2 ISYS ITMP ITRM ITSK IUSR
DFH$CFLA	COBOL	FILEA sample applications	**Map sets**: DFH0CGA DFH0CGB DFH0CGC DFH0CGD DFH0CGK DFH0CGL **Programs**: DFH0CALL DFH0CBRW DFH0CCOM DFH0CMNU DFH0CREN DFH0CREP **Transactions**: ADDS BRWS INQY MENU OREN OREQ REPT UPDT
DFH$CNSL		Sample console definitions	**Typeterms**: DFH$JCLC DFH$CONS **Terminals**: CJCL CNSL CN02

Table 95. CICS sample applications - resource definitions (continued)

Group Name	Language	Description	Resources Defined
DFH$CTXT	COBOL	CUA® text model application	**Files**: DFH0FCAI DFH0FCUS DFH0FHLP **Map sets**: DFH0AB DFH0ABT DFH0BRW DFH0DEL DFH0FPD DFH0HLP DFH0HP DFH0HPD DFH0LST DFH0NEW DFH0OPN DFH0PRT DFH0SAS DFH0T1 DFH0UPD **Programs**: DFH0VAB DFH0VABT DFH0VBRW DFH0VDEL DFH0VDQ DFH0VHLP DFH0VHP DFH0VLIO DFH0VLST DFH0VNEW DFH0VOL DFH0VOPN DFH0PRT DFH0VRIO DFH0VSAS DFH0VTBL DFH0VT1 DFH0VUPD **Profile**: DFH$CUA2 **Transactions:**AC2A AC2C AC2D AC2E AC2F AC20 AC21 AC22 AC23 AC24 AC25 AC26 AC27 AC28 DELQ
DFH$DFLA	C/370	FILEA sample applications	**Map sets**: DFH$DGA DFH$DGB DFH$DGC DFH$DGD DFH$DGK DFH$DGL **Programs**: DFH$DALL DFH$DBRW DFH$DCOM DFH$DMNU DFH$DREN DFH$DREP **Transactions**: DADD DBRW DINQ DMNU DORD DORQ DREP DUPD
DFH$DLIV		IMS installation verification procedure	**Programs**: DFH$DLAC DFH$DLAE DFH$DLCC DFH$DLCE DFH$DLPC DFH$DLPE **Transactions:**ASMC ASME COBC COBE PLIC PLIE
DFH$EXCI		EXCI batch call interface samples	**Connections**: EXCG EXCS **Program**: DFH$AXCS **Sessions**: EXCG EXCS **Transaction**: EXCI
DFH$FILA		FILEA samples	**File:** FILEA
DFH$ICOM	Assembler	Intersystem communication (ISC)	**Map sets**: DFH$IGB DFH$IGC DFH$IGS DFH$IGX DFH$IG1 DFH$IG2 **Programs**: DFH$ICIC DFH$IFBL DFH$IFBR DFH$IMSN DFH$IMSO DFH$IQRD DFH$IQRL DFH$IQRR DFH$IQXL DFH$IQXR **Transactions**: ICIC IFBL IFBR IMSN IMSO IQRD IQRL IQRR IQXL IQXR
DFH$IIOP	LE370	IIOP sample application	**File**: BANKACCT **Map set**: BANKINQ **Programs**: DFH$IIBI DFH$IIBQ DFH$IICC DFJ$IIBS **Transactions**: BNKI BNKQ BNKS
DFH$JAVA	LE370	JAVA sample applications for CICS java support via VisualAge for Java, Enterprise ToolKit for OS/390	**Programs**: DFH$JSAM DFH$LCCA DFJ$JHE1 DFJ$JHE2 DFJ$JPC1 DFJ$JPC2 DFJ$JTD1 DFJ$JTS1 DFJ$JTSC **TDQueue**: JTD1 **Transactions**: JHE1 JHE2 JPC1 JPC2 JTD1 JTS1

Table 95. CICS sample applications - resource definitions (continued)

Group Name	Language	Description	Resources Defined
DFH$JVM	LE370	JAVA sample applications for CICS java support via JVM	**Programs**: DFH$JSAM DFH$LCCA DFJ$JHE1 DFJ$JHE2 DFJ$JPC1 DFJ$JPC2 DFJ$JTD1 DFJ$JTS1 DFJ$JTSC **TDQueue**: JTD1 **Transactions**: JHE1 JHE2 JPC1 JPC2 JTD1 JTS1
DFH$PFLA	PL/I	FILEA sample applications	**Map sets**: DFH$PGA DFH$PGB DFH$PGC DFH$PGD DFH$PGK DFH$PGL **Programs**: DFH$PALL DFH$PBRW DFH$PCOM DFH$PMNU DFH$PREN DFH$PREP **Transactions**: PADD PBRW PINQ PMNU PORD PORQ PREP PUPD
DFH$SOT	CICS-supplied TCPIPSERVICEs	**TCP/IP Services**: IIOPNSSL HTTPNSSL HTTPSSL	-
DFH$SIGN		Some kinds of transactions	**Map sets**: DFHSNLE DFHSNSE **Programs**: DFHCEGN DFHCESC DFHSFP DFHSNP **Transactions**: CEGN CESC CESF CESN
DFH$STAT		Statistics	**Program**: DFH$STED
DFH$SXP		Message domain exits	**Programs**: DFH$SXP1 DFH$SXP2 DFH$SXP3 DFH$SXP4 DFH$SXP5 DFH$SXP6
DFH$UTIL	Assembler	Transient data utility dynamic allocation	**Programs**: DFH$TDWT DFH99 **Transactions**: ADYN TDWT
DFH$VTAM		Terminal definitions	**Typeterms**: DFH$L77 DFH$L78 DFH$L79 DFH$L86 **Terminals**: L77C L77D L78A L79A L86A
DFHAI62		Starter APPC connections	**Program**: DFHZATDY **Connections**: CBPS CBSS CCPS **Sessions**: CBPS CBSS CCPS
DFHMROAR		Starter MRO systems	**Connections**: CICD CICT **Sessions**: CICSRD CICSRT
DFHMRODR		Starter MRO systems	**Connection**: CICA **Session**: CICSRA
DFHMROFA		Starter MRO systems	**File**: FILEA **Map sets**: DFH$AGA DFH$AGB DFH$AGC DFH$AGD DFH$AGK DFH$AGL **Programs**: DFH$AALL DFH$ABRW DFH$ACOM DFH$AMNU DFH$AREN DFH$AREP **Transactions**: AADD ABRW AINQ AMNU AORD AORQ AREP AUPD
DFHMROFD		Starter MRO systems	**File**: FILEA
DFHMROFT		Starter MRO systems	**Transactions**: AADD ABRW AINQ AMNU AORD AORQ AREP AUPD
DFHMROTR		Starter MRO systems	**Connection**: CICA **Session**: CICSRA

CICS transactions supplied by IBM

Here is a list in alphabetic order of the CICS transactions supplied by IBM, together with the name of the program that the transaction invokes.

Table 96. CICS transactions supplied by IBM

Transaction	Group	Program
AADD	DFH$AFLA	DFH$AALL
AADD	DFHMROFA	DFH$AALL
AADD	DFHMROFT	
ABRW	DFH$AFLA	DFH$ABRW
ABRW	DFHMROFA	DFH$ABRW
ABRW	DFHMROFT	
ACCT	DFH$ACCT	ACCT00
ACEL	DFH$ACCT	ACCT03
ACLG	DFH$ACCT	ACCT03
AC01	DFH$ACCT	ACCT01
AC02	DFH$ACCT	ACCT02
AC03	DFH$ACCT	ACCT03
AC05	DFH$ACCT	ACCT03
AC06	DFH$ACCT	ACCT03
AC2A	DFH$CTXT	DFH0VSAS
AC2C	DFH$CTXT	DFH0VHLP
AC2D	DFH$CTXT	DFH0VAB
AC2E	DFH$CTXT	DFH0VHP
AC2F	DFH$CTXT	DFH0VABT
AC20	DFH$CTXT	DFH0VT1
AC21	DFH$CTXT	DFH0VOL
AC22	DFH$CTXT	DFH0VOPN
AC23	DFH$CTXT	DFH0VLST
AC24	DFH$CTXT	DFH0VNEW
AC25	DFH$CTXT	DFH0VBRW
AC26	DFH$CTXT	DFH0VUPD
AC27	DFH$CTXT	DFH0VDEL
AC28	DFH$CTXT	DFH0VPRT
ADDS	DFH$CFLA	DFH0CALL
ADYN	DFH$UTIL	DFH99
AINQ	DFH$AFLA	DFH$AALL
AINQ	DFHMROFA	DFH$AALL
AINQ	DFHMROFT	
AMNU	DFH$AFLA	DFH$AMNU
AMNU	DFHMROFA	DFH$AMNU
AMNU	DFHMROFT	
AORD	DFH$AFLA	DFH$AREN
AORD	DFHMROFA	DFH$AREN
AORD	DFHMROFT	
AORQ	DFH$AFLA	DFH$ACOM
AORQ	DFHMROFA	DFH$ACOM
AORQ	DFHMROFT	
AREP	DFH$AFLA	DFH$AREP
AREP	DFHMROFA	DFH$AREP
AREP	DFHMROFT	
ASMC	DFH$DLIV	DFH$DLAC
ASME	DFH$DLIV	DFH$DLAE
AUPD	DFH$AFLA	DFH$AALL

Table 96. CICS transactions supplied by IBM (continued)

	Transaction	Group	Program
	AUPD	DFHMROFA	DFH$AALL
	AUPD	DFHMROFT	
	BRWS	DFH$CFLA	DFH0CBRW
	CAFB	DFHAUGRP	CAUCAFB1
	CAFF	DFHAUGRP	CAUCAFF1
	CATA	DFHSPI	DFHZATA
	CATD	DFHSPI	DFHZATD
	CATR	DFHSPI	DFHZATR
	CATS	DFHCOMP3	DFHZATS
	CAUT	DFHCOMP1	DFHSTSP
|	**CBAM**	DFHOPER	DFHECBAM
	CBRC	DFHDLI	DFHBRCP
#	**CCIN**	DFHCLNT	DFHZCN1
	CCMF	DFHCOMP1	DFHCCMF
	CDBC	DFHDBCTL	DFHDBME
	CDBD	DFHDBCTL	DFHDBDI
|	**CDBF**	DFHDB2	DFHD2CM3
	CDBI	DFHDBCTL	DFHDBIQ
	CDBN	DFHDBCTL	DFHDBCON
	CDBO	DFHDBCTL	DFHDBCT
|	**CDBQ**	DFHDB2	DFHD2CM2
	CDBT	DFHDBCTL	DFHDBDSC
|	**CDFS**	DFHISC	DFHDFST
	CDTS	DFHSPI	DFHZATS
	CEBR	DFHEDF	DFHEDFBR
	CECI	DFHINTER	DFHECIP
	CECS	DFHINTER	DFHECSP
	CEDA	DFHSPI	DFHEDAP
	CEDB	DFHSPI	DFHEDAP
	CEDC	DFHSPI	DFHEDAP
	CEDF	DFHEDF	DFHEDFP
	CEGN	DFHSIGN	DFHCEGN
	CEHP	DFHISC	DFHCHS
	CEHS	DFHISC	DFHCHS
	CEMT	DFHOPER	DFHEMTP
	CEOT	DFHOPER	DFHEOTP
	CESC	DFHSIGN	DFHCESC
	CESD	DFHSDAP	DFHCESD
	CESF	DFHSIGN	DFHSFP
	CESN	DFHSIGN	DFHSNP
	CEST	DFHOPER	DFHESTP
	CEST	DFHCOMP1	DFHESTP
	CEST	DFHCOMP3	DFHESTP
	CETR	DFHOPER	DFHCETRA
|	**CEX2**	DFHDB2	DFHD2EX2
#	**CFCL**	N/A	DFHFCDL
#	**CFOR**	N/A	DFHFCOR
#	**CFQR**	N/A	DFHFCQT
#	**CFQS**	N/A	DFHFCQT
#	**CFSL**	N/A	DFHDTLX
	CFTS	DFHSPI	DFHZATS
#	**CGRP**	N/A	DFHZCGRP
	CIND	DFHINDT	DFHINDT

Table 96. CICS transactions supplied by IBM (continued)

	Transaction	Group	Program
#	**CIOD**	DFHIIOP	DFHIIOPA
#	**CIOF**	DFHCLNT	DFHZCN1
#	**CIOR**	DFHIIOP	DFHIIOP
	CLQ2	DFHISC	DFHLUP
|	**CLR1**	DFHISCT	DFHZLS1
|	**CLR2**	DFHISC	DFHLUP
	CLS1	DFHISC	DFHZLS1
	CLS2	DFHISC	DFHLUP
	CLS3	DFHISC	DFHLUP
	CLS4	DFHISC	DFHCLS4
	CMAC	DFHCMAC	DFHCMAC
	CMPX	DFHISC	DFHMXP
	CMSG	DFHMSWIT	DFHMSP
	CMTS	DFHSPI	DFHZATS
	COBC	DFH$DLIV	DFH0DLCC
	COBE	DFH$DLIV	DFH0DLCE
#	**COVR**	N/A	DFHZCOVR
#	**CPLT**	N/A	DFHSIPLT
	CPMI	DFHISC	DFHMIRS
#	**CQPI**	DFHISC	DFHCLS5
#	**CQPO**	DFHISC	DFHCLS5
	CQRY	DFHSTAND	DFHQRY
#	**CRPA**	DFHRPC	DFHRPAS
#	**CRPC**	DFHRPC	DFHRPC00
#	**CRPM**	DFHRPC	DFHRPMS
	CRSQ	DFHISC	DFHCRQ
	CRSR	DFHISC	DFHCRS
	CRSY	DFHRMI	DFHRMSY
	CRTE	DFHISC	DFHRTE
	CRTX	DFHISC	########
	CSAC	DFHSTAND	DFHACP
	CSCY	DFHVTAMP	DFHCPY
	CSFE	DFHFE	DFHFEP
#	**CSFR**	N/A	DFHFCRD
	CSFU	DFHOPCLS	DFHFCU
	CSGM	DFHVTAM	DFHGMM
	CSGX	DFHDLI	DFHDLG
#	**CSHA**	N/A	N/A
#	**CSHQ**	N/A	DFHSHSY
#	**CSHR**	DFHISC	DFHCICSA
	CSIR	DFHCOMP1	DFHCRR
	CSJC	DFHCOMP3	DFHJCBSP
	CSKP	DFHAKP	DFHAKP
	CSLG	DFHRSPLG	DFHZRLG
	CSMI	DFHISC	DFHMIRS
	CSMT	DFHCOMP1	DFHMTPA
	CSM1	DFHISC	DFHMIRS
	CSM2	DFHISC	DFHMIRS
	CSM3	DFHISC	DFHMIRS
	CSM5	DFHISC	DFHMIRS
	CSNC	DFHISC	DFHCRNP
	CSNC	DFHCOMP1	DFHCRNP
	CSNC	DFHCOMP2	DFHCRNP

Table 96. CICS transactions supplied by IBM (continued)

	Transaction	Group	Program
	CSNE	DFHVTAM	DFHZNAC
#	**CSOP**	N/A	N/A
	CSOT	DFHCOMP1	DFHMTPA
	CSPG	DFHBMS	DFHTPR
	CSPK	DFHVTAMP	DFHPRK
	CSPP	DFHHARDC	DFHP3270
	CSPQ	DFHBMS	DFHTPQ
	CSPS	DFHBMS	DFHTPS
#	**CSQC**	DFHLGQC	DFHLGQC
	CSRK	DFHVTAMP	DFHRKB
	CSRS	DFHRSEND	DFHZRSP
	CSSC	DFHSIGN	DFHCSSC
	CSSF	DFHISC	DFHRTC
	CSSN	DFHCOMP1	DFHSNP
	CSST	DFHCOMP1	DFHMTPA
	CSSX	DFHDLI	DFHDLS
#	**CSSY**	N/A	DFHAPATT
	CSTA	DFHCOMP1	DFHTAJP
	CSTE	DFHSTAND	DFHTACP
#	**CSTP**	N/A	DFHZCSTP
	CSTT	DFHCOMP1	DFHSTKC
	CSZI	DFHFEPI	DFHSZRMP
#	**CTIN**	DFHCLNT	DFHZCT1
#	**CTSD**	N/A	DFHTSDQ
	CVMI	DFHISC	DFHMIRS
	CVST	DFHCOMP1	DFHVAP
#	**CWBA**	DFHWEB	DFHWBA
#	**CWBC**	DFHWEB	DFHWBC00
#	**CWBG**	DFHWEB	DFHWBGB
#	**CWBM**	DFHWEB	DFHWBM
	CWTO	DFHCONS	DFHCWTO
#	**CWXN**	DFHWEB	DFHWBXN
	CXCU	DFHSTAND	DFHCXCU
	CXRE	DFHSTAND	DFHZXRE
	CXRT	DFHISC	DFHCRT
	DADD	DFH$DFLA	DFH$DALL
	DBRW	DFH$DFLA	DFH$DBRW
	DELQ	DFH$CTXT	DFH0VDQ
	DINQ	DFH$DFLA	DFH$DALL
	DMNU	DFH$DFLA	DFH$DMNU
	DORD	DFH$DFLA	DFH$DREN
	DORQ	DFH$DFLA	DFH$DCOM
	DREP	DFH$DFLA	DFH$DREP
|	**DSNC**	DFHDB2	DFHD2CM1
	DUPD	DFH$DFLA	DFH$DALL
	EXCI	DFH$EXCI	DFH$MIRS
	ICIC	DFH$ICOM	DFH$ICIC
	IFBL	DFH$ICOM	DFH$IFBL
	IFBR	DFH$ICOM	DFH$IFBR
	IMSN	DFH$ICOM	DFH$IMSN
	IMSO	DFH$ICOM	DFH$IMSO
	INQY	DFH$CFLA	DFH0CALL
	IQRD	DFH$ICOM	DFH$IQRD

Table 96. CICS transactions supplied by IBM (continued)

Transaction	Group	Program
IQRL	DFH$ICOM	DFH$IQRL
IQRR	DFH$ICOM	DFH$IQRR
IQXL	DFH$ICOM	DFH$IQXL
IQXR	DFH$ICOM	DFH$IQXR
MENU	DFH$CFLA	DFH0CMNU
OREN	DFH$CFLA	DFH0CREN
OREQ	DFH$CFLA	DFH0CCOM
PADD	DFH$PFLA	DFH$PALL
PBRW	DFH$PFLA	DFH$PBRW
PINQ	DFH$PFLA	DFH$PALL
PLIC	DFH$DLIV	DFH$DLPC
PLIE	DFH$DLIV	DFH$DLPE
PMNU	DFH$PFLA	DFH$PMNU
PORD	DFH$PFLA	DFH$PREN
PORQ	DFH$PFLA	DFH$PCOM
PPKO	DFH$BMSP	DFH$PPKO
PPLA	DFH$BMSP	DFH$PPLA
PREP	DFH$PFLA	DFH$PREP
PUPD	DFH$PFLA	DFH$PALL
REPT	DFH$CFLA	DFH0CREP
TDWT	DFH$UTIL	DFH$TDWT
UPDT	DFH$CFLA	DFH0CALL
XPKO	DFH$BMSP	DFH0CPKO
XPLA	DFH$BMSP	DFH0CPLA

TYPETERM definitions in group DFHTYPE

The CICS-supplied CSD group DFHTYPE contains the following TYPETERM definitions:

DFHCONS
MVS console

DFHLU2
SNA logical unit type 2 (3270 displays)

DFHLU3
SNA logical unit type 3 (3270 printers)

DFHSCSP
SNA logical unit type 1 (3270 SCS printers)

DFH3270
Locally attached (non-SNA) 3270 displays

DFH3270P
Locally attached (non-SNA) 3270 printers

DFHLU62T
SNA logical unit type 6.2 (APPC) single session terminal.

DFHLU0E2
Non-SNA model 2 with extended data stream (Query)

DFHLU0M2
Non-SNA model 2

Supplied TYPETERMs

DFHLU0M3
: Non-SNA model 3

DFHLU0M4
: Non-SNA model 4

DFHLU0M5
: Non-SNA model 5

DFHLU2E2
: SNA LU type 2 model 2 with extended data stream (Query)

DFHLU2M2
: SNA LU type 2 model 2

DFHLU2E3
: SNA LU type 2 model 3 with extended data stream (Query)

DFHLU2M3
: SNA LU type 2 model 3

DFHLU2E4
: SNA LU type 2 model 4 with extended data stream (Query)

DFHLU2M4
: SNA LU type 2 model 4

DFHLU2M5
: SNA LU type 2 model 5

DFHLU2E5
: SNA LU type 2 model 5 with extended data stream (Query)

The following figures show the contents of each definition.

```
TYPETERM(DFHCONS)  GROUP(DFHTYPE)
DEVICE(CONSOLE)
PAGESIZE(1,124)    AUTOPAGE(NO)
BRACKET(YES)       BUILDCHAIN(YES)     ROUTEDMSGS(NONE)
UCTRAN(YES)
```

Figure 64. DFHCONS TYPETERM definition—MVS console

```
TYPETERM(DFHLU2)   GROUP(DFHTYPE)
DEVICE(LUTYPE2)    TERMMODEL(2)
DEFSCREEN(24,80)   PAGESIZE(24,80)     AUTOPAGE(NO)
BRACKET(YES)       BUILDCHAIN(YES)     ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)  EXTENDEDDS(YES)     UCTRAN(YES)
SENDSIZE(1536)     RECEIVESIZE(256)    IOAREALEN(256,4000)
ERRLASTLINE(YES)   ERRINTENSIFY(YES)   ATI(YES)  TTI(YES)
DISCREQ(YES)       RELREQ(YES)         AUTOCONNECT(YES)
LOGONMSG(YES)      QUERY(ALL)          CREATESESS(NO)
```

Figure 65. DFHLU2 TYPETERM definition—SNA LU type 2. This definition is for a 3278 Model 2 display. It is suitable for the following devices: 3178, 3179, 3277, 3278, 3279, 3290, 3270PC, 3270PC/G, 3270PC/GX, 8775, and 5550.

```
TYPETERM(DFHLU3)   GROUP(DFHTYPE)
DEVICE(LUTYPE3)    TERMMODEL(2)
DEFSCREEN(24,80)   PAGESIZE(24,80)     AUTOPAGE(YES)
BRACKET(YES)       BUILDCHAIN(NO)      ROUTEDMSGS(ALL)
SENDSIZE(256)      RECEIVESIZE(256)    IOAREALEN(512,0)
EXTENDEDDS(YES)    QUERY(ALL)          ATI(YES)  TTI(YES)
DISCREQ(YES)       RELREQ(YES)         AUTOCONNECT(YES)
LOGONMSG(NO)                           CREATESESS(NO)
```

Figure 66. DFHLU3 TYPETERM definition—SNA LU type 3. This definition is for a 3287 printer. It is suitable for the following devices: 3262, 3268, 3284, 3286, 3287, 3288, 3289, and 5550.

```
TYPETERM(DFHSCSP)  GROUP(DFHTYPE)
DEVICE(SCSPRINT)
PAGESIZE(24,80)    AUTOPAGE(YES)
BRACKET(YES)       BUILDCHAIN(NO)      ROUTEDMSGS(ALL)
SENDSIZE(256)      RECEIVESIZE(256)    IOAREALEN(512,0)
EXTENDEDDS(YES)    QUERY(ALL)          ATI(YES)  TTI(YES)
DISCREQ(YES)       RELREQ(YES)         AUTOCONNECT(YES)
LOGONMSG(NO)                           CREATESESS(NO)
```

Figure 67. DFHSCSP TYPETERM definition—SNA LU type 1. This definition is for a 3287 printer. It is suitable for the following devices: 3262, 3268, 3287, 3289, and 5550.

```
TYPETERM(DFH3270)  GROUP(DFHTYPE)
DEVICE(3270)       TERMMODEL(2)
DEFSCREEN(24,80)   PAGESIZE(24,80)     AUTOPAGE(NO)
BRACKET(YES)       BUILDCHAIN(NO)      ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)  EXTENDEDDS(YES)     UCTRAN(YES)
SENDSIZE(0)        RECEIVESIZE(0)   IOAREALEN(512,0)
ERRLASTLINE(YES)   ERRINTENSIFY(YES)   ATI(YES)  TTI(YES)
DISCREQ(YES)       RELREQ(YES)         AUTOCONNECT(YES)
LOGONMSG(YES)      QUERY(ALL)          CREATESESS(NO)
```

Figure 68. DFH3270 TYPETERM definition—locally attached (non-SNA). This definition is for a 3278 Model 2 display. It is suitable for the following devices: 3178, 3179, 3277, 3278, 3279, and 3290.

```
TYPETERM(DFH3270P) GROUP(DFHTYPE)
DEVICE(3270P)      TERMMODEL(2)
DEFSCREEN(24,80)   PAGESIZE(24,80)     AUTOPAGE(YES)
BRACKET(YES)       BUILDCHAIN(NO)      ROUTEDMSGS(ALL)
SENDSIZE(0)        RECEIVESIZE(0)      IOAREALEN(512,0)
EXTENDEDDS(YES)    QUERY(ALL)          ATI(YES)  TTI(YES)
DISCREQ(YES)       RELREQ(YES)         AUTOCONNECT(YES)
LOGONMSG(NO)                           CREATESESS(NO)
```

Figure 69. DFH3270P TYPETERM definition—locally attached (non-SNA). This definition is for a 3284 Model 2 printer. It is suitable for the following devices: 3262, 3268, 3284, 3287, 3288, 3289, and 5550.

```
TYPETERM(DFHLU62T) GROUP(DFHTYPE)
DEVICE(APPC)
                   PAGESIZE(1,40)      AUTOPAGE(YES)
BRACKET(YES)       BUILDCHAIN(YES)     ROUTEDMSGS(NONE)
SENDSIZE(2048)     RECEIVESIZE(2048)   IOAREALEN(0,0)
                                       ATI(YES) TTI(YES)
LOGONMSG(NO)                           CREATESESS(NO)
```

Figure 70. DFHLU62T TYPETERM definition—APPC (LU6.2) terminal. This definition is for an APPC single session terminal and is also suitable for the following devices: DISPLAYWRITER, SCANMASTER, and SYSTEM/38.

```
TYPETERM(DFHLU0E2) GROUP(DFHTYPE)
DEVICE(3270)       TERMMODEL(2)        SHIPPABLE(YES)
DEFSCREEN(24,80)   PAGESIZE(24,80)     AUTOPAGE(NO)
ALTSCREEN(0,0)     ALTPAGE(0,0)
BRACKET(YES)       BUILDCHAIN(NO)      ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)  EXTENDEDDS(YES)     UCTRAN(YES)
RECEIVESIZE(0)     SENDSIZE(0)         IOAREALEN(512,0)
ERRASTLINE(YES)    ERRINTENSIFY(YES)   QUERY(ALL)
DISCREQ(YES)       RELREQ(YES)         AUTOCONNECT(NO)
LOGONMSG(YES)      CREATESESS(NO)
ATI(YES)           TTI(YES)
```

Figure 71. DFHLU0E2 TYPETERM definition. Non-SNA model 2 with extended data stream (Query). This definition matches the VTAM-supplied LOGMODE NSX32702

```
TYPETERM(DFHLU0M2)  GROUP(DFHTYPE)
DEVICE(3270)        TERMMODEL(2)         SHIPPABLE(YES)
DEFSCREEN(24,80)    PAGESIZE(24,80)      AUTOPAGE(NO)
ALTSCREEN(0,0)      ALTPAGE(0,0)
BRACKET(YES)        BUILDCHAIN(NO)       ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)   EXTENDEDDS(NO)       UCTRAN(YES)
RECEIVESIZE(0)      SENDSIZE(0)          IOAREALEN(512,0)
ERRLASTLINE(YES)    ERRINTENSIFY(YES)    QUERY(NO)
DISCREQ(YES)        RELREQ(YES)          AUTOCONNECT(NO)
LOGONMSG(YES)       CREATESESS(NO)
ATI(YES)            TTI(YES)
```

Figure 72. DFHLU0M2 TYPETERM definition. Non-SNA model 2 with extended data stream (Query). This definition matches the VTAM-supplied LOGMODE D4B32782

```
TYPETERM(DFHLU0M3)  GROUP(DFHTYPE)
DEVICE(3270)        TERMMODEL(2)         SHIPPABLE(YES)
DEFSCREEN(24,80)    PAGESIZE(24,80)      AUTOPAGE(NO)
ALTSCREEN(32,80)    ALTPAGE(32,80)
BRACKET(YES)        BUILDCHAIN(NO)       ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)   EXTENDEDDS(NO)       UCTRAN(YES)
RECEIVESIZE(0)      SENDSIZE(0)          IOAREALEN(512,0)
ERRLASTLINE(YES)    ERRINTENSIFY(YES)    QUERY(NO)
DISCREQ(YES)        RELREQ(YES)          AUTOCONNECT(NO)
LOGONMSG(YES)       CREATESESS(NO)
ATI(YES)            TTI(YES)
```

Figure 73. DFHLU0M3 TYPETERM definition—non-SNA model 3. This definition matches the VTAM-supplied LOGMODE D4B32783.

```
TYPETERM(DFHLU0M4)      GROUP(DFHTYPE)
DEVICE(3270)            TERMMODEL(2)            SHIPPABLE(YES)
DEFSCREEN(24,80)        PAGESIZE(24,80)         AUTOPAGE(NO)
ALTSCREEN(43,80)        ALTPAGE(43,80)
BRACKET(YES)            BUILDCHAIN(NO)          ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)       EXTENDEDDS(NO)          UCTRAN(YES)
RECEIVESIZE(0)          SENDSIZE(0)             IOAREALEN(512,0)
ERRLASTLINE(YES)        ERRINTENSIFY(YES)       QUERY(NO)
DISCREQ(YES)            RELREQ(YES)             AUTOCONNECT(NO)
LOGONMSG(YES)           CREATESESS(NO)
ATI(YES)                TTI(YES)
```

Figure 74. DFHLU0M4 TYPETERM definition—non-SNA model 4. This definition matches the VTAM-supplied LOGMODE D4B32784.

```
TYPETERM(DFHLU0M5)      GROUP(DFHTYPE)
DEVICE(3270)            TERMMODEL(2)            SHIPPABLE(YES)
DEFSCREEN(24,80)        PAGESIZE(24,80)         AUTOPAGE(NO)
ALTSCREEN(27,132)       ALTPAGE(27,132)
BRACKET(YES)            BUILDCHAIN(NO)          ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)       EXTENDEDDS(NO)          UCTRAN(YES)
RECEIVESIZE(0)          SENDSIZE(0)             IOAREALEN(512,0)
ERRLASTLINE(YES)        ERRINTENSIFY(YES)       QUERY(NO)
DISCREQ(YES)            RELREQ(YES)             AUTOCONNECT(NO)
LOGONMSG(YES)           CREATESESS(NO)
ATI(YES)                TTI(YES)
```

Figure 75. DFHLU0M5 TYPETERM definition—non-SNA model 5. This definition matches the VTAM-supplied LOGMODE D4B32785.

```
TYPETERM(DFHLU2E2)      GROUP(DFHTYPE)
DEVICE(LUTYPE2)         TERMMODEL(2)            SHIPPABLE(YES)
DEFSCREEN(24,80)        PAGESIZE(24,80)         AUTOPAGE(NO)
ALTSCREEN(0,0)          ALTPAGE(0,0)
BRACKET(YES)            BUILDCHAIN(YES)         ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)       EXTENDEDDS(YES)         UCTRAN(YES)
RECEIVESIZE(1024)       SENDSIZE(3840)          IOAREALEN(256,4000)
ERRLASTLINE(YES)        ERRINTENSIFY(YES)       QUERY(ALL)
DISCREQ(YES)            RELREQ(YES)             AUTOCONNECT(NO)
LOGONMSG(YES)           CREATESESS(NO)
ATI(YES)                TTI(YES)
```

Figure 76. DFHLU2E2 TYPETERM definition. SNA LU type 2 model 2 with extended data stream (Query). This definition matches the VTAM-supplied LOGMODE SNX32702.

```
TYPETERM(DFHLU2M2)      GROUP(DFHTYPE)
DEVICE(LUTYPE2)         TERMMODEL(2)            SHIPPABLE(YES)
DEFSCREEN(24,80)        PAGESIZE(24,80)         AUTOPAGE(NO)
ALTSCREEN(0,0)          ALTPAGE(0,0)
BRACKET(YES)            BUILDCHAIN(YES)         ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)       EXTENDEDDS(NO)          UCTRAN(YES)
RECEIVESIZE(1024)       SENDSIZE(1536)          IOAREALEN(256,4000)
ERRLASTLINE(YES)        ERRINTENSIFY(YES)       QUERY(NO)
DISCREQ(YES)            RELREQ(YES)             AUTOCONNECT(NO)
LOGONMSG(YES)           CREATESESS(NO)
ATI(YES)                TTI(YES)
```

Figure 77. DHLU2M2 TYPETERM definition—SNA LU type 2 model 2. This definition matches the VTAM-supplied LOGMODE D4A32782.

```
TYPETERM(DFHLU2E3)      GROUP(DFHTYPE)
DEVICE(LUTYPE2)         TERMMODEL(2)            SHIPPABLE(YES)
DEFSCREEN(24,80)        PAGESIZE(24,80)         AUTOPAGE(NO)
ALTSCREEN(32,80)        ALTPAGE(32,80)
BRACKET(YES)            BUILDCHAIN(YES)         ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)       EXTENDEDDS(YES)         UCTRAN(YES)
RECEIVESIZE(1024)       SENDSIZE(3840)          IOAREALEN(256,4000)
ERRLASTLINE(YES)        ERRINTENSIFY(YES)       QUERY(ALL)
DISCREQ(YES)            RELREQ(YES)             AUTOCONNECT(NO)
LOGONMSG(YES)           CREATESESS(NO)
ATI(YES)                TTI(YES)
```

Figure 78. DFHLU2E3 TYPETERM definition. SNA LU type 2 model 3 with extended data stream (Query). This definition matches the VTAM-supplied LOGMODE SNX32703.

```
TYPETERM(DFHLU2M3)      GROUP(DFHTYPE)
DEVICE(LUTYPE2)         TERMMODEL(2)            SHIPPABLE(YES)
DEFSCREEN(24,80)        PAGESIZE(24,80)         AUTOPAGE(NO)
ALTSCREEN(32,80)        ALTPAGE(32,80)
BRACKET(YES)            BUILDCHAIN(YES)         ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)       EXTENDEDDS(NO)          UCTRAN(YES)
RECEIVESIZE(1024)       SENDSIZE(1536)          IOAREALEN(256,4000)
ERRLASTLINE(YES)        ERRINTENSIFY(YES)       QUERY(NO)
DISCREQ(YES)            RELREQ(YES)             AUTOCONNECT(NO)
LOGONMSG(YES)           CREATESESS(NO)
ATI(YES)                TTI(YES)
```

Figure 79. DFHLU2M3 TYPETERM definition—SNA LU type 2 model 3. This definition matches the VTAM-supplied LOGMODE D4A32783.

```
TYPETERM(DFHLU2E4)      GROUP(DFHTYPE)
DEVICE(LUTYPE2)         TERMMODEL(2)            SHIPPABLE(YES)
DEFSCREEN(24,80)        PAGESIZE(24,80)         AUTOPAGE(NO)
ALTSCREEN(43,80)        ALTPAGE(43,80)
BRACKET(YES)            BUILDCHAIN(YES)         ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)       EXTENDEDDS(YES)         UCTRAN(YES)
RECEIVESIZE(1024)       SENDSIZE(3840)          IOAREALEN(256,4000)
ERRLASTLINE(YES)        ERRINTENSIFY(YES)       QUERY(ALL)
DISCREQ(YES)            RELREQ(YES)             AUTOCONNECT(NO)
LOGONMSG(YES)           CREATESESS(NO)
ATI(YES)                TTI(YES)
```

Figure 80. DFHLU2E4 TYPETERM definition. SNA LU type 2 model 4 with extended data stream (Query). This definition matches the VTAM-supplied LOGMODE SNX32704.

```
TYPETERM(DFHLU2M4)      GROUP(DFHTYPE)
DEVICE(LUTYPE2)         TERMMODEL(2)            SHIPPABLE(YES)
DEFSCREEN(24,80)        PAGESIZE(24,80)         AUTOPAGE(NO)
ALTSCREEN(43,80)        ALTPAGE(43,80)
BRACKET(YES)            BUILDCHAIN(YES)         ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)       EXTENDEDDS(NO)          UCTRAN(YES)
RECEIVESIZE(1024)       SENDSIZE(1536)          IOAREALEN(256,4000)
ERRLASTLINE(YES)        ERRINTENSIFY(YES)       QUERY(NO)
DISCREQ(YES)            RELREQ(YES)             AUTOCONNECT(NO)
LOGONMSG(YES)           CREATESESS(NO)
ATI(YES)                TTI(YES)
```

Figure 81. DFHLU2M4 TYPETERM definition—SNA LU type 2 model 4. This definition matches the VTAM-supplied LOGMODE D4A32784.

```
TYPETERM(DFHLU2E5)      GROUP(DFHTYPE)
DEVICE(LUTYPE2)         TERMMODEL(2)            SHIPPABLE(YES)
DEFSCREEN(24,80)        PAGESIZE(24,80)         AUTOPAGE(NO)
ALTSCREEN(27,132)       ALTPAGE(27,132)
BRACKET(YES)            BUILDCHAIN(YES)         ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)       EXTENDEDDS(NO)          UCTRAN(YES)
RECEIVESIZE(1024)       SENDSIZE(3480)          IOAREALEN(256,4000)
ERRLASTLINE(YES)        ERRINTENSIFY(YES)       QUERY(ALL)
DISCREQ(YES)            RELREQ(YES)             AUTOCONNECT(NO)
LOGONMSG(YES)           CREATESESS(NO)
ATI(YES)                TTI(YES)
```

Figure 82. DFHLU2E5 TYPETERM definition—SNA LU type 2 model 5. SNA LU type 2 model 5 with extended data stream (Query). This definition matches the VTAM-supplied LOGMODE LSX32705.

```
TYPETERM(DFHLU2M5)      GROUP(DFHTYPE)
DEVICE(LUTYPE2)         TERMMODEL(2)            SHIPPABLE(YES)
DEFSCREEN(24,80)        PAGESIZE(24,80)         AUTOPAGE(NO)
ALTSCREEN(27,132)       ALTPAGE(27,132)
BRACKET(YES)            BUILDCHAIN(YES)         ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)       EXTENDEDDS(NO)          UCTRAN(YES)
RECEIVESIZE(1024)       SENDSIZE(1536)          IOAREALEN(256,4000)
ERRLASTLINE(YES)        ERRINTENSIFY(YES)       QUERY(NO)
DISCREQ(YES)            RELREQ(YES)             AUTOCONNECT(NO)
LOGONMSG(YES)           CREATESESS(NO)
ATI(YES)                TTI(YES)
```

Figure 83. DFHLU2M5 TYPETERM definition—SNA LU type 2 model 5. This definition matches the VTAM-supplied LOGMODE D4A32785.

Model TERMINAL definitions in group DFHTERM

The CICS-supplied CSD group DFHTERM contains the following model TERMINAL definitions for automatic installation:

LU2	SNA 3270 Model 2 display using TYPETERM DFHLU2
LU3	SNA 3270 Model 2 printer using TYPETERM DFHLU3
SCSP	SNA 3278 Model 2 printer using TYPETERM DFHSCSP
3270	Non-SNA 3270 Model 2 display using TYPETERM DFH3270
3284	Non-SNA 3270 Model 2 printer using TYPETERM DFH3270P
LU62	APPC (LU6.2) single session terminal using TYPETERM DFHLU62T
L0E2	Non-SNA Model 2 display using TYPETERM DFHLU0E2
L0M2	Non-SNA Model 2 display using TYPETERM DFHLU0M2
L0M3	Non-SNA Model 3 display using TYPETERM DFHLU0M3
L0M4	Non-SNA Model 4 display using TYPETERM DFHLU0M4
L0M5	Non-SNA Model 5 display using TYPETERM DFHLU0M5
L2E2	SNA LU2 Model 2 display using TYPETERM DFHLU2E2
L2M2	SNA LU2 Model 2 display using TYPETERM DFHLU2M2
L2E3	SNA LU2 Model 3 display using TYPETERM DFHLU2E3
L2M3	SNA LU2 Model 3 display using TYPETERM DFHLU2M3

Supplied model TERMINALs

L2E4 SNA LU2 Model 4 display using TYPETERM DFHLU2E4

L2M4 SNA LU2 Model 4 display using TYPETERM DFHLU2M4

L2E5 SNA LU2 Model 5 display using TYPETERM DFHLU2E5

L2M5 SNA LU2 Model 5 display using TYPETERM DFHLU2M5

The following figures show the contents of each definition.

```
TERMINAL(LU2)       GROUP(DFHTERM)      TYPETERM(DFHLU2)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU2)
```

Figure 84. LU2 model TERMINAL definition—SNA LU type 2. This definition is for a 3278 Model 2 display. It is suitable for the following devices: 3178, 3179, 3277, 3278, 3279, 3290, 3270PC, 3270PC/G, 3270PC/GX, 8775, and 5550.

```
TERMINAL(LU3)       GROUP(DFHTERM)      TYPETERM(DFHLU3)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU3)
```

Figure 85. LU3 model TERMINAL definition—SNA LU type 3. This definition is for a 3287 printer. It is suitable for the following devices: 3262, 3268, 3284, 3286, 3287, 3288, 3289, and 5550.

```
TERMINAL(SCSP)      GROUP(DFHTERM)      TYPETERM(DFHSCSP)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHSCSP)
```

Figure 86. SCSP model TERMINAL definition—SNA LU type 1. This definition is for a 3287 printer. It is suitable for the following devices: 3262, 3268, 3287, 3289, and 5550.

```
TERMINAL(3270)      GROUP(DFHTERM)      TYPETERM(DFH3270)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFH3270)
```

Figure 87. 3270 model TERMINAL definition—locally attached (non-SNA). This definition is for a 3278 Model 2 display. It is suitable for the following devices: 3178, 3179, 3277, 3278, 3279, and 3290.

```
TERMINAL(3284)      GROUP(DFHTERM)      TYPETERM(DFH3270P)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFH3270P)
```

Figure 88. 3284 model TERMINAL definition—locally attached (non-SNA). This definition is for a 3284 Model 2 printer. It is suitable for the following devices: 3262, 3268, 3284, 3287, 3288, 3289, and 5550.

```
TERMINAL(LU62)      GROUP(DFHTERM)       TYPETERM(DFHLU62T)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU62T)
```

Figure 89. LU62 model TERMINAL definition—APPC (LU6.2) terminal. This definition is for an APPC single session terminal and is also suitable for the following devices: DISPLAYWRITER, SCANMASTER, and SYSTEM/38.

```
TERMINAL(L0E2)      GROUP(DFHTERM)       TYPETERM(DFHLU0E2)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU0E2)
```

Figure 90. L0E2 model TERMINAL definition. Non-SNA model 2 with extended data stream (Query). This definition matches the VTAM-supplied LOGMODE NSX32702.

```
TERMINAL(L0M2)      GROUP(DFHTERM)      TYPETERM(DFHLU0M2)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU0M2)
```

Figure 91. L0M2 model TERMINAL definition—non-SNA model 2. This definition matches the VTAM-supplied LOGMODE D4B32782.

```
TERMINAL(L0M3)      GROUP(DFHTERM)      TYPETERM(DFHLU0M3)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU0M3)
```

Figure 92. L0M3 model TERMINAL definition—non-SNA model 3. This definition matches the VTAM-supplied LOGMODE D4B32783.

```
TERMINAL(L0M4)      GROUP(DFHTERM)       TYPETERM(DFHLU0M4)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU0M4)
```

Figure 93. L0M4 model TERMINAL definition—non-SNA model 4. This definition matches the VTAM-supplied LOGMODE D4B32784.

```
TERMINAL(L0M5)      GROUP(DFHTERM)       TYPETERM(DFHLU0M5)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU0M5)
```

Figure 94. L0M5 model TERMINAL definition—non-SNA model 5. This definition matches the VTAM-supplied LOGMODE D4B32785.

```
TERMINAL(L2E2)      GROUP(DFHTERM)       TYPETERM(DFHLU2E2)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU2E2)
```

Figure 95. L2E2 model TERMINAL definition. SNA LU type 2 model 2 with extended data stream (Query). This definition matches the VTAM-supplied LOGMODE SNX32702.

```
TERMINAL(L2M2)      GROUP(DFHTERM)       TYPETERM(DFHLU2M2)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU2M2)
```

Figure 96. L2M2 model TERMINAL definition—SNA LU type 2 model 2. This definition matches the VTAM-supplied LOGMODE D4A32782.

```
TERMINAL(L2E3)      GROUP(DFHTERM)       TYPETERM(DFHLU2E3)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU2E3)
```

Figure 97. L2E3 model TERMINAL definition. SNA LU type 2 model 3 with extended data stream (Query). This definition matches the VTAM-supplied LOGMODE SNX32703.

```
TERMINAL(L2M3)      GROUP(DFHTERM)       TYPETERM(DFHLU2M3)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU2M3)
```

Figure 98. L2M3 model TERMINAL definition—SNA LU type 2 model 3. This definition matches the VTAM-supplied LOGMODE D4A32783.

```
TERMINAL(L2E4)      GROUP(DFHTERM)       TYPETERM(DFHLU2E4)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU2E4)
```

Figure 99. L2E4 model TERMINAL definition. SNA LU type 2 model 4 with extended data stream (Query). This definition matches the VTAM-supplied LOGMODE SNX32704.

Supplied model TERMINALs

```
TERMINAL(L2M4)      GROUP(DFHTERM)        TYPETERM(DFHLU2M4)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU2M4)
```

Figure 100. L2M4 model TERMINAL definition—SNA LU type 2 model 4. This definition matches the VTAM-supplied LOGMODE D4A32784.

```
TERMINAL(L2E5)      GROUP(DFHTERM)        TYPETERM(DFHLU2E5)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU2E5)
```

Figure 101. L2E5 model TERMINAL definition. SNA LU type 2 model 5 with extended data stream (Query). This definition matches the VTAM-supplied LOGMODE LSX32705.

```
TERMINAL(L2M5)      GROUP(DFHTERM)        TYPETERM(DFHLU2M5)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU2M5)
```

Figure 102. L2M5 model TERMINAL definition—SNA LU type 2 model 5. This definition matches the VTAM-supplied LOGMODE D4A32785.

PROFILE definitions in group DFHISC

The CICS-supplied CSD group DFHISC contains the following PROFILE definitions for intersystem communication sessions.

DFHCICSF

CICS uses this profile for the session to the remote system or region when a CICS application program issues a function shipping request.

The definition is as follows:

```
PROFILE(DFHCICSF)
GROUP(DFHISC)   SCRNSIZE(DEFAULT)   PRINTERCOMP(NO)
JOURNALLING     MSGJRNL(NO)
PROTECTION      MSGINTEG(NO)        ONEWTE(NO)   PROTECT(NO)
PROTOCOLS       DVSUPRT(ALL)        INBFMH(ALL)  RAQ(NO)
                LOGREC(NO)
RECOVERY        NEPCLASS(0)
```

Figure 103. DFHCICSF profile definition

DFHCICSR

CICS uses this profile in transaction routing for communication between the user transaction (running in the application-owning region) and the interregion link or APPC link.

The definition is as follows:

```
PROFILE(DFHCICSR)
GROUP(DFHISC)   SCRNSIZE(DEFAULT)   PRINTERCOMP(NO)
JOURNALLING     MSGJRNL(NO)
PROTECTION      MSGINTEG(NO)        ONEWTE(NO)   PROTECT(NO)
PROTOCOLS       DVSUPRT(ALL)        INBFMH(ALL)  RAQ(NO)
                LOGREC(NO)
RECOVERY        NEPCLASS(0)
```

Figure 104. DFHCICSR profile definition

DFHCICSS

CICS uses this profile in transaction routing for communication between the relay transaction (running in the terminal-owning region) and the interregion link or APPC link. You can specify a different profile by means of the TRPROF option on the TRANSACTION definition.

The definition is as follows:

```
PROFILE(DFHCICSS)
GROUP(DFHISC)   SCRNSIZE(DEFAULT)   PRINTERCOMP(NO)
JOURNALLING     MSGJRNL(NO)
PROTECTION      MSGINTEG(NO)        ONEWTE(NO)   PROTECT(NO)
PROTOCOLS       DVSUPRT(ALL)        INBFMH(ALL)  RAQ(NO)
                LOGREC(NO)
RECOVERY        NEPCLASS(0)
```

Figure 105. DFHCICSS profile definition

PROFILE definitions in group DFHSTAND

The CICS-supplied CSD group DFHSTAND contains the following PROFILE definitions.

DFHCICSA

This is the default profile for alternate facilities acquired by the application program ALLOCATE command. A different profile can be named explicitly on the ALLOCATE command.

The definition is as follows:

```
PROFILE(DFHCICSA)
GROUP(DFHSTAND) SCRNSIZE(DEFAULT)   PRINTERCOMP(NO)
JOURNALLING     MSGJRNL(NO)
PROTECTION      MSGINTEG(NO)        ONEWTE(NO)   PROTECT(NO)
PROTOCOLS       DVSUPRT(ALL)        INBFMH(ALL)  RAQ(NO)
                LOGREC(NO)
RECOVERY        NEPCLASS(0)
```

Figure 106. DFHCICSA profile definition

DFHCICSE

This is the error profile for principal facilities. CICS uses this profile to pass an error message to the principal facility when the required profile cannot be found.

The definition is as follows:

```
PROFILE(DFHCICSE)
GROUP(DFHSTAND) SCRNSIZE(DEFAULT)   PRINTERCOMP(NO)
JOURNALLING     MSGJRNL(NO)
PROTECTION      MSGINTEG(NO)        ONEWTE(NO)   PROTECT(NO)
PROTOCOLS       DVSUPRT(ALL)        INBFMH(NO)   RAQ(NO)
                LOGREC(NO)
RECOVERY        NEPCLASS(0)
```

Figure 107. DFHCICSE profile definition

DFHCICSP

This is the default profile for the page retrieval transaction CSPG. You can specify a different profile for a particular transaction by means of the PROFILE option on the TRANSACTION definition.

The definition is as follows:

```
PROFILE(DFHCICSP)
GROUP(DFHSTAND) SCRNSIZE(DEFAULT)   PRINTERCOMP(NO)
UCTRAN(YES)
JOURNALLING     MSGJRNL(NO)
PROTECTION      MSGINTEG(NO)        ONEWTE(NO)   PROTECT(NO)
PROTOCOLS       DVSUPRT(ALL)        INBFMH(NO)   RAQ(NO)
LOGREC(NO)      RECOVERY            NEPCLASS(0)
```

Figure 108. DFHCICSP profile definition

DFHCICST

This is the default profile for principal facilities. You can specify a different profile for a particular transaction by means of the PROFILE option on the TRANSACTION definition.

The definition is as follows:

```
PROFILE(DFHCICST)
GROUP(DFHSTAND) SCRNSIZE(DEFAULT)   PRINTERCOMP(NO)
JOURNALLING     MSGJRNL(NO)
PROTECTION      MSGINTEG(NO)        ONEWTE(NO)   PROTECT(NO)
PROTOCOLS       DVSUPRT(ALL)        INBFMH(NO)   RAQ(NO)
                LOGREC(NO)
RECOVERY        NEPCLASS(0)
```

Figure 109. DFHCICST profile definition

DFHCICSV

This is the profile for principal facilities, when the transaction supports only VTAM devices.

The definition is as follows:

```
PROFILE(DFHCICSV)
GROUP(DFHSTAND) SCRNSIZE(DEFAULT)   PRINTERCOMP(NO)
JOURNALLING     MSGJRNL(NO)
PROTECTION      MSGINTEG(NO)        ONEWTE(NO)   PROTECT(NO)
PROTOCOLS       DVSUPRT(VTAM)       INBFMH(NO)   RAQ(NO)
                LOGREC(NO)
RECOVERY        NEPCLASS(0)
```

Figure 110. DFHCICSV profile definition

DFHPPF01 and DFHPPF02

Profiles DFHPPF01 and DFHPPF02 are used during CICS initialization, for tasks that are attached before the CSD file definitions have been installed.

The definitions are as follows:

```
PROFILE(DFHPPF01)  GROUP(DFHSTAND)  DESCRIPTION(VTAM-ONLY PROFILE)
SCRNSIZE(DEFAULT)  MSGJRNL(NO)      MSGINTEG(NO)
ONEWTE(NO)         PROTECT(NO)      DVSUPRT(VTAM)
INBFMH(NO)         RAQ(NO)          LOGREC(NO)
NEPCLASS(000)
PROFILE(DFHPPF02)  GROUP(DFHSTAND)  DESCRIPTION(ALL-NULLS PROFILE)
SCRNSIZE(DEFAULT)  MSGJRNL(NO)      MSGINTEG(NO)
ONEWTE(NO)         PROTECT(NO)      DVSUPRT(ALL)
INBFMH(NO)         RAQ(NO)          LOGREC(NO)
NEPCLASS(000)
```

Figure 111. DFHPPF01 and DFHPPF02 profile definitions

Model definitions in group DFHPGAIP

The CICS-supplied CSD group DFHPGAIP contains the following model definitions for use with autoinstall for programs.

DFHPGAPG

This is the default PROGRAM definition for program autoinstall.

The definition is as follows:

```
PROGRAM(DFHPGAPG)  GROUP(DFHPGAIP)
DESCRIPTION(default program for program autoinstall)
                   RELOAD(NO)           RESIDENT(NO)
USAGE(NORMAL)      USELPACOPY(NO)       STATUS(ENABLED)
RSL(00)            CEDF(YES)            DATALOCATION(BELOW)
EXECKEY(USER)      EXECUTIONSET(FULLAPI)
```

Figure 112. DFHPGAGP program definition

DFHPGAMP

This is the default MAPSET definition for program autoinstall.

The definition is as follows:

```
MAPSET(DFHPGAMP)  GROUP(DFHPGAIP)
DESCRIPTION(default mapset for program autoinstall)
RESIDENT(NO)       USAGE(NORMAL)
USELPACOPY(NO)     STATUS(ENABLED)
RSL(00)
```

Figure 113. DFHPGAMP mapset definition

DFHPGAPT

This is the default PARTITION definition for program autoinstall.

The definition is as follows:

Macro resource definition

```
PARTITIONSET(DFHPGAPT)  GROUP(DFHPGAIP)
DESCRIPTION(default partitionset for program autoinstall)
RESIDENT(NO)       USAGE(NORMAL)
USELPACOPY(NO)     STATUS(ENABLED)
RSL(00)
```

Figure 114. DFHPGAPT partitionset definition

Appendix C. Attribute cross-reference tables

This appendix contains two cross-reference tables relating macro operands (past and present) to RDO attributes.

Macro operand to RDO attribute

This list is in alphabetic order of macro operands, giving the equivalent RDO attribute. "RDO attribute to macro operand" on page 537 gives the same list in order of RDO attribute.

Macro operand	Resourcetype	RDO attribute
ACCMETH=INDIRECT	Connection	ACCESSMETHOD(INDIRECT)
ACCMETH=IRC	Connection	ACCESSMETHOD(IRC)
ACCMETH=VTAM	Connection	ACCESSMETHOD(VTAM)
ACCMETH=(IRC,XM)	Connection	ACCESSMETHOD(XM)
ALTPGE	Typeterm	ALTPAGE
ALTPRT(,COPY)	Terminal	ALTPRINTCOPY
ALTPRT(label,)	Terminal	ALTPRINTER
ALTSCRN	Typeterm	ALTSCREEN
ALTSFX	Typeterm	ALTSUFFIX
AUTH=(string)	See note 8	AUTHID(name)
AUTH=(GROUP)	See note 8	AUTHTYPE(GROUP)
AUTH=(SIGNID)	See note 8	AUTHTYPE(SIGN)
AUTH=(TERM)	See note 8	AUTHTYPE(TERM)
AUTH=(TXID)	See note 8	AUTHTYPE(TX)
AUTH=(USERID)	See note 8	AUTHTYPE(USERID)
AUTH=(USER)	See note 8	AUTHTYPE(OPID)
BASE	File	NSRGROUP
BLKSIZE	TD queue	BLOCKSIZE
BMSFEAT (see note 2)	Typeterm	ROUTEDMSGS(ALL)
BMSFEAT=FMHPARM	Typeterm	FMHPARM(YES)
BMSFEAT=NOROUTE	Typeterm	ROUTEDMSGS(NONE)
BMSFEAT=NOROUTEALL	Typeterm	ROUTEDMSGS(SPECIFIC)
BMSFEAT=OBFMT	Typeterm	OBFORMAT(YES)
BMSFEAT=OBOPID	Typeterm	OBOPERID(YES)
BRACKET	Typeterm	BRACKET
BUFFER	Typeterm	SENDSIZE
BUFFER	Sessions	SENDSIZE
BUFFERS=(1K(count))	Lsrpool	DATA1K
BUFFERS=(2K(count))	Lsrpool	DATA2K
BUFFERS=(4K(count))	Lsrpool	DATA4K
BUFFERS=(8K(count))	Lsrpool	DATA8K
BUFFERS=(12K(count))	Lsrpool	DATA12K
BUFFERS=(16K(count))	Lsrpool	DATA16K
BUFFERS=(20K(count))	Lsrpool	DATA20K
BUFFERS=(24K(count))	Lsrpool	DATA24K
BUFFERS=(28K(count))	Lsrpool	DATA28K
BUFFERS=(32K(count))	Lsrpool	DATA32K
BUFFERS=(512(count))	Lsrpool	DATA512
BUFFERS=(4K(,count))	Lsrpool	HSDATA4K
BUFFERS=(8K(,count))	Lsrpool	HSDATA8K
BUFFERS=(12K(,count))	Lsrpool	HSDATA12K

Macro to RDO

	Macro operand	Resourcetype	RDO attribute
	BUFFERS=(16K(,count))	Lsrpool	HSDATA16K
	BUFFERS=(20K(,count))	Lsrpool	HSDATA20K
	BUFFERS=(24K(,count))	Lsrpool	HSDATA24K
	BUFFERS=(28K(,count))	Lsrpool	HSDATA28K
	BUFFERS=(32K(,count))	Lsrpool	HSDATA32K
	BUFND	File	DATABUFFERS
	BUFNI	File	INDEXBUFFERS
	BUFNO	TD queue	DATABUFFERS
	CHNASSY	Sessions	BUILDCHAIN
	CHNASSY	Typeterm	BUILDCHAIN
	CONNECT	Typeterm	AUTOCONNECT
	CONNECT	Connection	AUTOCONNECT
	CONNECT	Sessions	AUTOCONNECT
	CONSLID	Terminal	CONSOLE
|	DATAID	Tsmodel	PREFIX
	DATASTR	Connection	DATASTREAM
	DEFSCRN	Typeterm	DEFSCREEN
	DESTFAC	TD queue	ATIFACILITY
	DESTID	TD queue	TDQUEUE
	DESTRCV	TD queue	RECOVSTATUS
	DISP	File	DISPOSITION
	DPMODE	DB2conn	PRIORITY (see note 9)
	DPMODI	See note 10	
	DSCNAME	TD queue	DDNAME
	DSNAME	File	DSNAME
	DSNSHR	File	DSNSHARING
	ERRATT=BLINK	Typeterm	ERRHILIGHT(BLINK)
	ERRATT=color	Typeterm	ERRCOLOR
	ERRATT=INTENSIFY	Typeterm	ERRINTENSIFY
	ERRATT=LASTLINE	Typeterm	ERRLASTLINE
	ERRATT=REVERSE	Typeterm	ERRHILIGHT(REVERSE)
	ERRATT=UNDERLINE	Typeterm	ERRHILIGHT(UNDERLINE)
	ERRDEST=(dest1)	DB2conn	MSGQUEUE1(name)
	ERRDEST=(dest2)	DB2conn	MSGQUEUE2(name)
	ERRDEST=(dest3)	DB2conn	MSGQUEUE3(name)
	ERROPT	TD queue	ERROROPTION
#	FEATURE=3720E	Typeterm	See Note 1
	FEATURE=APLKYBD	Typeterm	APLKYBD
	FEATURE=APLTEXT	Typeterm	APLTEXT
#	FEATURE=3270E	Typeterm	See Note 1
	FEATURE=ASCII-7	Typeterm	ASCII(7)
	FEATURE=ASCII-8	Typeterm	ASCII(8)
	FEATURE=AUDALARM	Typeterm	AUDIBLEALARM
	FEATURE=BTRANS	Typeterm	BACKTRANS
	FEATURE=COLOR	Typeterm	COLOR
	FEATURE=COPY	Typeterm	COPY
	FEATURE=DCKYBD	Typeterm	DUALCASEKYBD
	FEATURE=EXTDS	Typeterm	EXTENDEDDS
	FEATURE=HILIGHT	Typeterm	HILIGHT
	FEATURE=KATAKANA	Typeterm	KATAKANA
	FEATURE=MSRCNTRL	Typeterm	MSRCONTROL
	FEATURE=OUTLINE	Typeterm	OUTLINE
	FEATURE=PARALLEL	Connection	SINGLESESS(NO)
	FEATURE=PARTNS	Typeterm	PARTITIONS

Macro operand	Resourcetype	RDO attribute
FEATURE=PTRADAPT	Typeterm	PRINTADAPTER
FEATURE=PS	Typeterm	PROGSYMBOLS
FEATURE=QUERYALL	Typeterm	QUERY(ALL)
FEATURE=QUERYCOLD	Typeterm	QUERY(COLD)
FEATURE=SELCTPEN	Typeterm	LIGHTPEN
FEATURE=SINGLE	Connection	SINGLESESS(YES)
FEATURE=SOSI	Typeterm	SOSI
FEATURE=TEXTKYBD	Typeterm	TEXTKYBD
FEATURE=TEXTPRINT	Typeterm	TEXTPRINT
FEATURE=UCTRAN	Typeterm	UCTRAN
FEATURE=VALIDATION	Typeterm	VALIDATION
FF	Typeterm	FORMFEED
FILE	File	FILE
FILSTAT=ENABLED|DISABLED	File	STATUS
FILSTAT=OPENED|CLOSED	File	OPENTIME
GMMSG	Typeterm	LOGONMSG
HF	Typeterm	HORIZFORM
INDEST	TD queue	INDIRECTNAME
INDSYS	Connection	INDSYS
JID	File	FWDRECOVLOG
JID=SYSTEM	File	JOURNAL(1)
JREQ=WN	File	JNLADD
JREQ=RU	File	JNLREAD(UPDATEONLY)
JREQ=RO	File	JNLREAD(READONLY)
JREQ=SYN	File	JNLSYNCREAD
JREQ=ASY	File	JNLSYNCWRITE(NO)
JREQ=WU	File	JNLUPDATE
KEYLEN	File	KEYLENGTH
KEYLEN	Lsrpool	MAXKEYLENGTH
LDC	Typeterm	LDCLIST
LENGTH	TD queue	RECORDSIZE
LOG=YES	File	RECOVERY
LOGMODE	Typeterm	LOGMODE
LRECL	File	RECORDSIZE
LSRPOOL	File	See note 5
MAXSESS	Sessions	MAXIMUM
MODENAM	Sessions	MODENAME
MODENAM	Terminal	MODENAME
NATLANG	Terminal	NATLANG
NEPCLAS	Typeterm	NEPCLASS
NETNAME	Connection	NETNAME
NETNAME	Terminal	NETNAME
NETNAMQ	Sessions	NETNAMEQ
OPEN	TD queue	OPENTIME
PASSWD	File	PASSWORD
PCTEROP=AEY9	DB2conn	THREADERROR(ABEND)
PCTEROP=N906	DB2conn	THREADERROR(N906)
PCTEROP=N906D	DB2conn	THREADERROR(N906D)
PGESIZE	Typeterm	PAGESIZE
PGESTAT=AUTOPAGE	Typeterm	AUTOPAGE(YES)
PGESTAT=PAGE	Typeterm	AUTOPAGE(NO)
PIPELN	Terminal	POOL
PLAN	DB2conn	PLAN (see note 9)
PLANI	See note 10	

Macro to RDO

Macro operand	Resourcetype	RDO attribute
PLNEXIT	See note 1	
PLNPGME	DB2conn	PLANEXITNAME (see note 9)
PLNPGMI	See note 10	
PLNXTR1	See note 1	
PLNXTR2	See note 1	
POOL see note 12	Tsmodel	POOLNAME
PRINTTO(,COPY)	Terminal	PRINTERCOPY
PRINTTO(label,)	Terminal	PRINTER
PURGEC	DB2conn	PURGECYCLE
RECEIVE=(,n)	Sessions	RECEIVECOUNT
RECEIVE=(x,)	Sessions	RECEIVEPFX
RECFM	Connection	RECORDFORMAT
RECFORM=U|V|F)	File	RECORDFORMAT
RECFORM	TD queue	See note 7
RECSIZE	TD queue	RECORDSIZE
RELREQ=(,NO|YES)	Sessions	DISCREQ
RELREQ=(NO|YES,)	Sessions	RELREQ
RELREQ=(,NO|YES)	Typeterm	DISCREQ
RELREQ=(NO|YES,)	Typeterm	RELREQ
REWIND	TD queue	REWIND
RMTNAME	File	REMOTENAME
RMTNAME	TD queue	REMOTENAME
RMTNAME	Terminal	REMOTENAME
RMTNAME	Tsmodel	REMOTEPREFIX
RNOTIFY	Typeterm	RECOVNOTIFY
ROLBE	DB2conn	DROLLBACK (see note 9)
ROLBI	See note 10	
ROPTION	Typeterm	RECOVOPTION
RSCLMT	Lsrpool	SHARELIMIT
RUSIZE	Sessions	RECEIVESIZE
RUSIZE	Typeterm	RECEIVESIZE
SEND=(,n)	Sessions	SENDCOUNT
SEND=(x,)	Sessions	SENDPFX
SERVREQ=ADD	File	ADD
SERVREQ=BROWSE	File	BROWSE
SERVREQ=DELETE	File	DELETE
SERVREQ=READ	File	READ
SERVREQ=UPDATE	File	UPDATE
SESTYPE	Typeterm	SESSIONTYPE (see note 13)
SHDDEST	DB2CONN	STATSQUEUE
SIGNID	DB2conn	SIGNID
SIGNOFF	Typeterm	SIGNOFF
SIZE=number	File	MAXNUMRECS
SNAP	See note 1	
STANDBY=ABEND	DB2conn	CONNECTERROR(ABEND)
STANDBY=SQLCODE	DB2conn	CONNECTERROR(SQLCODE)
STRTWT=YES	DB2conn	STANDBYMODE(CONNECT)
STRTWT=NO	DB2conn	STANDBYMODE(NOCONNECT)
STRTWT=AUTO	DB2conn	STANDBYMODE(RECONNECT)
SUBID=name	DB2conn	DB2ID(name)
SUFFIX	See note 1	
STRNO	File	STRINGS
STRNO	Lsrpool	STRINGS

Macro operand	Resourcetype	RDO attribute
SYSIDNT	Connection	CONNECTION
SYSIDNT	File	REMOTESYSTEM
SYSIDNT	Sessions	CONNECTION
SYSIDNT	TD queue	REMOTESYSTEM
SYSIDNT	Terminal	REMOTESYSTEM
SYSIDNT	Tsmodel	REMOTESYSTEM
TASKNO	Terminal	TASKLIMIT
TASKREQ	(DSNCRCT macro)	See note 1
TCTUAL	Sessions	USERAREALEN
TCTUAL	Typeterm	USERAREALEN
THRDA	See note 8	THREADLIMIT
THRDM	See note 1	
THRDMAX	DB2conn	TCBLIMIT
THRDS	DB2entry	PROTECTNUM
TIOAL	Sessions	IOAREALEN
TIOAL	Typeterm	IOAREALEN
TOKENE	DB2conn	ACCOUNTREC (see note 9)
TOKENI	See note 10	
TRACEID	See note 1	
TRANSID	TD queue	TRANSID
TRANSID	Terminal	TRANSACTION
TRIGLEV	TD queue	TRIGGERLEVEL
TRMIDNT	Sessions	SESSNAME
TRMIDNT	Terminal	TERMINAL
TRMMODL	Typeterm	TERMMODEL
TRMPRTY	Sessions	SESSPRIORITY
TRMPRTY	Terminal	TERMPRIORITY
TRMSTAT (see note 3)	Typeterm	ATI(NO) and TTI(NO)
TRMSTAT (see note 4)	Typeterm	CREATESESS(YES)
TRMSTAT=INPUT	See note 1	
TRMSTAT=INTLOG	Typeterm	CREATESESS(YES)
TRMSTAT=NOINTLOG	Typeterm	CREATESESS(NO)
TRMSTAT='OUT OF SERVICE'	Connection	INSERVICE(No)
TRMSTAT='OUT OF SERVICE'	Sessions	INSERVICE(No)
TRMSTAT='OUT OF SERVICE'	Terminal	INSERVICE(No)
TRMSTAT=RECEIVE	Typeterm	ATI(YES) and TTI(NO)
TRMSTAT=TRANSACTION	Typeterm	ATI(NO) and TTI(YES)
TRMSTAT=TRANSCEIVE	Typeterm	ATI(YES) and TTI(YES)
TRMTYPE	Connection	PROTOCOL
TRMTYPE	Sessions	PROTOCOL
TRMTYPE	Typeterm	DEVICE
TWAIT	DB2conn	THREADWAIT (see note 9)
TWAITI	See note 10	
TXID	DB2entry	TRANSID (see note 11)
TXIDSO	DB2conn	ACCOUNTREC (see note 9)
TYPE=CICSTABLE|USERTABLE	File	TABLE
TYPE=EXTRA	TD queue	TYPE(EXTRA)
TYPE=INDIRECT	TD queue	TYPE(INDIRECT)
TYPE=INTRA	TD queue	TYPE(INTRA)
TYPE=RECOVERY	Tsmodel	RECOVERY(Yes)
TYPE=REMOTE	TD queue	See note 6
TYPE=SDSCI	TD queue	See note 1

Macro operand	Resourcetype	RDO attribute
TYPE=SECURITY	Tsmodel	SECURITY(Yes)
TYPEFLE	TD queue	TYPEFILE
USERID	Sessions	USERID
USERID	TD queue	USERID
USERID	Terminal	USERID
USERSEC	Connection	ATTACHSEC
USERSEC	Terminal	ATTACHSEC
VF	Typeterm	VERTICALFORM
XSNAME	Connection	SECURITYNAME
XSNAME	Terminal	SECURITYNAME

Notes:

1. No RDO equivalent is provided.
2. The RDO equivalent of not specifying either NOROUTE or NOROUTEALL for BMSFEAT is ROUTEDMSGS(ALL).
3. The RDO equivalent of not specifying any of TRANCEIVE, RECEIVE, or TRANSACTION is ATI(NO), TTI(NO).
4. The RDO equivalent of not specifying either INTLOG or NOINTLOG for TRMSTAT is CREATESESS(YES).
5. The RDO equivalent of the macro attribute LSRPOOL is LSRPOOLID on CEDA DEFINE LSRPOOL or on CEDA DEFINE FILE.
6. You can find out the RDO equivalent of this macro operand by omitting the queue type.
7. In the DFHDCT macro, this value of the RECFORM attribute contains several pieces of information. The RDO attributes RECORDFORMAT, BLOCKFORMAT, and PRINTCONTROL supply these pieces of information.
8. The AUTH macro parameters defined on the TYPE=POOL and TYPE=ENTRY macros correspond to the AUTHID or AUTHTYPE RDO parameters as shown, and can be specified on either a DB2CONN or a DB2ENTRY resource definition, as appropriate. The AUTH parameters on the TYPE=COMD macro, however, correspond to the COMAUTHID or COMAUTHTYPE on a DB2CONN resource definition.

   Similarly, the THRDA macro parameters defined on the TYPE=POOL and TYPE=ENTRY macros correspond to the THREADLIMIT RDO parameter as shown, and can be specified on either a DB2CONN or a DB2ENTRY resource definition, as appropriate. The THRDA parameter on the TYPE=COMD macro, however, corresponds to the COMTHREADLIM parameter on a DB2CONN resource definition.
9. These macro parameters can also be specified on a DB2ENTRY resource definition.
10. There is no equivalent for these RCT parameters, which allow you to define defaults for other parameters (for example, TOKENI in the TYPE=INIT macro specifies a default if TOKENE is not defined on the TYPE=ENTRY macro. To override RDO system defaults, use the USERDEFINE method described on pages 67 (for CEDA) and 113 (for DFHCSDUP)
11. This parameter can also be defined as the RDO TRANSID parameter on the DB2TRAN resource definition.
12. The POOL parameter in a TYPE=SHARED macro instruction will map to the RDO POOLNAME attribute, ONLY if there is a TYPE=REMOTE macro instruction with the same SYSIDNT in the TST that is being migrated.
13. Not relevant for LU6.1

Note that there are no DFHDCT macro equivalents for the following attributes on the TDQUEUE RDO definition:
- BLOCKFORMAT
- DISPOSITION
- DSNAME
- PRINTCONTROL
- SYSOUT
- WAIT
- WAITACTION

To use these attributes, you must use RDO for transient data queues.

RDO attribute to macro operand

This list is in alphabetic order of RDO attribute, giving the equivalent macro operand. “Macro operand to RDO attribute” on page 531 gives the same list in order of macro operand.

RDO attribute	Resourcetype	Macro operand
ACCESSMETHOD	Connection	ACCMETH
ACCOUNTREC (see note 6)	DB2conn	TXIDSO
ACCOUNTREC (see note 6)	DB2conn	TOKENE
ADD	File	SERVREQ=ADD
ALTPAGE	Typeterm	ALTPGE
ALTPRINTCOPY	Terminal	ALTPRT(,COPY)
ALTPRINTER	Terminal	ALTPRT(label,)
ALTSCREEN	Typeterm	ALTSCRN
ALTSUFFIX	Typeterm	ALTSFX
APLKYBD	Typeterm	FEATURE=APLKYBD
APLTEXT	Typeterm	FEATURE=APLTEXT
ASCII(7)	Typeterm	FEATURE=ASCII-7
ASCII(8)	Typeterm	FEATURE=ASCII-8
ATIFACILITY	TD queue	DESTFAC
ATI(NO) and TTI(YES)	Typeterm	TRMSTAT=TRANSACTION
ATI(YES) and TTI(NO)	Typeterm	TRMSTAT=RECEIVE
ATI(YES) and TTI(YES)	Typeterm	TRMSTAT=TRANSCEIVE
ATTACHSEC	Connection	USERSEC=IDENTIFY
ATTACHSEC	Connection	USERSEC=LOCAL
ATTACHSEC	Connection	USERSEC=MIXIDPE
ATTACHSEC	Connection	USERSEC=PERSISTENT
ATTACHSEC	Connection	USERSEC=VERIFY
ATTACHSEC	Terminal	USERSEC=IDENTIFY
ATTACHSEC	Terminal	USERSEC=LOCAL
ATTACHSEC	Terminal	USERSEC=MIXIDPE
ATTACHSEC	Terminal	USERSEC=PERSISTENT
ATTACHSEC	Terminal	USERSEC=VERIFY
AUDIBLEALARM	Typeterm	FEATURE=AUDALARM
AUTHID(name)	See note 5	AUTH=(string)
AUTHTYPE(GROUP)	See note 5	AUTH=(GROUP)
AUTHTYPE(SIGN)	See note 5	AUTH=(SIGNID)
AUTHTYPE(TERM)	See note 5	AUTH=(TERM)

RDO to Macro

RDO attribute	Resourcetype	Macro operand
AUTHTYPE(TX)	See note 5	AUTH=(TXID)
AUTHTYPE(USERID)	See note 5	AUTH=(USERID)
AUTHTYPE(OPID)	See note 5	AUTH=(USER)
AUTINSTMODEL	Terminal	See note 1
AUTINSTNAME	Terminal	See note 1
AUTOCONNECT	Connection	CONNECT
AUTOCONNECT	Sessions	CONNECT
AUTOCONNECT	Typeterm	CONNECT
AUTOPAGE(NO)	Typeterm	PGESTAT=PAGE
AUTOPAGE(YES)	Typeterm	PGESTAT=AUTOPAGE
BACKTRANS	Typeterm	FEATURE=BTRANS
BACKUPTYPE	File	See note 1
BINDPASSWORD	Connection	BINDPWD
BINDPASSWORD	Terminal	BINDPWD
BLOCKFORMAT	TD queue	See note 1
BLOCKSIZE	TD queue	BLKSIZE
BRACKET	Typeterm	BRACKET
BROWSE	File	SERVREQ=BROWSE
BUILDCHAIN	Sessions	CHNASSY=YES
BUILDCHAIN	Typeterm	CHNASSY
CGCSGID	Typeterm	See note 1
COLOR	Typeterm	FEATURE=COLOR
CONNECTERROR(ABEND)	DB2conn	STANDBY=ABEND
CONNECTERROR(SQLCODE)	DB2conn	STANDBY=SQLCODE
CONNECTION	Connection	SYSIDNT
CONNECTION	Sessions	SYSIDNT
CONNTYPE	Connection	See note 1
CONSOLE	Terminal	CONSLID
COPY	Typeterm	FEATURE=COPY
CREATESESS(NO)	Typeterm	TRMSTAT=NOINTLOG
CREATESESS(YES)	Typeterm	See note 2
DATABUFFERS	File	BUFND
DATABUFFERS	TD queue	BUFNO
DATASTREAM	Connection	DATASTR
DATA1K	Lsrpool	BUFFERS=(1K(count))
DATA2K	Lsrpool	BUFFERS=(2K(count))
DATA4K	Lsrpool	BUFFERS=(4K(count))
DATA8K	Lsrpool	BUFFERS=(8K(count))
DATA12K	Lsrpool	BUFFERS=(12K(count))
DATA16K	Lsrpool	BUFFERS=(16K(count))
DATA20K	Lsrpool	BUFFERS=(20K(count))
DATA24K	Lsrpool	BUFFERS=(24K(count))
DATA28K	Lsrpool	BUFFERS=(28K(count))
DATA32K	Lsrpool	BUFFERS=(32K(count))
DATA512	Lsrpool	BUFFERS=(512(count))
DB2ID(name)	DB2conn	SUBID=name
DDNAME	TD queue	DSCNAME
DEFSCREEN	Typeterm	DEFSCRN
DELETE	File	SERVREQ=DELETE
DEVICE	Typeterm	TRMTYPE
DISCREQ	Sessions	RELREQ=(,NO|YES)
DISCREQ	Typeterm	RELREQ=(,NO|YES)

RDO attribute	Resourcetype	Macro operand
DISPOSITION	File	DISP
DISPOSITION	TD queue	See note 1
DROLLBACK (see note 6)	DB2conn	ROLBE
DSNAME	File	DSNAME
DSNAME	TD queue	See note 1
DSNSHARING	File	DSNSHR
DUALCASEKYBD	Typeterm	FEATURE=DCKYBD
ERRCOLOR	Typeterm	ERRATT=color
ERRHILIGHT(BLINK)	Typeterm	ERRATT=BLINK
ERRHILIGHT(REVERSE)	Typeterm	ERRATT=REVERSE
ERRHILIGHT(UNDERLINE)	Typeterm	ERRATT=UNDERLINE
ERRINTENSIFY	Typeterm	ERRATT=INTENSIFY
ERRLASTLINE	Typeterm	ERRATT=LASTLINE
ERROROPTION	TD queue	ERROPT
EXTENDEDDS	Typeterm	FEATURE=EXTDS
FILE	File	FILE
FMHPARM(YES)	Typeterm	BMSFEAT=FMHPARM
FORMFEED	Typeterm	FF
FWDRECOVLOG	File	JID
GROUP	Terminal	See note 1
GROUP	Tsmodel	See note 1
GROUP	Typeterm	See note 1
HILIGHT	Typeterm	FEATURE=HILIGHT
HORIZFORM	Typeterm	HF
HSDATA4K	Lsrpool	BUFFERS=(4K(,count))
HSDATA8K	Lsrpool	BUFFERS=(8K(,count))
HSDATA12K	Lsrpool	BUFFERS=(12K(,count))
HSDATA16K	Lsrpool	BUFFERS=(16K(,count))
HSDATA20K	Lsrpool	BUFFERS=(20K(,count))
HSDATA24K	Lsrpool	BUFFERS=(24K(,count))
HSDATA28K	Lsrpool	BUFFERS=(28K(,count))
HSDATA32K	Lsrpool	BUFFERS=(32K(,count))
HSINDEX4K	Lsrpool	See note 1
HSINDEX8K	Lsrpool	See note 1
HSINDEX12K	Lsrpool	See note 1
HSINDEX16K	Lsrpool	See note 1
HSINDEX20K	Lsrpool	See note 1
HSINDEX24K	Lsrpool	See note 1
HSINDEX28K	Lsrpool	See note 1
HSINDEX32K	Lsrpool	See note 1
INDEXBUFFERS	File	BUFNI
INDIRECTNAME	TD queue	INDEST
INDSYS	Connection	INDSYS
INSERVICE (No)	Connection	TRMSTAT='OUT OF SERVICE'
INSERVICE (No)	Sessions	TRMSTAT='OUT OF SERVICE'
INSERVICE (No)	Terminal	TRMSTAT='OUT OF SERVICE'
IOAREALEN	Sessions	TIOAL
IOAREALEN	Typeterm	TIOAL
JNLADD(BEFORE)	File	JREQ=WN
JNLREAD(UPDATEONLY)	File	JREQ=RU
JNLREAD(READONLY)	File	JREQ=RO
JNLSYNCREAD	File	JREQ=SYN
JNLSYNCWRITE(NO)	File	JREQ=ASY
JNLUPDATE	File	JREQ=WU

RDO attribute	Resourcetype	Macro operand
JOURNAL	File	JID
KATAKANA	Typeterm	FEATURE=KATAKANA
KEYLENGTH	File	KEYLEN
LDCLIST	Typeterm	LDC
LIGHTPEN	Typeterm	FEATURE=SELCTPEN
LOCATION(Auxiliary)	Tsmodel	See note 1
LOCATION(Main)	Tsmodel	See note 1
LOGMODE	Typeterm	LOGMODE
LOGONMSG	Typeterm	GMMSG
LSRPOOL	Lsrpool	See note 1
LSRPOOLID	File	LSRPOOL
LSRPOOLID	Lsrpool	See note 4
MAXIMUM	Sessions	MAXSESS
MAXKEYLENGTH	Lsrpool	KEYLEN
MAXNUMRECS	File	SIZE
MAXQTIME	Connection	See note 1
MODENAME	Sessions	MODENAM
MODENAME	Terminal	MODENAM
MSGQUEUE1(name)	DB2conn	ERRDEST=(dest1)
MSGQUEUE2(name)	DB2conn	ERRDEST=(dest2)
MSGQUEUE3(name)	DB2conn	ERRDEST=(dest3)
MSRCONTROL	Typeterm	FEATURE=MSRCNTRL
NATLANG	Terminal	NATLANG
NEPCLASS	Sessions	See note 1
NEPCLASS	Typeterm	NEPCLAS
NETNAME	Connection	NETNAME
NETNAME	Terminal	NETNAME
NETNAMEQ	Sessions	NETNAMQ
NSRGROUP	File	BASE
OBFORMAT(YES)	Typeterm	BMSFEAT=OBFMT
OBOPERID(YES)	Typeterm	BMSFEAT=OBOPID
OPENTIME	File	FILSTAT=OPEN|CLOSED
OPENTIME	TD queue	OPEN
PASSWORD	File	PASSWD
PLAN (see note 6)	DB2conn	PLAN
PLANEXITNAME (see note 6)	DB2conn	PLNPGME
POOL	Terminal	PIPELN
POOLNAME	Tsmodel	POOL see note 8
PREFIX	Tsmodel	DATAID
PRINTADAPTER	Typeterm	FEATURE=PTRADAPT
PRINTCONTROL	TD queue	See note 1
PRINTER	Terminal	PRINTTO(label,)
PRINTERCOPY	Terminal	PRINTTO(,COPY)
PRIORITY (see note 6)	DB2conn	DPMODE
PROGSYMBOLS	Typeterm	FEATURE=PS
PROTECTNUM	DB2entry	THRDS
PROTOCOL	Connection	TRMTYPE
PROTOCOL	Sessions	TRMTYPE
PSRECOVERY	Connection	See note 1
PURGECYCLE	DB2conn	PURGEC
QUERY(ALL)	Typeterm	FEATURE=QUERYALL
QUERY(COLD)	Typeterm	FEATURE=QUERYCOLD
QUEUELIMIT	Connection	See note 1
READ	File	SERVREQ=READ

	RDO attribute	Resourcetype	Macro operand
	RECEIVECOUNT	Sessions	RECEIVE=(,n)
	RECEIVEPFX	Sessions	RECEIVE=(x,)
	RECEIVESIZE	Sessions	RUSIZE
	RECEIVESIZE	Typeterm	RUSIZE
	RECORDFORMAT	Connection	RECFM
	RECORDFORMAT	File	RECFORM=(U|V|F)
	RECORDSIZE	File	LRECL
	RECORDSIZE	TD queue	LENGTH
	RECORDSIZE	TD queue	RECSIZE
	RECOVERY	File	LOG=YES
|	RECOVERY(No)	Tsmodel	See note 1
|	RECOVERY(Yes)	Tsmodel	TYPE=RECOVERY
	RECOVNOTIFY	Typeterm	RNOTIFY
	RECOVOPTION	Sessions	ROPTION
	RECOVOPTION	Typeterm	ROPTION
	RECOVSTATUS	TD queue	DESTRCV
	RELREQ	Sessions	RELREQ=(NO|YES,)
	RELREQ	Typeterm	RELREQ=(NO|YES,)
	REMOTENAME	Connection	RMTNAME
	REMOTENAME	File	RMTNAME
	REMOTENAME	TD queue	RMTNAME
	REMOTENAME	Terminal	RMTNAME
|	REMOTEPREFIX	Tsmodel	RMTNAME
	REMOTESYSNET	Connection	See note 1
	REMOTESYSTEM	Connection	SYSIDNT
	REMOTESYSTEM	File	SYSIDNT
	REMOTESYSTEM	TD queue	SYSIDNT
	REMOTESYSTEM	Terminal	SYSIDNT
|	REMOTESYSTEM	Tsmodel	SYSIDNT
	REWIND	TD queue	REWIND
	ROUTEDMSGS(NONE)	Typeterm	BMSFEAT=NOROUTE
	ROUTEDMSGS(SPECIFIC)	Typeterm	BMSFEAT=NOROUTEALL
	ROUTEDMSGS(ALL)	Typeterm	See note 3
|	SECURITY(No)	Tsmodel	See note 1
|	SECURITY(Yes)	Tsmodel	TYPE=SECURITY
	SECURITYNAME	Connection	XSNAME
	SECURITYNAME	Terminal	XSNAME
	SENDCOUNT	Sessions	SEND=(,n)
	SENDPFX	Sessions	SEND=(x,)
	SENDSIZE	Sessions	BUFFER
	SENDSIZE	Typeterm	BUFFER
	SESSIONS	Sessions	See note 1
	SESSIONTYPE (see note 9)	Typeterm	SESTYPE
	SESSNAME	Sessions	TRMIDNT
	SESSPRIORITY	Sessions	TRMPRTY
	SHARELIMIT	Lsrpool	RSCLMT
	SHIPPABLE	Typeterm	See note 1
	SIGNID	DB2conn	SIGNID
	SIGNOFF	Typeterm	SIGNOFF
	SINGLESESS	Connection	FEATURE=SINGLE
	STANDBYMODE(CONNECT)	DB2conn	STRTWT=YES
	STANDBYMODE(NOCONNECT)	DB2conn	STRTWT=NO
	STANDBYMODE(RECONNECT)	DB2conn	STRTWT=AUTO
	SOSI	Typeterm	FEATURE=SOSI

RDO attribute	Resourcetype	Macro operand
STATSQUEUE	DB2CONN	SHDDEST
STATUS	File	FILSTAT=ENABLED|DISABLED
STRINGS	File	STRNO
STRINGS	Lsrpool	STRNO
SYSOUT	TD queue	See note 1
TABLE	File	TYPE=CICSTABLE|USERTABLE
TASKDATAKEY	Connection	See note 1
TASKLIMIT	Terminal	TASKNO
TCBLIMIT	DB2conn	THRDMAX
TDQUEUE	TD queue	DESTID
TERMINAL	Terminal	TRMIDNT
TERMMODEL	Typeterm	TRMMODL
TERMPRIORITY	Terminal	TRMPRTY
TEXTKYBD	Typeterm	FEATURE=TEXTKYBD
TEXTPRINT	Typeterm	FEATURE=TEXTPRINT
THREADERROR(ABEND)	DB2conn	PCTEROP=AEY9
THREADERROR(N906)	DB2conn	PCTEROP=N906
THREADERROR(N906D)	DB2conn	PCTEROP=N906D
THREADLIMIT	THRDA	See note 5
THREADWAIT (see note 6)	DB2conn	TWAIT
TRANSACTION	Terminal	TRANSID
TRANSID	TD queue	TRANSID
TRANSID (see note 7)	DB2entry	TXID
TRIGGERLEVEL	TD queue	TRIGLEV
TTI(NO) and ATI(YES)	Typeterm	TRMSTAT=RECEIVE
TTI(YES) and ATI(NO)	Typeterm	TRMSTAT=TRANSACTION
TTI(YES) and ATI(YES)	Typeterm	TRMSTAT=TRANCEIVE
TYPEFILE	TD queue	TYPEFLE
TYPETERM	Terminal	See note 1
TYPETERM	Typeterm	See note 1
TYPE(EXTRA)	TD queue	TYPE=EXTRA
TYPE(INDIRECT)	TD queue	TYPE=INDIRECT
TYPE(INTRA)	TD queue	TYPE=INTRA
UCTRAN	Typeterm	FEATURE=UCTRAN
UPDATE	File	SERVREQ=UPDATE
USERAREALEN	Sessions	TCTUAL
USERAREALEN	Typeterm	TCTUAL
USERID	Sessions	USERID
USERID	TD queue	USERID
USERID	Terminal	USERID
WAIT	TD queue	See note 1
WAITACTION	TD queue	See note 1
VALIDATION	Typeterm	FEATURE=VALIDATION
VERTICALFORM	Typeterm	VF
XRFSIGNOFF	Typeterm	See note 1

Notes:

1. There is no macro equivalent.
2. CREATESESS(YES) is equivalent to specifying neither INTLOG nor NOINTLOG for TRMSTAT.
3. ROUTEDMSGS(ALL) is equivalent to specifying neither NOROUTE nor NOROUTEALL for BMSFEAT.

4. The macro equivalent of the LSRPOOLID attribute on CEDA DEFINE LSRPOOL is LSRPOOL on the DFHFCT macro.
5. The AUTH macro parameters defined on the TYPE=POOL and TYPE=ENTRY macros correspond to the AUTHID or AUTHTYPE RDO parameters as shown, and can be specified on either a DB2CONN or a DB2ENTRY resource definition, as appropriate. The AUTH parameters on the TYPE=COMD macro, however, correspond to the COMAUTHID or COMAUTHTYPE on a DB2CONN resource definition.

   Similarly, the THRDA macro parameters defined on the TYPE=POOL and TYPE=ENTRY macros correspond to the THREADLIMIT RDO parameter as shown, and can be specified on either a DB2CONN or a DB2ENTRY resource definition, as appropriate. The THRDA parameter on the TYPE=COMD macro, however, corresponds to the COMTHREADLIM parameter on a DB2CONN resource definition.
6. These macro parameters can also be specified on a DB2ENTRY resource definition.
7. This parameter can also be defined as the RDO TRANSID parameter on the DB2TRAN resource definition.
8. The POOL parameter in a TYPE=SHARED macro instruction will map to the RDO POOLNAME attribute, ONLY if there is a TYPE=REMOTE macro instruction with the same SYSIDNT in the TST that is being migrated.
9. Not relevant for LU6.1

Note that the RDO equivalent of the DFHDCT TYPE=REMOTE macro can be found by leaving the TYPE attribute blank.

Appendix D. Migrating the TCT to the CSD file

This appendix describes the process of converting terminal and system definition macros for VTAM devices into their RDO equivalents. Before you read further, note that you **must** use RDO for the TCT if you use VTAM as your telecommunication access method for at least some of your resources, because VTAM TCTs are no longer supported online (other than LDCs).

Note also that there is a special facility, called **autoinstall**, that can remove the need for some migration. You may decide that you don't need to migrate all of your DFHTCT macros, because you are going to use autoinstall. It would be wise, nevertheless, to read the following section and maybe perform the migration process anyway. This allows you to study the RDO definitions that the utility produces, and could help you decide how many different **autoinstall models** you need. For more information, see "Chapter 8. Autoinstall for VTAM terminals" on page 121.

This appendix is divided into three sections:
1. The steps involved in the migration process.
2. The steps involved in migrating remote terminal definitions, depending on the method of defining remote terminals you choose.
3. Some notes on migrating different types of macro, including links and sessions with other systems.

The needs of your installation may introduce various complications into the migration process:
- You may wish to introduce autoinstall in stages, or for some terminals but not all
- If you have remote terminals for transaction routing, there are three approaches you could adopt toward migrating your terminal definitions, as described in "Migrating remote terminal definitions" on page 550.

Whatever your choices, you finish up with a TCT built from one or more of the following:
- Entries built automatically through autoinstall, which are dynamic and are present in the TCT until autoinstalled
- Entries defined and installed using RDO and the GRPLIST system initialization parameter
- Entries for non-VTAM terminals defined using DFHTCT macros

Autoinstalled TCT entries never override any others. If a table entry already exists for a given terminal, CICS does not try to autoinstall it. (For more information see "Chapter 8. Autoinstall for VTAM terminals" on page 121.)

Summary of implementation

To migrate to RDO, do the following:
1. Edit your existing DFHTCT source.
2. Assemble and link-edit the edited source.

3. Process the link-edited table using the MIGRATE command of the DFHCSDUP utility.
4. Check the output produced by the MIGRATE command.
5. Use the DFHCSDUP ADD command to create a list of groups on your CSD file. Name this list in your SIT for initialization.
6. Modify your macro source so that it no longer generates table entries that have been migrated, then reassemble it to produce the non-VTAM part of your TCT (if any).
7. Start up your system with the non-VTAM TCT (if any) and the list you have created. Use one of the terminals whose definition was included in the list to further modify your list and, perhaps, to regroup and rename migrated definitions.

The rest of this section of the appendix discusses these steps in more detail.

Editing your existing DFHTCT source

The first step in migrating your DFHTCT macros to the CSD file is to assemble the source. If your TCT is large, you might find it most convenient to assemble small segments of the source individually.

Each resource definition on the CSD file must belong to a CSD group. DFHCSDUP MIGRATE places all definitions in groups as it processes them.

You probably already have good reasons for grouping resources. For example, you might like to group TERMINAL definitions according to the physical location of the terminals, or according to the department or function of the people using them.

You can specify the names to be given to groups of definitions generated from your DFHTCT macros. You do this by adding special macro instructions to the table source. The rules for group names are given on page 303. The form of each macro instruction is:

```
DFHTCT TYPE=GROUP,GROUP=xxxxxxxx           (up to 8 characters)
```

All definitions following a particular TYPE=GROUP macro statement are migrated into the named group in the CSD file. A new TYPE=GROUP statement overrides all previous ones.

If you do not insert such macros into your deck, the whole table is migrated into the same group on the CSD file. That group is, by default, named TCT*xx*, a substring of your table's name, DFHTCT*xx*.

It is better to create many small groups for your TERMINALs than one vast group. Aim to have no more than 100 resource definitions in any one group. (See “How many resource definitions should a group contain?” on page 10.)

You must include ACCMETH=VTAM in your DFHTCT TYPE=INITIAL macro. This is to enable the necessary VTAM control blocks, for example the ACB, to be built as part of the TCT. Note that the default for the ACCMETH= operand is ACCMETH=NONVTAM.

Some information from the TYPE=INITIAL macro is not migrated, because it is now specified using the DFHSIT macro. For further guidance on this, see the *CICS System Definition Guide*.

Assembling and link-editing

When you have edited your terminal control table source, assemble it using the CICS Transaction Server for OS/390 macro library, specifying MIGRATE=YES on the DFHTCT TYPE=INITIAL macro, and then link-edit the assembler output.

The assembly and link-edit of a TCT leads to the creation of two separate load modules. Assembly of a suffixed TCT (source name DFHTCT*xx*) produces a single text file. However, when this is link-edited into a load library, two members are created:
- DFHTCT*xx*, which contains the non-RDO-eligible definitions in control block format
- DFHRDTxx, which contains the RDO-eligible definitions in command format

You need to be aware of the existence of these two tables if you have to copy or move assembled TCT tables between load libraries.

If you get a return code greater than 4, remove the cause of the error and reassemble. (For example, internally duplicated entries are not acceptable.)

Assemble and link-edit your tables into your CICS load library, using procedure DFHAUPLE.

Using the MIGRATE command

After assembling it and link-editing it, you migrate the revised part of your table using the DFHCSDUP MIGRATE command. The form of the command is:

```
MIGRATE TABLE(DFHTCTxx) [TYPESGROUP(tgrpname)]
```

For guidance on the JCL for DFHCSDUP, see the *CICS Operations and Utilities Guide*.

The migration process triggered off by this MIGRATE command results in the creation of CSD file records for the following:
- TYPETERM definitions
- TERMINAL definitions
- SESSIONS definitions
- CONNECTION definitions

For the RDO equivalents of macro operands, see “Macro operand to RDO attribute” on page 531.

For the macro equivalents of RDO attribute keywords, see “RDO attribute to macro operand” on page 537.

The most important points about these definitions, for the purposes of migration, are described here.

TYPETERM definitions

These are derived from attributes of TYPE=TERMINAL macros, which are often identical for many terminals.

They are put into the CSD group named in the TYPESGROUP parameter. If no TYPESGROUP is specified, they are put in the group currently being created, with the TERMINAL definitions.

TCT migration

The “typeterm” attributes of each TYPE=TERMINAL table macro are checked with existing TYPETERM definitions and if they do not match any of these, a new TYPETERM is added to the CSD file.

The existing TYPETERMs checked are:

- TYPETERMs in the group currently being created
- TYPETERMs in the group specified in the TYPESGROUP parameter of the MIGRATE command

However, the scope of the checking is never extended to include any other TYPETERMs in other groups already on the CSD file. Such groups may have been created using RDO or by a previous MIGRATE.

For this reason, it is a good idea to use the TYPESGROUP parameter to avoid creating duplicate TYPETERMs in different groups. It is convenient to keep the TYPETERMs in a separate group anyway.

TYPETERMs created on the CSD file during the migration are named systematically, in a way related to the TRMTYPE parameter of the original terminal definition. The name consists of a prefix (of 3–5 characters) with a 3-character suffix.

For example, a TYPETERM defining attributes for a 3270 printer is named 3270P001. Variants with the same TRMTYPE are named 3270P002, and so on. The migration process ensures that this name is used as the TYPETERM parameter of every TERMINAL definition that references it.

Migration may produce some TYPETERM names that would be incomprehensible to those who have to use them when defining terminals. You can later rename such TYPETERMs using the RDO command RENAME, and ALTER the TERMINAL definitions that refer to them. The naming rules for TYPETERM identifiers are given on page 363.

TERMINAL definitions

Those operands of a TYPE=TERMINAL macro that are not accounted for by the TYPETERM named on the new TERMINAL definition become the other attributes of the TERMINAL definition. The TERMINAL name comes from the old TRMIDNT. The naming rules for TERMINAL identifiers in RDO are given on page 307.

This means that if the existing TRMIDNT contains characters not belonging to this set, the definition is not migrated from the TCT to the CSD file, and you must define these resources with CEDA, using new names acceptable to RDO.

All the TERMINAL definitions created by the migration process have AUTINSTMODEL(NO) and they all point to a TYPETERM definition with SHIPPABLE(NO).

SESSIONS definitions

SESSIONS definitions created by migration are different for different types of links and sessions. They are described in “Migrating different macro types” on page 553. SESSIONS identifiers in RDO are subject to the same character set restrictions as TERMINAL identifiers.

CONNECTION definitions

CONNECTION definitions created by migration are different for different types of links and sessions. They are described in “Migrating different macro types” on page 553 page 553. CONNECTION identifiers in RDO are subject to the same character set restrictions as TYPETERM identifiers. This restriction has always applied to the SYSIDNT name in the DFHTCT macro.

Checking the output from MIGRATE

When you have migrated some table entries, check that the process has worked satisfactorily.

The output listing from the MIGRATE utility tabulates all definitions successfully migrated to the CSD file. The listing contains diagnostic messages for resources that failed to migrate for some reason. For example, you may have terminals whose TRMIDNTs contain characters not acceptable to RDO.

You can use RDO to DEFINE resources that have failed to migrate. Make sure that you define these resources on the CSD file, and make any changes to application programs that depended on TERMINAL names that you have had to change, **before** you reassemble the TCT with MIGRATE=COMPLETE.

Using DFHCSDUP to add migrated groups to a list

You must install some migrated definitions in the system when you initialize it. You do this by means of a list named in the GRPLIST system initialization parameter. The MIGRATE command created groups of resource definitions. You must create a list by using the ADD command to add some of your groups to it. For more information about this, see “ADD” on page 38. You can choose a name for the list: you specify this name in the GRPLIST operand.

To start off with, include a group containing a definition of a terminal that you can use for RDO: to create other lists and groups, and to install other groups of definitions in the active CICS system. To enable you to use RDO, include in your list the CICS-supplied definitions for the resources RDO itself uses, and for the resources used by other CICS-supplied transactions, including CEMT, that you want to use. The easiest way to do this is to use the APPEND command to append the list called DFHLIST to your own list. For more information about this, see “APPEND” on page 89.

Operations after migration

When you are happy that you have no problems using RDO, how you continue depends on whether you can completely abandon the use of the TCT.

Continue to use a TCT containing resource definition macros if you have any of the following resources:
- BTAM terminals in a remote system
- Sequential devices
- Logical device codes (LDCs)

If you cannot abandon your TCT, follow the advice in “Final modification of the macro source”. If you can abandon your TCT, follow the advice in “Abandoning the use of DFHTCT macros (if possible)” on page 550.

Final modification of the macro source

If you need to retain a TCT to manage resources that you cannot define using RDO, you should eventually remove from your TCT source all the definitions that have been successfully migrated to the CSD file, or that are now being created by autoinstall. This saves time on table assemblies.

TCT migration

After reducing the TCT source to a bare minimum, reassemble it to ensure that you have not made any mistakes.

You continue to suffix the TCT, and code TCT=*xx*, where *xx* is the table name suffix, as a system initialization parameter. If you have VTAM resources **and** DFHTCT macros, code ACCMETH=(VTAM,NONVTAM) in the TCT.

Abandoning the use of DFHTCT macros (if possible)

If you are able to use RDO, with or without autoinstall, to manage **all** the resources formerly defined using the DFHTCT macroinstructions, you can now abandon your DFHTCT macro source altogether. Proceed as follows:

1. First check that the assembly confirms that all the definitions in the TCT were eligible for RDO.
2. Check again that the CSD file now contains all the definitions needed for operation with RDO. If you are using autoinstall, make sure that the CSD file contains all necessary model TERMINAL definitions.
3. If the CSD file now contains all the necessary definitions, code TCT=NO as a system initialization parameter. This ensures that a “dummy” TCT (DFHTCTDY) is used.

   DFHTCTDY is supplied by IBM. It contains predefined values for the DFHTCT TYPE=INITIAL parameters. If you wish to use different DFHTCT TYPE=INITIAL parameters, you may code and assemble your own version of DFHTCTDY. Note that some of the old DFHTCT TYPE=INITIAL options are now coded as system initialization parameters. For further guidance on this, see the *CICS System Definition Guide*.

Where next?

If you have only straightforward terminal definitions to migrate, you can go ahead and migrate them now.

If you have remote terminal definitions for transaction routing, you can find more guidance about migrating them in the next section.

This appendix ends with a summary of the different types of TCT macro, showing what the migration process does with each type. You may find this particularly helpful in planning your migration if you have intercommunication resources.

Migrating remote terminal definitions

“Terminals for transaction routing” on page 291 describes three methods of defining terminals so that they can be used for transaction routing. If you use transaction routing with MRO or APPC ISC, read that section before migrating your TCT.

When you have decided which method you want to use to define transaction routing terminals, follow the appropriate migration procedure described below. How you perform the migration depends on the type of macros you have used to define your remote terminals. If you have TYPE=REMOTE entries, you may have to migrate them to become remote definitions, just as you migrate the ordinary TYPE=TERMINAL entries to become local definitions. If you have TYPE=REGION macros, you probably use copy book definitions for both local and remote entries, as shown in this example. We refer to the migration procedure for this example in the procedures below.

Table DFHTCTLA used for local system (ACIC)

```
DFHTCT TYPE=INITIAL,SUFFIX=LA,SYSIDNT=ACIC
  .
  .
  .
COPY COMTERMS
  .
  .
  .
DFHTCT TYPE=FINAL
```

Table DFHTCTRB used for remote system (BCIC)

```
DFHTCT TYPE=INITIAL,SUFFIX=RB,SYSIDNT=BCIC
   .
   .
   .
DFHTCT TYPE=REGION,SYSIDNT=ACIC
COPY COMTERMS
   .
   .
   .
DFHTCT TYPE=FINAL
```

Method 1

Maintaining local and remote definitions separately

To migrate your TCT definitions to one or more CSD files, do the following:

1. Assemble the TCT for the terminal-owning system.
2. Assemble the TCT for the application-owning system. If you have more than one application-owning system, they may be able to share remote definitions, so you may not need to assemble more than one application-owning TCT.
3. Allocate definitions for different systems to different groups, if sharing a CSD file between systems.

   In the copy book example, if you use TYPE=GROUP macros to delimit groups within COMTERMS, edit COMTERMS after assembling DFHTCTLA and before assembling DFHTCTRB to change the TYPE=GROUP macros to name a different set of groups. (This does not apply if the tables are to be migrated to different CSD files.)
4. Use DFHCSDUP to migrate all the TCTs. The commands look like this:

   ```
   MIGRATE TABLE(DFHTCTLA) TYPESGROUP(TTS)
   MIGRATE TABLE(DFHTCTRB) TYPESGROUP(TTS)
   ```

   Migration of the remote terminals normally creates definitions that use the same TYPETERMs created for the corresponding local definitions.
5. Include the groups containing the local definitions in the GRPLIST for the terminal-owning system.
6. Include the groups containing the remote definitions in the GRPLIST for each application-owning system.

If there are only a small number of terminals that need more than one definition, it is probably best to migrate all the definitions in the terminal-owning TCT and then to use RDO to create the corresponding remote definitions with the required REMOTESYSTEM attribute. You can use the COPY and ALTER commands to do this.

Method 2

Sharing terminal definitions

To migrate your TCT definitions to the shared CSD file:

1. Suppress the assembly of remote definitions in all the TCTs to be migrated, by removing (or commenting-out) the TCT source for:
   - TYPE=REMOTE entries
   - TYPE=REGION and subsequent TYPE=TERMINAL entries
2. Assemble the TCT for the terminal-owning system, suppressing any macros for remote definitions that you may have if there is more than one terminal-owning system.
3. Assemble the TCT for the application-owning system, if it contains any definitions for local terminals. Again, suppress any macros for remote definitions if you need to assemble this TCT.
4. Use DFHCSDUP to migrate all the TCTs. The commands look like this:

   ```
       MIGRATE TABLE(DFHTCTLA) TYPESGROUP(TTS)
       MIGRATE TABLE(DFHTCTRB) TYPESGROUP(TTS)
   ```

   Migrating the remote terminals normally creates definitions that use the same TYPETERMs created for the corresponding local definitions.
5. Initialize the CICS terminal-owning system, installing at least one TERMINAL definition and other resource definitions necessary for using RDO.
6. Use the ALTER command to name the REMOTESYSTEM as the SYSIDNT of the terminal-owning system, on all the TERMINAL definitions that might be shared. You can probably use a generic name to do this in one command. For example:

   ```
   CEDA ALTER TERMINAL(*) GROUP(TTS) REMOTESYSTEM(ACIC)
   ```
7. Include the groups containing the definitions in the GRPLIST for each system, terminal-owning and application-owning.

Method 3

Making terminal definitions shippable

To migrate your TCT definitions to one or more CSD files:

1. Suppress the assembly of remote definitions in all the TCTs to be migrated, by removing (or commenting-out) the TCT source for:
   - TYPE=REMOTE entries
   - TYPE=REGION and subsequent TYPE=TERMINAL entries
2. Assemble the TCT for the terminal-owning system, suppressing any macros for remote definitions that you may have if there is more than one terminal-owning system.
3. Assemble the TCT for the application-owning system, if it contains any definitions for local terminals. Again, suppress any macros for remote definitions if you need to assemble this TCT.
4. Use DFHCSDUP to migrate the TCT. The command looks like this:

   ```
       MIGRATE TABLE(DFHTCTLA) TYPESGROUP(tgrpname)
   ```
5. Initialize the CICS terminal-owning system, installing at least one TERMINAL definition and other resource definitions necessary for using RDO.
6. Use the ALTER command to change the SHIPPABLE attribute to YES, on all the TYPETERM definitions that might be used for shipping. You can probably use a generic name to do this in one command. For example:

```
CEDA ALTER TYPETERM(*) GROUP(grpname) SHIPPABLE(YES)
```

7. Include the groups containing the definitions in the GRPLIST for the terminal-owning system.

Migrating different macro types

The rest of this appendix summarizes the different types of TCT macro that you might have, and what the results of the RDO migration are. It tells you how to migrate macros for the following devices and systems:

- Remote terminals for transaction routing
- MVS Consoles
- Pipeline terminals for VTAM pooled sessions
- Devices with LDC lists
- Terminals referencing printers
- MRO links and sessions
- LUTYPE6.1 CICS-CICS ISC links and sessions
- LUTYPE6.1 CICS-IMS links and sessions
- APPC (LUTYPE6.2) links and parallel sessions
- APPC (LUTYPE6.2) single session terminal
- INDIRECT connections

Remote terminals for transaction routing

You may have coded your macros in one of two ways:

- Individual terminals naming remote system:

```
DFHTCT TYPE=REMOTE,SYSIDNT=ssss,TRMIDNT=tttt,TRMTYPE=.....
```

- A series of TYPE=TERMINAL macros not naming the SYSIDNT, but following a TYPE=REGION that names the SYSIDNT:

```
DFHTCT TYPE=REGION,SYSIDNT=ssss
DFHTCT TYPE=TERMINAL,TRMIDNT=ttt1,TRMTYPE=yyyy...
DFHTCT TYPE=TERMINAL,TRMIDNT=ttt2,TRMTYPE=yyyy...
       ... and so on.
```

DFHCSDUP MIGRATE always produces a TERMINAL-TYPETERM pair of definitions:

```
DEFINE TERMINAL(tttt) GROUP(g) TYPETERM(xxxxxxxx)
       REMOTESYSTEM(ssss)
DEFINE TYPETERM(xxxxxxxx) GROUP(g) DEVICE(dddddddd)
```

Matching TYPETERMs are eliminated by DFHCSDUP MIGRATE, as described for ordinary terminals.

MVS consoles

The definition of an MVS console is essentially a special case of a local terminal definition. The TYPETERM in RDO identifies the device as a console, with the CONSOLE value for the DEVICE attribute. The console identifier for a particular console is specified on the TERMINAL definition as the CONSOLE attribute, which is the equivalent of the CONSLID= value on the macro. It is also possible to use the CONSNAME attribute of the TERMINAL definition (see page 302).

Pipeline terminals for VTAM pooled sessions

These terminals represent a special case of the definition of VTAM terminals.

A sequence of TYPE=TERMINAL macros is coded, the last one being tagged with PIPELN=LAST to indicate that the pool is complete:

```
DFHTCT TYPE=GROUP,GROUP=poolg
DFHTCT TYPE=TERMINAL,TRMIDNT=ttt1,TRMTYPE=(3600|3650),
      NETNAME=nnnnnnn1,SESTYPE=PIPELN,PIPELN=POOL
DFHTCT TYPE=TERMINAL,TRMIDNT=ttt2,TRMTYPE=(3600|3650),
      NETNAME=nnnnnnn2,SESTYPE=PIPELN,PIPELN=POOL
DFHTCT TYPE=TERMINAL,TRMIDNT=ttt3,TRMTYPE=(3600|3650),
      NETNAME=nnnnnnn3,SESTYPE=PIPELN,PIPELN=POOL
DFHTCT TYPE=TERMINAL,TRMIDNT=ttt4,TRMTYPE=(3600|3650),
      NETNAME=nnnnnnn4,SESTYPE=PIPELN,PIPELN=LAST,TASKNO=nn
```

When migrated using:

```
MIGRATE TABLE(DFHTCTxx) TYPESGROUP(typeg)
```

these macros result in the following definitions:

```
DEFINE TERMINAL(ttt1) GROUP(poolg) POOL(nnnnnnn1) TYPETERM(xxxxxxx1)
      NETNAME(nnnnnnn1)
DEFINE TERMINAL(ttt2) GROUP(poolg) POOL(nnnnnnn1) TYPETERM(xxxxxxx1)
      NETNAME(nnnnnnn2)
DEFINE TERMINAL(ttt3) GROUP(poolg) POOL(nnnnnnn1) TYPETERM(xxxxxxx1)
      NETNAME(nnnnnnn3)
DEFINE TERMINAL(ttt4) GROUP(poolg) POOL(nnnnnnn1) TYPETERM(xxxxxxx1)
      NETNAME(nnnnnnn4) TASKLIMIT(nn)
DEFINE TYPETERM(xxxxxxx1) GROUP(typeg) DEVICE(3600|3650)
      SESSIONTYPE(PIPELINE)
```

The POOL name is automatically generated using the NETNAME of the first TERMINAL in the POOL.

When the pool is installed, the terminal IDs are sorted in ascending alphabetic order. The first terminal to be installed becomes the pool header.

Matching TYPETERMs are eliminated by DFHCSDUP MIGRATE, as described for ordinary terminals.

Devices with LDC lists

For 3600, 3770 batch, 3770 and 3790 batch data interchange, and LUTYPE4 logical units, you can specify the name of an LDC list (Logical Device Code list). In RDO this information is held on the TYPETERM definition.

You cannot define the LDC list itself using RDO. The LDC list and its contents must still be defined by the macro method. The TERMINAL and TYPETERM using the LDC list can be created in RDO, and the LDC list defined by using the macro is named by the LDCLIST attribute on the TYPETERM definition.

You cannot define a list of LDC codes explicitly on the TYPETERM definition.[6] This simplifies the interface to these facilities and allows tables to be migrated to the CSD file. If you have a DFHTCT TYPE=TERMINAL macro with an LDC= specification of the form:

```
LDC=(aa=nnn,bb=nnn,...)
```

the assembly produces a level 8 MNOTE. This tells you to recode the list either as a **local LDC list** or as an **extended local LDC list**. (See page 351.)

Then recode the terminal entry in this form:

6. This also applies to the DFHTCT TYPE=TERMINAL macro, although in earlier CICS releases you could define of a list of LDC codes explicitly.

```
DFHTCT TYPE=TERMINAL,TRMIDNT=tttt,TRMTYPE=uuuuu,
       ACCMETH=VTAM,LDC=nnnnnnnn
```

where nnnnnnnn is the name you gave your LDC list when you defined it.

DFHCSDUP MIGRATE always produces a TERMINAL-TYPETERM pair of definitions:

```
DEFINE TERMINAL(tttt) TYPETERM(xxxxxxxx)
DEFINE TYPETERM(xxxxxxxx) DEVICE(dddddddd) LDCLIST(nnnnnnnn)
```

Terminals referencing printers

When a terminal references a printer, A pair of TCT entries could be related by using the PRINTTO or ALTPRT operands in the TCT macro, which referred to the printer TCTTE by means of the assembler label of the printer entry.

In RDO, the terminal refers to its associated printer by the 4-character TERMINAL name (old TRMIDNT) of the definition for the printer. The corresponding keywords in RDO are PRINTER and ALTPRINTER.

```
        DFHTCT TYPE=TERMINAL,TRMIDNT=tttt,
               TRMTYPE=3270,PRINTTO=label1
label1  DFHTCT TYPE=TERMINAL,TRMIDNT=pppp,TRMTYPE=3270P
```

When migrated, this becomes:

```
DEFINE TERMINAL(tttt) TYPETERM(xxxxxxxx) PRINTER(pppp)
DEFINE TERMINAL(pppp) TYPETERM(xxxxxxxx)
```

Links and sessions—method 1

This method applies both to MRO and to LUTYPE6.1 CICS-CICS ISC links and sessions.

MRO links and sessions

An MRO link and a set of parallel sessions are defined by:

```
DFHTCT TYPE=SYSTEM,SYSIDNT=ssss,ACCMETH=IRC|(IRC,XM)
```

DFHCSDUP MIGRATE always produces a CONNECTION-SESSIONS pair of definitions:

```
DEFINE CONNECTION(ssss) ACCESSMETHOD(IRC|XM)
DEFINE SESSIONS(sssssprp) CONNECTION(ssss) PROTOCOL(LU61)
       SENDPFX(sp)        SENDCOUNT(m)
       RECEIVEPFX(rp)     RECEIVECOUNT(n)
```

The SESSIONS name *sssssprp* is synthesized by concatenating the names of the SYSIDNT, SEND(*sp*) and RECEIVE(*rp*), for example:

```
SYSIDNT=BCIC,SEND=(SA,5),RECEIVE=(RA,3)
        =====>   SESSIONS(BCICSARA)
```

LUTYPE6.1 CICS-CICS ISC links and sessions

These are as for MRO, but with ACCESSMETHOD(VTAM).

LUTYPE6.1 CICS-CICS ISC links and sessions may also be defined and migrated as for method 2.

Links and sessions—method 2

This method applies to both LUTYPE6.1 CICS-CICS ISC and LUTYPE6.1 CICS-IMS links and sessions.

LUTYPE6.1 CICS-IMS links and sessions

The ISC link is defined by:

```
DFHTCT TYPE=SYSTEM,SYSIDNT=ssss,ACCMETH=VTAM
```

The parallel sessions for the link are defined individually by:

```
DFHTCT TYPE=TERMINAL,TRMIDNT=tttt,SYSIDNT=tttt,TRMTYPE=LUTYPE6,
      SESTYPE(SEND|RECEIVE),NETNAMQ=nnnnnnnn
```

DFHCSDUP MIGRATE produces a CONNECTION definition from the TYPE=SYSTEM macro:

```
DEFINE CONNECTION(ssss) ACCESSMETHOD(VTAM) PROTOCOL(LU61)
```

and a SESSIONS definition from each subsequent TYPE=TERMINAL macro.

For a SEND session:

```
DEFINE SESSIONS(sssstttt) CONNECTION(ssss) PROTOCOL(LU61)
      SESSNAME(tttt)     NETNAMEQ(nnnnnnnn)
      SENDCOUNT(1)
```

For a RECEIVE session:

```
DEFINE SESSIONS(sssstttt) CONNECTION(ssss) PROTOCOL(LU61)
      SESSNAME(tttt)     NETNAMEQ(nnnnnnnn)
      RECEIVECOUNT(1)
```

The SESSIONS name *sssstttt* is synthesized by concatenating the old SYSIDNT and TRMIDNT values.

The SESSNAME name *tttt* is the macro TRMIDNT value.

The NETNAMEQ name *nnnnnnnn* is the macro NETNAMQ value.

LUTYPE6.1 CICS-CICS ISC links and sessions

These are as for CICS-IMS but without NETNAMEQ.

LUTYPE6.1 CICS-CICS ISC links and sessions may also be defined and migrated as for method 1.

APPC (LUTYPE6.2) links and parallel sessions

For APPC, the sessions are grouped into modesets. Each modeset is defined in the macro method with a TYPE=MODESET macro, and the equivalent in RDO is one SESSIONS definition on the CSD file.

The ISC link is defined by:

```
DFHTCT TYPE=SYSTEM,SYSIDNT=ssss,ACCMETH=VTAM,TRMTYPE=LUTYPE62
```

The parallel sessions for the link are defined collectively by:

```
DFHTCT TYPE=MODESET,MODENAM=mmmmmmmm,SYSIDNT=ssss,
      MAXSESS=(m1,m2)
```

DFHCSDUP MIGRATE produces a CONNECTION definition from the TYPE=SYSTEM macro:

```
DEFINE CONNECTION(ssss) ACCESSMETHOD(VTAM) PROTOCOL(APPC)
```

and a SESSIONS definition from the TYPE=MODESET macro:

```
DEFINE SESSIONS(xxxxxxx)  CONNECTION(ssss) PROTOCOL(APPC)
       MAXIMUM(m1,m2)     MODENAME(mmmmmmmm)
```

The SESSIONS name *xxxxxxxx* is derived from the old SYSIDNT value concatenated with a 3-character identifier generated using the same algorithm as the old macro. For example, for SYSIDNT=SYS1:

```
MODESET1: MAXSESS=4  gives SESSIONS(SYS1AAC) (starting count)
MODESET2: MAXSESS=2  gives SESSIONS(SYS1AAG) (MODESET1 + 4)
MODESET3: MAXSESS=3  gives SESSIONS(SYS1AAI) (MODESET2 + 2)
```

The MODENAME name *mmmmmmmm* is the macro MODENAM value.

APPC (LUTYPE6.2) single session terminal

A single TYPE=SYSTEM macro is used:

```
DFHTCT TYPE=SYSTEM,SYSIDNT=ssss,ACCMETH=VTAM,TRMTYPE=LUTYPE62,
       FEATURE=SINGLE,MODENAM=mmmmmmmm
```

DFHCSDUP MIGRATE produces a CONNECTION definition and a SESSIONS definition:

```
DEFINE CONNECTION(ssss) ACCESSMETHOD(VTAM) PROTOCOL(APPC)
       SINGLESESS(YES)

DEFINE SESSIONS(xxxxxxx)  CONNECTION(ssss) PROTOCOL(APPC)
       MODENAME(mmmmmmmm) MAXIMUM(1,0)
```

The SESSIONS name *xxxxxxxx* is derived from the old SYSIDNT value concatenated with a 3-character identifier generated using a similar algorithm to the old macro. For example, for SYSIDNT=SYS1:

```
1st definition: gives SESSIONS(SYS1AAC) (starting count)
2nd definition: gives SESSIONS(SYS1AAF) ...
```

The MODENAME name *mmmmmmmm* is the macro MODENAM value.

You cannot autoinstall these CONNECTION and SESSIONS definitions. If you want to use autoinstall for your APPC single session terminals, you must redefine them as TERMINALs referencing a TYPETERM with DEVICE(APPC) (see page 290).

INDIRECT connections

The intermediate system is defined as:

```
DFHTCT TYPE=SYSTEM,SYSIDNT=yyyy,ACCMETH=IRC|(IRC,XM)|VTAM
```

This is migrated as any other TYPE=SYSTEM macro would be.

The indirect link is defined as:

```
DFHTCT TYPE=SYSTEM,SYSIDNT=xxxx,ACCMETH=INDIRECT,INDSYS=yyyy,
       NETNAME=nnnnnnnn
```

The migration of the macro for the indirect link produces an RDO definition of the form:

```
DEFINE CONNECTION(xxxx) INDSYS(yyyy) ACCESSMETHOD(INDIRECT)
       NETNAME(nnnnnnnn)
```

Index

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 request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail, to this address:

  Information Development Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  United Kingdom
- By fax:
  - From outside the U.K., after your international access code use 44–1962–870229
  - From within the U.K., use 01962–870229
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink™: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

- The publication number and title
- 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.

SC33-1684-33

Spine information:

CICS TS for OS/390 CICS Resource Definition Guide Release 3