CICS® Transaction Server for VSE/ESA™

System Definition Guide

Release 1

SC33-1651-05

CICS® Transaction Server for VSE/ESA™

System Definition Guide

Release 1

SC33-1651-05

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 341.

Sixth Edition (September 2005)

This edition applies to Release 1 of CICS Transaction Server for VSE/ESA, program number 5648-054, 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.

The CICS for VSE/ESA Version 2.3 *System Definition and Operations Guide* remains applicable and current for users of CICS for VSE/ESA Version 2.3.

Order publications through your IBM representative or the IBM branch office serving your locality.

At the back of this publication is a page entitled “Sending your comments to IBM”. If you want to make any comments, please use one of the methods described there.

Contents

Preface

What this book is about

This book describes the system definitions you need to run a CICS Transaction Server for VSE/ESA Release 1 in a Virtual Storage Extended/Enterprise System Architecture (VSE/ESA) environment.

CICS is supplied as part of the VSE/ESA system package. A working CICS system is automatically installed to run in partition F2 of VSE/ESA.

VSE/ESA allows you to install further predefined CICS systems to run in other partitions. See the books in the VSE/ESA library, particularly the *VSE/ESA Administration* manual, for more information about how to do this.

Who this book is for

Read this book if you are a system programmer responsible for specifying and installing the system definitions and resources for a CICS system.

What you need to know to understand this book

You should have experience of the VSE/ESA operating system, and either have previous experience of CICS, or at least be familiar with CICS concepts and terminology.

It is also assumed, when describing the jobs required to tailor CICS resource definitions, that you are familiar with VSE job control language (JCL) and other cataloged procedures.

How to use this book

The topics of this book are self-contained. Use an individual topic where it contains information about the particular task you are engaged in. For example, see Part 1, “Installing resource definitions” on page 1 if you are defining CICS data sets.

Notes on terminology

The terms listed in Table 1 are commonly used in the CICS Transaction Server for VSE/ESA Release 1 library. See the *CICS Glossary* for a comprehensive definition of terminology.

Table 1 (Page 1 of 2). Commonly used words and abbreviations in CICS Transaction Server for VSE/ESA Release 1

Term	**Definition (and abbreviation if appropriate)**
$(the dollar symbol)	In the character sets and programming examples given in this book, the dollar symbol ($) 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.
BSM	BSM is used to indicate the basic security management supplied as part of the VSE/ESA product. It is RACROUTE-compliant, and provides the following functions: • Signon security • Transaction attach security
C	The C programming language
CICSplex	A CICSplex consists of two or more regions that are linked using CICS intercommunication facilities. Typically, a CICSplex has at least one terminal-owning region (TOR), more than one application-owning region (AOR), and may have one or more regions that own the resources accessed by the AORs
CICS Data Management Facility	The new CICS Transaction Server for VSE/ESA Release 1 facility to which all statistics and monitoring data is written, generally referred to as “DMF”
CICS/VSE	The CICS product running under the VSE/ESA operating system, frequently referred to as simply “CICS”
COBOL	The COBOL programming language
DB2 for VSE/ESA	Database 2 for VSE/ESA which was previously known as “SQL/DS”.

Table 1 (Page 2 of 2). Commonly used words and abbreviations in CICS Transaction Server for VSE/ESA Release 1

Term	**Definition (and abbreviation if appropriate)**
ESM	ESM is used to indicate a RACROUTE-compliant external security manager that supports some or all of the following functions: • Signon security • Transaction attach security • Resource security • Command security • Non-terminal security • Surrogate user security • MRO/ISC security (MRO, LU6.1 or LU6.2) • FEPI security.
FOR (file-owning region)—also known as a DOR (data-owning region)	A CICS region whose primary purpose is to manage VSAM and DAM files, and VSAM data tables, through function provided by the CICS file control program.
IBM C for VSE/ESA	The Language Environment version of the C programming language compiler. Generally referred to as “C/VSE”.
IBM COBOL for VSE/ESA	The Language Environment version of the COBOL programming language compiler. Generally referred to as “COBOL/VSE”.
IBM PL/I for VSE/ESA	The Language Environment version of the PL/I programming language compiler. Generally referred to as “PL/I VSE”.
IBM Language Environment for VSE/ESA	The common runtime interface for all LE-conforming languages. Generally referred to as “LE/VSE”.
PL/I	The PL/I programming language
VSE/POWER	Priority Output Writers Execution processors and input Readers. The VSE/ESA spooling subsystem which is exploited by the report controller.
VSE/ESA System Authorization Facility	The new VSE facility which enables the new security mechanisms in CICS TS for VSE/ESA R1, generally referred to as “SAF”
VSE/ESA Central Functions component	The new name for the VSE Advanced Function (AF) component
VSE/VTAM	“VTAM”

Summary of changes

In previous versions of CICS for VSE/ESA, the *System Definition and Operations Guide* published information about system definition and operational procedures. In CICS Transaction Server for VSE/ESA Release 1, the *System Definition and Operations Guide* is split into two books:

- The *System Definition Guide*
- The *Operations and Utilities Guide*.

The *System Definition Guide* deals with system definition (system initialization parameters, the startup job stream, and creating CICS data sets and installing them).

Information describing CICS operations (such as starting CICS up and shutting it down) and using the various CICS utilities (such as DFHCSDUP) which used to be in the *System Definition and Operations Guide* is now in the *Operations and Utilities Guide*.

Although this book may seem familiar (we have tried to keep it looking as much like the old *System Definition and Operations Guide* as possible), you still need to read it carefully, as there are numerous changes. For example, there are updated examples and sample jobs, and new, changed and deleted options and values on system initialization parameters.

Information removed owing to obsolete function

Support for the following functions is withdrawn, and information relating to these subjects is removed from this book.

Table 2 (Page 1 of 2). Summary of information removed from this book

Function	Reason
System generation and the DFHSG macros	The need for system generation is removed by the provision of more CICS components in standard, pregenerated form; and by increasing the scope of the RDO and system initialization options available.
Macro-level programs	CICS Transaction Server for VSE/ESA Release 1 runs in a command-level only programming environment. You must convert macro-level programs to command-level. You can use the DFHMSCAN tool to locate macro-level statements, and the CICS Application Migration Aid (AMA) to help you convert your macro-level programs to command-level. See page 42 for more information about these tools. Programs include user application programs, user-replaceable modules and user exits.
VS COBOL II, DOS/VS PL/I and C/370 compilers	Support for programs compiled using these compilers has been removed because they are no longer supported. VS COBOL II compiled programs will continue to run under LE but DOS/VS PL/I and C/370 programs ***must*** be recompiled with the appropriate LE-enabled compiler.
RPG application programs	Information relating to RPG is removed because the RPG programming language is no longer supported.

Table 2 (Page 2 of 2). Summary of information removed from this book

Function	Reason
PPT and PCT resources, and TCT VTAM resources	Support for these tables is removed in favour of resource definition online (RDO). These tables are retained only for the purpose of migrating your existing resources to RDO. DFHTCT macro support is also retained for the purpose of defining non-VTAM devices such as sequential terminals, remote terminals and logical device codes.
Local BTAM terminals	Support for BTAM terminals is removed, and information relating to BTAM is removed from this book. However, BTAM terminals running in an earlier release of CICS are supported by transaction routing from those CICS systems to a CICS Transaction Server for VSE/ESA Release 1 system.
CICS internal security	Security is now handled by an external security manager (ESM). Support for the CICS signon tables (SNTs) and the user-replaceable security programs are also obsolete. You should define operator definitions in the ESM database.
Allocating storage for CICS dynamic storage areas and their cushions	Changes to the CICS DSAs (see Table 3 on page xii) means that the SCS system initialization parameter is removed. CICS dynamically manages the individual DSA and cushion sizes to optimize storage usage. You no longer need to restart CICS to tune these areas enabling CICS regions to operate continuously for longer than in previous releases.
Shutdown statistics and transient data queues	The CSSL, CSSM and CSSN queues are obsolete. Ensure that any JCL you have does not refer to them.

New information

Information for the following new functions is added to this book.

Table 3. Summary of new information in this book

Function	Description	See page ...
Data Management Handler (DMF) data sets	All monitoring and statistical data is written in SMF 110 record format to DMF data sets.	125.
Front End Programming Interface support (FEPI)	In support of the new Front End Programming Interface (FEPI) feature, the FEPI system initialization parameter and the transient data queues, CSZL and CSZX are added.	105.
Persistent sessions support	The RMTRAN system initialization parameter is added to support persistent sessions.	258.
Persistent verification	You can specify how long entries can remain in signed-on-from lists for those connections that have persistent verification specified in a CONNECTION resource definition using the PVDELAY system initialization parameter.	256.
VSAM global and local catalogs	The global catalog (DFHGCD) and the local catalog (DFHLCD) replace the restart data set (RSD) and preserve information about CICS resources and the status of CICS domains between separate executions of CICS.	Chapter 16, “Defining and using catalog data sets” on page 151.

Changed information

Changes to the following existing CICS functions has led to information changes in this book.

Autoinstall

The following new system initialization parameters are added for autoinstall.

- AIEXIT allows you to specify the name of the autoinstall module to be used when autoinstalling VTAM terminals and APPC connections. See page 215 for more information.
- AILDELAY allows you to specify the delay period to elapse after a session between CICS and an autoinstalled terminal is ended before the terminal entry is deleted. See page 216 for more information.
- AIQMAX allows you to specify the maximum number of VTAM terminals and APPC connections that can be queued concurrently for autoinstall. See page 216 for more information.
- AIRDELAY allows you to specify the delay period to elapse after emergency restart before autoinstall entries that are not in session are deleted. See page 217 for more information.

Autoinstall for programs

There are three new system initialization parameters for autoinstall for programs:

- PGAICTLG allows you to specify whether autoinstalled program definitions are to be cataloged. See page 250 for more information.
- PGAIEXIT allows you to specify the name of the autoinstalled exit program. See page 251 for more information.
- PGAIPGM allows you to specify the state of the program autoinstall function at initialization. See page 251 for more information.

CICS monitoring

All CICS monitoring data is written in SMF 110 record format to Data Management Facility (DMF) Data Handler data sets. See Chapter 13, “Defining data sets for the Data Management Facility (DMF)” on page 125 for information about creating DMF data sets.

There are also several new system initialization parameters:

- The MN system initialization parameter switches CICS monitoring on or off. See page 244 for more information.
- The MNCONV system initialization parameter specifies whether conversational tasks are to have separate performance class records for each terminal I/O request. See page 245 for more information.
- The MNEXC system initialization parameter switches exception class monitoring on or off. See page 246 for more information.
- The MNFREQ system initialization parameter specifies the class interval when monitoring should automatically produce a separate performance class record for a long-running transaction. See page 246 for more information.

- The MNPER system initialization parameter switches performance class monitoring on or off. See page 246 for more information.
- The MNSYNC system initialization parameter specifies whether or not separate performance class records should be produced for each syncpoint request. This excludes syncpoint requests during task termination or rollback. See page 246 for more information.
- The MNTIME system initialization parameter specifies whether monitoring should return the timestamp fields to an application program in GMT or local time. See page 246 for more information.

CICS statistics

All CICS statistics data is written in SMF 110 record format to Data Management Facility (DMF) Data Handler data sets. See Chapter 13, "Defining data sets for the Data Management Facility (DMF)" on page 125 for information about creating DMF data sets.

The STATRCD system initialization parameter is new and allows you to set the statistics recording status on or off. See page 266 for more information.

CICS storage and the CICS dynamic storage areas

The number of CICS-managed DSAs is increased from one to eight. Four DSAs reside above the 16MB line, and four reside below:

- The CICS dynamic storage area (CDSA) and the extended CICS dynamic storage area (ECDSA) are used for all CICS-key programs, control blocks, and task-lifetime storage.
- The shared dynamic storage area (SDSA) and the extended shared dynamic storage area (ESDSA) are used for user-key non-reentrant programs, and for all storage obtained by programs issuing EXEC CICS GETMAIN commands with the SHARED option.
- The user dynamic storage area (UDSA) and the extended user dynamic storage area (EUDSA) are used for all user-key task-lifetime storage.
- The read-only dynamic storage area (RDSA) and the extended read-only dynamic storage area (ERDSA) are used for all reentrant programs and tables.

See "Storage protection" on page 301 for more information.

There are several new system initialization parameters in the area of storage:

- The DSALIM and EDSALIM system initialization parameters set the overall dynamic storage limits above and below the 16MB line. Within these overall storage limits, CICS controls the individual DSA sizes and their associated storage cushions. See page 231 and page 234 respectively for more information.
- The CDSASZE, RDASZE, SDSASZE, UDSASZE, ECDSASZE, ERDSASZE, ESDSASZE, EUDSASZE system initialization parameters specify whether you want CICS to fix the size of the individual DSA's. The default size for each is ***0*** indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA is fixed.

- The RENTPGM system initialization parameter specifies whether you want CICS to allocate the read-only DSAs, RDSA and ERDSA, from read-only key-0 protected storage. See page 257 for more information.
- The new storage protection override facility has the following new system initialization parameters:
  - CWAKEY allows you to specify the storage key for the common work area (CWA) if CICS is operating with storage protection override facility active. See page 228 for more information.
  - STGPROT allows you to activate or deactivate the storage protection override facility. See page 267 for more information.
  - TCTUAKEY allows you to specify the storage key for TCTUAs if storage protection is active. See page 271 for more information.

In general, more virtual storage is freed by moving CICS nucleus modules, CICS tables and BMS map sets above the 16MB[1] line The following system initialization parameters are new:

- CHKSTRM allows you to activate or deactivate terminal storage-violation checking. See page 221 for more information.
- CHKSTSK allows you to activate or deactivate task storage-violation checking. See page 221 for more information.
- STGRCVY allows you to specify whether CICS is to attempt to recover from a storage violation. See page 267 for more information.
- TSMGSET allows you to specify the number of entries for which dynamic storage is allocated for storing pointers to records put into the temporary storage message set. See page 275 for more information.

Dump

- The INFO/ANA dump analyzer, DFHDAP, is obsolete and is replaced by the new dump formatting program, DFHPD410. DFHPD410 processes unformatted VSE system dumps (SDUMPS) in the VSE/ESA SYSDUMP library. New formatting options allow you selectively format portions of the dump. See the *CICS Operations and Utilities Guide* for more information about DFHPD410.
- The CICS transaction dump formatting program, DFHDUP, is obsolete and is replaced by the new dump formatting program, DFHDU410. DFHDU410 formatting options allow you selectively format the contents of a transaction dump data set. See the *CICS Operations and Utilities Guide* for more information about DFHDU410.

You can specify whether CICS is to switch automatically to the next transaction dump data set when the first becomes full using the new DUMPSW system initialization parameter. See page 233 for more information.

[1] MB equals 1 048 576 bytes.

Dynamic transaction routing

You can specify the name of the transaction definition required for dynamic transaction routing on the new DTRTRAN system initialization parameter. See page 232 for more information.

Journaling

The introduction of support for automatic journal archiving allows you to write data to one journal data set while a second is being archived.

A new data set called the **journal archive control data set** (JACD) controls the submission of archiving jobs and controls the reuse of journal data sets. The JACD contains the status of the journal data sets. When an archiving operation completes, the status is updated in the JACD so that CICS can reuse the journal data set. See Chapter 12, "Defining data sets for journaling and archiving" on page 113 for more information about journaling.

Security

CICS internal security is obsolete. Security responsibilities are delegated to an ESM.

Support for Advanced-Program-to-Program (APPC) session security is added

The following system initialization parameters are new:

- CMDSEC allows you to specify whether you want CICS to honor the CMDSEC option specified on a transaction's resource definition. See page 222 for more information.
- DFLTUSER allows you to specify the ESM userid with the security attributes you want for all terminal users who have not signed on explicitly. See page 229 for more information.
- ESMEXITS allows you to specify whether you want installation data to be passed via the RACROUTE interface to your ESM for use in exits written for the ESM. See page 236 for more information.
- PLTPIUSR allows you to specify the userid that CICS is to use to for security checking for PLT programs that run during CICS initialization. See page 252 for more information.
- PLTPISEC allows you to specify whether or not CICS is to perform command security checking for PLT programs during CICS initialization. See page 252 for more information.
- SEC allows you to specify the level of external security you require. See page 259 for more information.
- XCMD allows you to specify whether CICS is to perform command security checking. See page 278 for more information.
- XUSER allows you to specify whether or not CICS is to peform surrogate user checks. See page 286 for more information.

The CICS system definition (CSD) file

You can specify read-only access to the CSD by coding

```
// EXEC DFHCSDUP,SIZE=DFHCSDUP,PARM='CSD(READONLY)'
```

on the PARM parameter of the EXEC statement. The default access to the CSD remains read/write.

You can associate a descriptive comment (up to 58 characters in length) with any resource definition in the CSD. The DESCRIPTION field is added to all CEDA and CEDB DEFINE and ALTER panels.

There are several new system initialization parameters for the CSD:

- CSDACC allows you to specify the type of access to the CSD. See page 225 for more information.
- CSDBUFND allows you to specify the number of buffers for CSD data. See page 226 for more information.
- CSDBUFNI allows you to specify the number of buffers for the CSD index. See page 226 for more information.
- CSDFRLOG allows you to specify a forward recovery journal identifier. See page 226 for more information.
- CSDJID allows you to specify an identifier for automatic journaling. See page 227 for more information.
- CSDLSRNO allows you to specify a VSAM local shared resource pool. See page 227 for more information.
- CSDRECOV allows you to specify whether or not the CSD is recoverable. See page 227 for more information.
- CSDSTRNO allows you to specify the number of strings for concurrent requests. See page 228 for more information.

Trace

There are new system initialization parameters to support the changes made to tracing facilities.

- AUXTRSW allows you to switch the auxiliary trace autoswitch facility on and off. See page 218 for more information.
- CONFDATA allows you to hide user data that might otherwise appear in CICS trace entries (or dumps) that contain the VTAM Receive Any Input Area (RAIA). See page 223 for more information.
- CONFTXT allows you specify whether CICS is to prevent VTAM from tracing user data. See page 225 for more information.
- INTTR allows you to activate the internal CICS trace destination at initialization time. See page 243 for more information.
- SPCTR allows you to set the tracing level for all CICS components. See page 262 for more information.
- SPCTRxx allows you to set the tracing level for individual CICS components. See page 263 for more information.

- STNTR allows you to set the level of tracing required for CICS as a whole. See page 267 for more information.
- SYSTR allows you to control the master trace flag. See page 269 for more information.
- TRTABSZ allows you to specify the size of the internal trace table. See page 273 for more information.
- USERTR allows you to switch the master trace flag on and off. See page 275 for more information.

VSE shared virtual area (SVA)

There are new modules that must reside in the SVA, including DFHCSVC, the CICS SVC module. See Chapter 5, “CICS programs in the VSE shared virtual area” on page 59 for more information.

XRF

The following new system initialization parameters are added for XRF:

- The XRFSOFF system initialization parameter to control whether all users signed onto the active CICS system are to remain signed-on following a takeover. See page 283 for more information.
- The XRFSTME system initialization parameter to specify the time delay for users who are still signed on when an XRF takeover occurs. See page 283 for more information.

Changes to the startup job streams

The sample start up job streams are extensively reworked to take into account the numerous changes introduced by CICS Transaction Server for VSE/ESA Release 1. See Chapter 23, “CICS startup” on page 289 for more information.

Part 1. Installing resource definitions

After you have installed the VSE/ESA™ system package, you must define and install the resources definitions that CICS® needs to run user transactions.

Table 4. Road map for installing CICS resource definitions

If you want ...	**Refer to...**
See an introduction to resource definition.	Chapter 1, “Resource definition—an introduction” on page 3.
Know how to define resources in CICS control tables.	Chapter 2, “Defining resources in CICS control tables” on page 7.
Know how to install mapsets and partitionsets.	Chapter 3, “Installing mapsets and partition sets” on page 15.
Know more about installing application programs.	Chapter 4, “Installing application programs” on page 33.
Know how to access DL/I databases using CICS online applications.	Chapter 6, “Accessing DL/I databases using CICS online applications” on page 67.
Know how to define terminal resources.	Chapter 7, “Defining terminal resources” on page 71.

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:

- Terminals
- Files
- Programs
- Journals
- Transactions
- Connections
- Databases

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 methods of **resource definition** shown in Table 5.

Table 5. Ways in which to define CICS resources

Method	Description
Resource definition online (RDO)	This method uses the CICS-supplied online transactions CEDA, CEDB, and CEDC. Definitions are stored on the CICS system definition file (CSD), and are installed into an active CICS system from the CSD.
DFHCSDUP offline utility	This method also stores definitions in the CSD. DFHCSDUP allows you to make changes to definitions in the CSD by means of a batch job submitted offline.
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.
Systems Programming (using the EXEC CICS CREATE commands)	You can use the EXEC CICS CREATE commands to create resources independently of the CSD file.
Macro definition	You can use assembler macro source to define resources. Definitions are stored in assembled tables in a program library, from where they are installed during CICS initialization.

Depending on the resources you want to define, you can use one or more of these methods.

Table 6 on page 4 shows you the methods you can use for each resource.

Table 7 on page 5 suggests some of the things you should consider when deciding which definition method to use when there is a choice.

Table 6. Resources and how you can define them

Resource	RDO/EXEC CICS CREATE commands	DFHCSDUP	Autoinstall	Macro
Connections	Yes	Yes	Yes	No
DL/I VSE Databases	No	No	No	Yes
Files (DAM)	No	No	No	Yes
Files (VSAM)	Yes	Yes	No	Yes
Journals	No	No	No	Yes
Local shared resource (LSR) pools	Yes	Yes	No	Yes
Mapsets	Yes	Yes	Yes	No
Partitionsets	Yes	Yes	Yes	No
Partners	Yes	Yes	No	No
Profiles	Yes	Yes	No	No
Programs	Yes	Yes	Yes	No
Queues (destinations)	No	No	No	Yes
Sessions	Yes	Yes	No	No
Shared data tables	Yes	Yes	No	Yes
Temporary storage	No	No	No	Yes
Terminals (non-VTAM)	No	No	No	Yes
Terminals (VTAM®)	Yes	Yes	Yes	No
Transactions	Yes	Yes	No	No
Transaction classes	Yes	Yes	No	No
Typeterms	Yes	Yes	No	No

Table 7. Methods of resource definition

Method	Description	Advantages	Disadvantages
Resource definition online (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, and you should use some form of auditing as a control mechanism.
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, or modify, them in a CEDA transaction.
DFHCSDUP offline utility	DFHCSDUP is an offline utility which allows you to define, list, and modify resources by means of a batch job. DFHCSDUP can be invoked as a batch program or from a user-written program running in batch mode. If using a user-written program, 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 CSD as long as it is not in use.	• You cannot install resources into an active CICS system. • If you are sharing a CSD between regions, it must not be in use by another CICS system while you are running DFHCSDUP.
Automatic installation (autoinstall)	This applies to VTAM terminals, LU62 sessions, programs, mapsets, and partitionsets. You set up “model” definitions using either RDO or DFHCSDUP, then CICS can create and install new definitions for these resources dynamically, based on the models.	If you have large numbers of resources, a lot of time could be taken up defining 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 tables	Using this method, you code and assemble macroinstructions to define resources in the form of tables.	Where possible, you should 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.

For information about CICS resource definition, see the *CICS Resource Definition Guide*. For information about the DFHCSDUP utility, see the *CICS Operations and Utilities Guide*.

Resource definitions in the CSD are stored in **groups**. On a COLD start of CICS you specify the resource definitions required in a particular run of CICS by a **list** of groups. You can specify up to four lists of groups to be installed during CICS initialization, using the GRPLIST system initialization parameter. You can also use the CEDA INSTALL command to install a resource definition or group of definitions defined in the CSD[2] dynamically on a running CICS system.

You should limit read/write access to resource definitions in the CSD to a small number of people. To do this, you can:

- Protect groups of resources by using the CEDA command LOCK
- Protect the list of resource groups specified in the system initialization parameter GRPLIST by using the CEDA command LOCK
- Use the CEDB transaction to create resource definitions, but not to INSTALL them
- Use the CEDC transaction for read-only access to resource definitions

CICS control tables contain resource definition records for resources that cannot be defined in the CSD. The tables and their resource definitions are created by using the CICS table assembly macro instructions. You must use macro instructions to define non-VTAM networks and terminals, non-VSAM files, databases, journals, queues, and resources for monitoring and system recovery. For more information about defining resources in CICS control tables, see Chapter 2, “Defining resources in CICS control tables” on page 7.

Using the CSD and control tables together

In two cases, you can mix resources defined in the CSD with resources defined in control tables. These are:

1. On a cold start, you can mix file control resources defined in the CSD with those that were defined using DFHFCT macros. VSAM and DAM file definitions are loaded from the DFHFCT load module first, then the definitions for VSAM files are loaded from the RDO groups specified in the GRPLIST system initialization parameter. When CICS is running, you can use CEDA commands to add more file resource definitions.

   However, if you mix resource definitions for files from the CSD and the FCT, the CSD definitions will replace the FCT definitions in the event of duplicated names, provided that there are no conflicting attributes. For more information, see the notes on the FCT system initialization parameter on 236.

2. You can also mix terminal resource definitions for non-VTAM[3] terminals defined in a TCT with resource definitions for VTAM terminals defined using RDO.

   However, avoid duplicate terminal IDs, because a TCT entry using the same terminal ID (TERMIDNT in the TCT) as a VTAM terminal in the CSD (TERMINAL name in the CSD), prevents CICS installing the VTAM definition.

[2] The CSD is independent of the running CICS system, because when you install the definitions in the CICS region, CICS copies the information and keeps it in its own storage. Because CICS does this, you can modify the CSD without interfering with the running CICS system. You can also change the definitions in the running CICS system by reinstalling them, or add more definitions by installing new ones.

[3] Non-VTAM terminals. TCT entries can be for SAM sequential devices, logical device codes (LDCs), and remote BTAM terminals required for ISC/MRO purposes.

Chapter 2. Defining resources in CICS control tables

This chapter describes what you should do to define resource definitions in CICS control tables. The tables and their resource definitions are created by using macros. You must use macros to define: non-VTAM networks and terminals, non-VSAM files, databases, journals, queues, and resources for monitoring and system recovery. For details about defining resource definitions in CICS control tables, and about migrating your tables to the CSD, see the *CICS Resource Definition Guide*.

For each of the CICS tables (listed on page 8) complete the following steps:

1. Code the resource definitions you require.
2. Assemble and link-edit these definitions to create a load module in the required CICS library. See "Assembling and link-editing control tables" on page 10 for more information about how to do this.

   The CICS-supplied macros used to create the CICS tables determine whether tables are loaded above the 16MB line. All tables, other than the JCT and TCT, are loaded above the 16MB line.
3. Name the suffix of the load module by a system initialization parameter. For most of the CICS tables, if you do not require the table you can code *tablename=NO*. The exceptions to this rule are as follows:
   - **The CLT**

     Specifying CLT=NO causes CICS to try and load DFHCLTNO.

     The CLT is only used in the alternate CICS, when you are running CICS with XRF, and is always required in that case.
   - **The SIT**

     Specifying SIT=NO causes CICS to try and load DFHSITNO.

     The SIT is always needed, and you can specify the suffix by coding the SIT system initialization parameter.
   - **The TCT**

     Specifying TCT=NO causes CICS to load a dummy TCT

     named DFHTCTDY, as explained on page 208.
   - **The TLT**

     Terminal list tables are specified by program entries

     in the CSD, and do not have a system initialization parameter.
   - **The MCT**

     Specifying MCT=NO causes the CICS monitoring domain to build

     dynamically a default monitoring control table. This ensures that default monitoring control table entries are always available for use when monitoring is on and a monitoring class (or classes) are active.
4. If you are running CICS with XRF, the active and the alternate CICS regions share the same versions of tables. However, to provide protection against DASD failure, you might want to run your active and alternate CICS systems

from separate sets of load libraries–in which case, you should make the separate copies ***after*** generating your control tables.

Table 8 lists the CICS tables that can be assembled, link-edited, and installed in your CICS libraries for use in your CICS system.

Table 8. CICS tables that you can assemble, link-edit, and install in libraries

Table	**Module Name**	**Abbreviation**
Command list table	DFHCLTxx	CLT
Destination control table	DFHDCTxx	DCT
File control table	DFHFCTxx	FCT
Journal control table	DFHJCTxx	JCT
Monitor control table	DFHMCTxx	MCT
Program list table	DFHPLTxx	PLT
System initialization table	DFHSITxx	SIT
System recovery table	DFHSRTxx	SRT
Terminal control table	DFHTCTxx	TCT
Terminal list table	DFHTLTxx	TLT
Temporary storage table	DFHTSTxx	TST
Transaction list table	DFHXLTxx	XLT

You can generate several versions of each CICS control table by specifying SUFFIX=xx in the macro that generates the table. This suffix is then appended to the default 6-character name of the load module.

To get you started, CICS provides the sample tables listed in Table 9 in the VSE/ESA sublibrary, PRD1.BASE.

Table 9. Sample CICS system tables

Table	**Suffix**	**Notes**
Command list table (CLT)	1$	XRF regions only
Destination control table (DCT)	2$	- - -
Journal control table (JCT)	2$	- - -
Monitor control table (MCT)	A$	For a CICS AOR
Monitor control table (MCT)	F$	For a CICS FOR
Monitor control table (MCT)	T$	For a CICS TOR
Monitor control table (MCT)	2$	CICS region with DL/I
System initialization table (SIT)	$$	Default system initialization parameters
System initialization table (SIT)	6$	- - -
System recovery table (SRT)	1$	- - -
Terminal control table (TCT)	5$	Non-VTAM terminals only

Unless you are using sequential devices, or need to define logical device code (LDC) lists, you do not need a TCT and should specify TCT=NO. (For information about the effect of TCT=NO, see page 208.)

You define VTAM terminals in the CSD only, either explicitly or by means of autoinstall model definitions, but for non-VTAM terminals you must use DFHTCT macros. For a summary of how you can define files and terminals to CICS, depending on the access methods you are using for these resources, see Table 6 on page 4.

Although you can modify and reassemble tables while CICS is running, you must shut down and restart CICS to make the new tables available to CICS. (The command list table (CLT) is an exception in that a new table can be brought into use without shutting down either the active CICS system or the alternate CICS system.)

Sample JCL to print the contents of a CICS control table

You can use the job shown in Figure 1 to print the contents of the CICS control tables.

```
// JOB PRTMEM
* JOB TO PRINT MEMBERS OF A LIBRARY
// EXEC LIBR
   ACCESS SUBLIB=PRD1.BASE
   LIST tablename.A
/*
/&
```

Figure 1. Sample job to print the contents of CICS tables

Migrating control tables

If you are using an external security manager (ESM), define operator characteristics in the ESM database. For migration considerations that apply specifically to security, see the *CICS Security Guide*.

Local BTAM terminals are not supported. However, BTAM terminals running in an earlier release of CICS are supported by transaction routing from those CICS systems to a CICS Transaction Server for VSE/ESA Release 1 system.

PCTs, PPTs, SNTs, VTAM TCTs, and local BTAM TCTs are not supported in CICS Transaction Server for VSE/ESA Release 1. You can use other control tables that you used with earlier releases. But, first check that the source statements are still valid (some parameters may have changed, or been added or deleted) then reassemble the tables against the CICS Transaction Server for VSE/ESA Release 1 libraries.

If you used PCTs, PPTs, or TCTs containing VTAM terminal definitions with an earlier release of CICS, these ***must*** be migrated to the CSD. CICS Transaction Server for VSE/ESA Release 1 provides macros (for migration purposes) for you to reassemble your PCTs, PPTs, or TCTs against before migrating them to the CSD using the DFHCSDUP utility MIGRATE command.

See the *CICS Resource Definition Guide* for further details of the DFHCSDUP utility MIGRATE command.

Defining control tables to CICS

You can assemble and link-edit more than one version of a table, and use a suffix to distinguish them. To specify which version you want CICS to use, you code a system initialization parameter of the form *tablename*=*xx*. (For example, TCT=5$.) However, you can code the SIT=xx parameter only as a startup override; that is, not in the DFHSIT table. For details of all the CICS system initialization parameters, see Chapter 22, “CICS system initialization” on page 187.

Other tables that have special requirements are terminal list tables (TLTs), and transaction list tables (XLTs). For each TLT, or XLT you must specify a program resource definition in the CSD, using the table name, including the suffix, as the program name. For example, to generate a TLT with a suffix of AA (DFHTLTAA), the CEDA command would be as follows:

```
CEDA DEFINE PROGRAM(DFHTLTAA) GROUP(grpname) LANGUAGE(ASSEMBLER)
```

For information about program and terminal list tables, see the **PLT—program list table** or **TLT (terminal list table)** in the *CICS Resource Definition Guide*.

The command list table (CLT) is used only by the alternate CICS in a CICS system running with XRF=YES, and differs in many other respects from the other CICS tables. For more guidance information, including information on resource definition specific to the CICS extended recovery facility, see the *CICS XRF Guide*.

Assembling and link-editing control tables

The steps and procedures for assembling and link-editing control tables are shown in Figure 2 on page 11.

You generate each table by assembling the appropriate macro (for example, the DFHSIT macro for the system initialization table) with the associated operands. The output from each macro assembly contains the linkage-editor control statements (PHASE and INCLUDE), needed to link-edit the table into your CICS or private sublibrary.

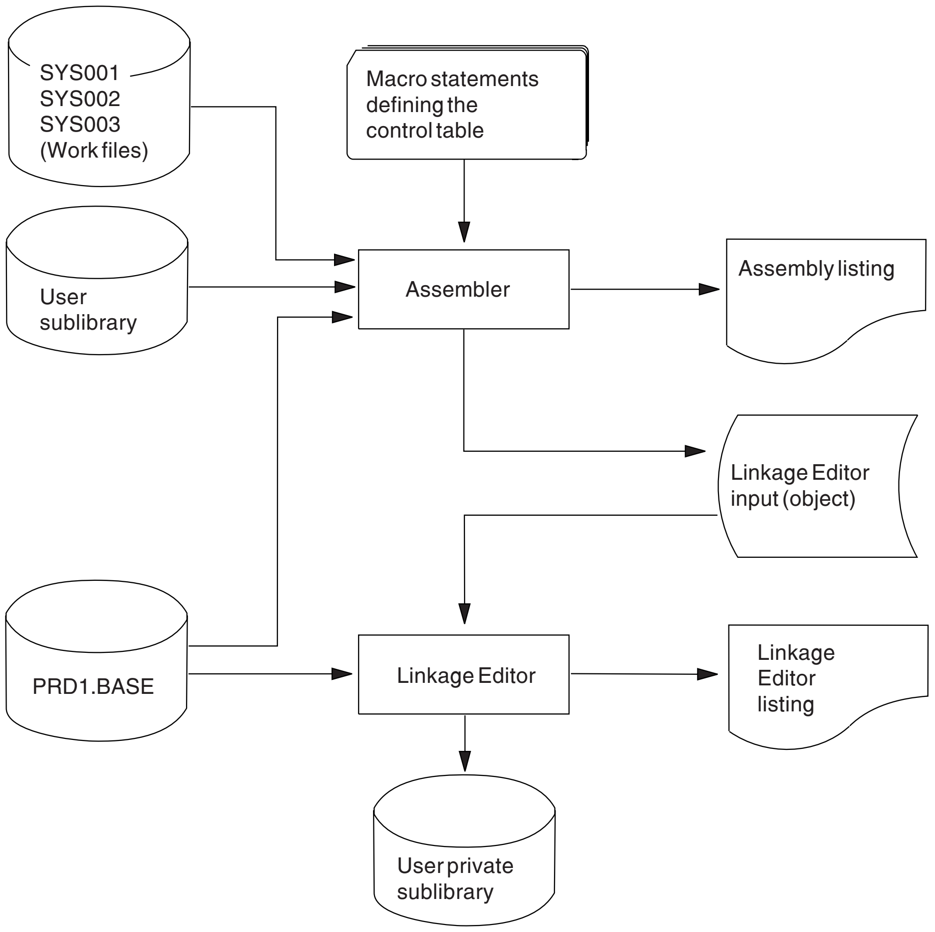


Figure 2. Installing the CICS control tables

Use an assembler END statement to terminate the group of source statements that define a control table. The END statement can include the label (symbol) of the table entry point, in the form DFHmmmBA, where mmm is the name of the table. (For example, DFHFCTBA for the file control table.)

If you don't provide an END statement, the assembler flags an error with a return code of 4. However, CICS still generates the correct DFHmmmBA entry point address from the TYPE=INITIAL macro for the table.

Figure 3 on page 12 gives an example of the job stream you need to assemble and link-edit a CICS control table.

```
// JOB ASMTABLE
* -----------------------------------------
* Assemble and link-edit CICS control tables
* -----------------------------------------
// OPTION  CATAL,NODECK,ALIGN
   ACTION SMAP                           1
// LIBDEF *,SEARCH=(user.source,PRD1.BASE),                            *
            CATALOG=user.tables,TEMP
// EXEC ASMA90,SIZE=ASMA90,PARM='SIZE(MAX-200K,ABOVE)'
         .
         macro source statements         2
         defining a CICS control table
         .
         END   DFHmmmBA
/*
// EXEC LNKEDT
/&
```

Figure 3. Assembling and link-editing a CICS control table

Notes for Figure 3:

1 You can use ACTION SMAP to produce a linkage-editor map, sorted into alphanumeric order of CSECTs.

2 If you are reassembling a sample table supplied by CICS, all of the sample tables are supplied in source form in the VSE/ESA library, PRD1.BASE, so you could use a COPY statement in place of the actual source statements. For example:

```
// EXEC ASMA90,SIZE=ASMA90,PARM='SIZE(MAX-200K,ABOVE)'
         COPY  DFHmmmss
         END   DFHmmmBA
/*
```

where `mmm` is the table mnemonic, and `ss` is the suffix.

General notes for JOB ASMTABLE:

- When a DCT or an FCT addressing DAM files is link-edited, the unresolved external reference $$$$$$$$ is listed and the module also has one unresolved address constant.
- **VSE link-edit message: 2158I NO CSECT LENGTH SUPPLIED**.
  If the “Linkage Editor Diagnostic of Input” report produces this

  message when you are assembling and link-editing a TCT, it is ***not*** an error, and you can safely ignore it.
- If you assemble and link-edit a TCT with ACCMETH=NONVTAM, the following unresolved external references are generated:

  `DFHZRPLS, DFHZRPL1, DFHZNIB, and DFHZBIND`

  These are ***not*** errors and you can safely ignore them. Do not ignore any other assembly or link-edit errors.

Defining control tables in the system initialization table

After successfully assembling and link-editing your table, define it in the system initialization table that you use to initialize your CICS system. Alternatively, you can change the existing SIT entry for the table at startup by specifying the new table explicitly as a system initialization override parameter. See Chapter 22, “CICS system initialization” on page 187 for further information.

Chapter 3. Installing mapsets and partition sets

This chapter describes how to assemble and link-edit mapsets and partition sets for use with Basic Mapping Support (BMS). BMS allows your application programs to:

- Generate device-dependent output data from a device-independent standard form
- Read device-dependent data and convert it to a device-independent standard form.

(It is assumed that you have defined these mapsets and partition sets using CICS-supplied macros.)

The chapter contains the following sections:

- “Defining maps and mapsets” which describes physical and symbolic description mapsets and how to catalog, assemble, and install them.
- “Using extended data stream terminals” on page 17 which looks at the macro support available for generating mapsets.
- “Defining maps and mapsets” on page 15 which shows you how to install physical and symbolic description mapsets, with sample jobs where applicable.
- “Installing partition sets” on page 29 which describes partition sets and shows you how to install them.

Defining maps and mapsets

When device-dependent output data is generated from a device-independent standard form, or when device-dependent data is read and converted into a device-independent standard form, the structure of the device-independent standard form and the layout of the data on the display terminal are determined by a **map**.

You define maps yourself and you can group related maps together (for example, maps used in the same application program) into a **mapset**.

CICS supports the definition of mapsets by assembler macros. You can also define them interactively, using licensed programs such as the Screen Definition Facility II VSE Release 6 (SDF II VSE). For further information about SDF II, see the *SDF II General Introduction* manual.

If you are a VM user, running VSE under the VM operating system, you can use the VM version of SDF II (program number 5664-307).

You must define two kinds of mapset:

1. Physical mapsets
2. Symbolic mapsets

For further guidance about the definition and use of maps and mapsets, see the *CICS Application Programming Guide*.

Physical mapsets

A physical mapset contains the device-specific information that BMS uses to translate data from the standard device-independent form used by your application programs to (and from) the device-dependent form required by terminals.

Cataloging physical mapsets

Catalog physical mapsets into a suitable sublibrary as phases, so that they are available when your CICS system is running.

Symbolic description mapsets

Symbolic description mapsets make it possible for your application programs to make symbolic references to fields in a physical mapset.

Application programs use symbolic description mapsets to define the standard device-independent form of the data. The form of this description depends on your application, as follows:

- A “DSECT” in assembler language
- A “struct” in C
- A “data definition” in COBOL
- A “BASED structure” or an “AUTOMATIC structure” in PL/I

Cataloging symbolic description mapsets

You can either catalog the symbolic description mapsets in a suitable sublibrary, or you can include them in the application program itself.

Generating physical and symbolic description mapsets

To produce the physical mapset that BMS uses, and also the symbolic storage definition that is copied into your application program, you assemble the mapset definition macros *twice*.

The two types of mapset can be distinguished by either:

- The TYPE operand of the DFHMSD macro
- The SYSPARM parameter in the OPTION statement (SYSPARM=‘MAP’ or SYSPARM=‘DSECT’). If you use the SYSPARM parameter, you must also code TYPE=&SYSPARM.

  The use of SYSPARM allows you to generate both the physical mapset and the symbolic description mapset from the same unchanged set of BMS mapset definition macros.

Assembling physical and symbolic description mapsets

You can assemble mapsets as either ***unaligned*** or ***aligned***. (An aligned map is one in which the length field is aligned on a halfword boundary.) You should use unaligned maps except in cases where an application package requires the use of aligned maps.

The distinction between aligned and unaligned mapsets is made by the use of the SYSPARM parameter on the OPTION statement when assembling the mapset. You use SYSPARM=‘A’ to indicate that you want aligned maps assembled; the type of map is taken from the TYPE operand on the DFHMSD macro.

Alternatively, you can specify both the mapset type and alignment by the SYSPARM parameter. SYSPARM=‘AMAP’ generates an aligned physical mapset, and SYSPARM=‘ADSECT’ generates an aligned symbolic description mapset.

BMS mapsets generated before CICS/DOS/VS 1.6

In releases of CICS earlier than CICS/DOS/VS 1.6, the form of BMS generated by the DFHSG PROGRAM=BMS macro assumes that all mapsets are either aligned, or unaligned. Aligned and unaligned mapsets cannot be mixed.

In releases of CICS later than CICS/DOS/VS 1.6, the physical mapset contains information indicating whether you have assembled it for aligned or unaligned maps. This information is tested at execution time, and the appropriate map alignment is used. Thus, aligned and unaligned mapsets can be mixed.

If you still use mapsets that were assembled before CICS/DOS/VS 1.6, the above alignment information is missing from the physical mapsets. Use the BMS operand of the DFHSIT macro, or the associated startup override parameter, to indicate whether mapsets assembled before CICS/DOS/VS 1.6 are aligned or unaligned. See page 219 for information about the BMS system initialization parameter.

Using extended data stream terminals

Offline map preparation support for the IBM® 3270 Information Display System was provided under earlier versions of BMS through the DFHMDI and DFHMDF macros, with no requirement for a DFHMSD macro. Such maps are no longer supported. You must modify the source definitions of such maps to use the DFHMSD macro and reassemble the resultant mapset.

Applications and maps designed for the 3270 Information Display System run unchanged on devices supporting extensions to the 3270 data stream such as color, extended highlight, programmed symbols, and validation. To use fixed extended attributes, such as color, you only need to reassemble the physical mapset. If your application program requires dynamic attribute modification, you must reassemble both the physical and symbolic description mapsets, and also reassemble (or recompile) the application program.

Installing mapsets

This section describes the installation of:

- Physical mapsets
- Symbolic description mapsets
- Physical mapsets and symbolic description mapsets in one job.

All the jobs use the SYSPARM parameter on the OPTION statement to distinguish between physical and symbolic description mapsets.

Installing physical mapsets

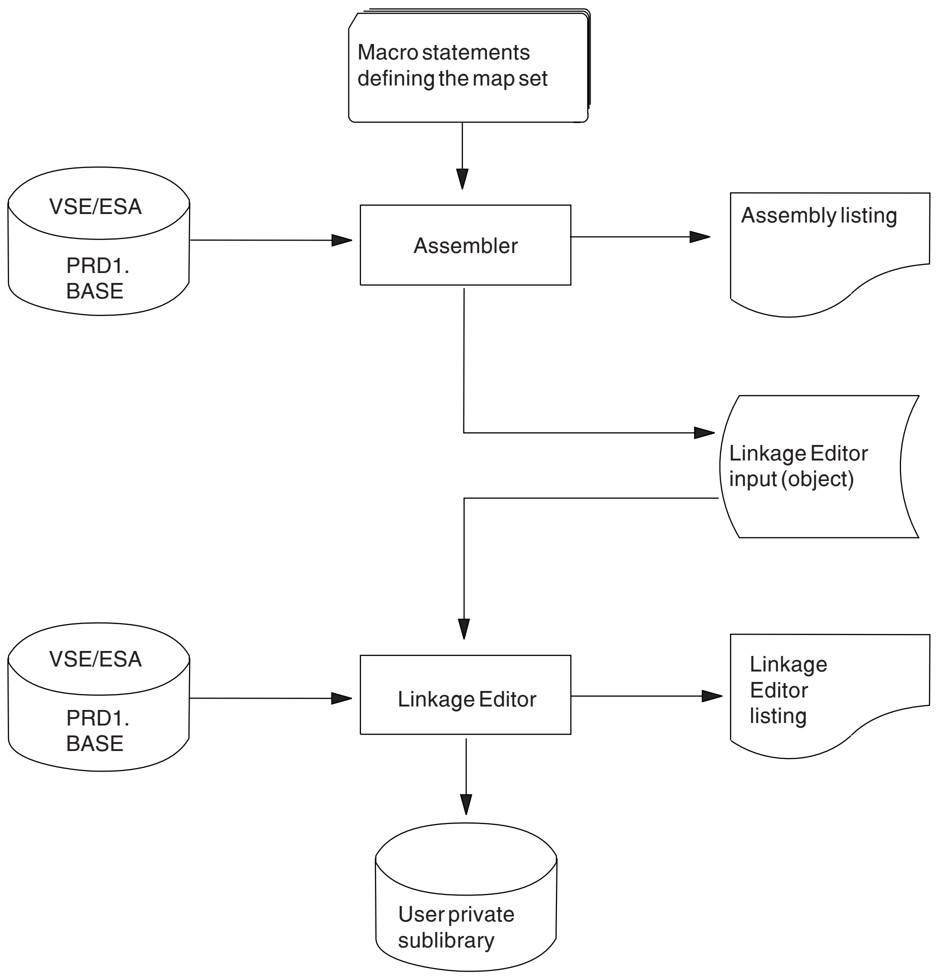


Figure 4. Installing physical mapsets

The job stream shown in Figure 5 on page 18 is an example of the assembly and link-edit job steps you need to install physical mapsets. (The second and third job steps are similar to the procedure for installing physical mapsets shown in Figure 4.)

```
// JOB BMSMAP
*  --------------------------------------------------
*  CICS for VSE/ESA Assemble and link-edit BMS mapset
*  --------------------------------------------------
// OPTION  CATAL,NODECK,ALIGN,SYSPARM='MAP'      1
   PHASE   name,*                                2
   MODE RMODE(24)                                3
// LIBDEF PHASE,CATALOG=cicsusr.cicslib
// LIBDEF SOURCE,SEARCH=PRD1.BASE
// EXEC ASMA90,SIZE=ASMA90,PARM='SIZE(MAX-200K,ABOVE)'
         .
         source statements defining the physical mapset
         .
         END
/*
// EXEC    LNKEDT
/&
```

Figure 5. Assembling and link-editing a physical mapset

Notes for Figure 5:

1 If you require halfword-aligned length fields, you must specify SYSPARM='AMAP' instead of SYSPARM='MAP'.

2 You must provide a PHASE statement to specify the name of the physical mapset you want BMS to load into storage. If the mapset is device-dependent, the PHASE name should be derived by appending the device suffix to the original 1- to 7-character mapset name used in the application program. The suffixes you can append for the various terminals supported by BMS depend on the value specified on the TERM or SUFFIX operand of the DFHMSD macro that defined the mapset. See the *CICS Application Programming Reference* for a complete list of mapset suffixes.

To use a physical mapset, you must define and install a resource definition for it. You can do this either by using the program autoinstall function, or by using the CEDA DEFINE MAPSET and INSTALL commands.

3 Physical mapsets are loaded into CICS-key storage, unless they are link-edited with the SVA option. If they are link-edited with this option, they are loaded into key-0 protected storage, provided that RENTPGM=PROTECT is specified on the RENTPGM system initialization parameter.

However, it is recommended that mapsets should not be link-edited with the SVA option because, in some cases, CICS modifies the mapset.

The MODE statement specifies whether the mapset is to be loaded above (RMODE(ANY)) or below (RMODE(24)) the 16MB line. RMODE(ANY) indicates that CICS can load the mapset anywhere in virtual storage, but tries to load it above the 16MB, if possible.

Defining entries for physical maps

Unless you are using program autoinstall, you can define entries for each physical mapset using the the CEDA DEFINE MAPSET command, the DFHCSDUP offline utility DEFINE MAPSET command, or an EXEC CICS CREATE MAPSET command. For more information about using the CEDA and DFHCSDUP DEFINE MAPSET commands, see the *CICS Resource Definition Guide*. For more information about the EXEC CICS CREATE command, see the *CICS System Programming Reference*.

Installing symbolic description mapsets

Symbolic description mapsets allow your application programs to make symbolic references to fields in a physical mapset. You assemble the source statements defining the symbolic description mapset, and direct the output to SYSPCH (which may be card, tape, or disk). If your installation uses many mapsets, or makes multiple use of common mapsets, you should place the symbolic description mapsets in a suitable sublibrary, from which they can be copied into any application program.

When you generate a symbolic description mapset under the same name for different programming languages, a separate copy must be placed on the sublibrary for each language. You can distinguish them with a different library member type, such as 'A' for assembler, 'H' for C, 'C' for COBOL, and 'P' for PL/I. Figure 6 illustrates the preparation of symbolic description maps for BMS.

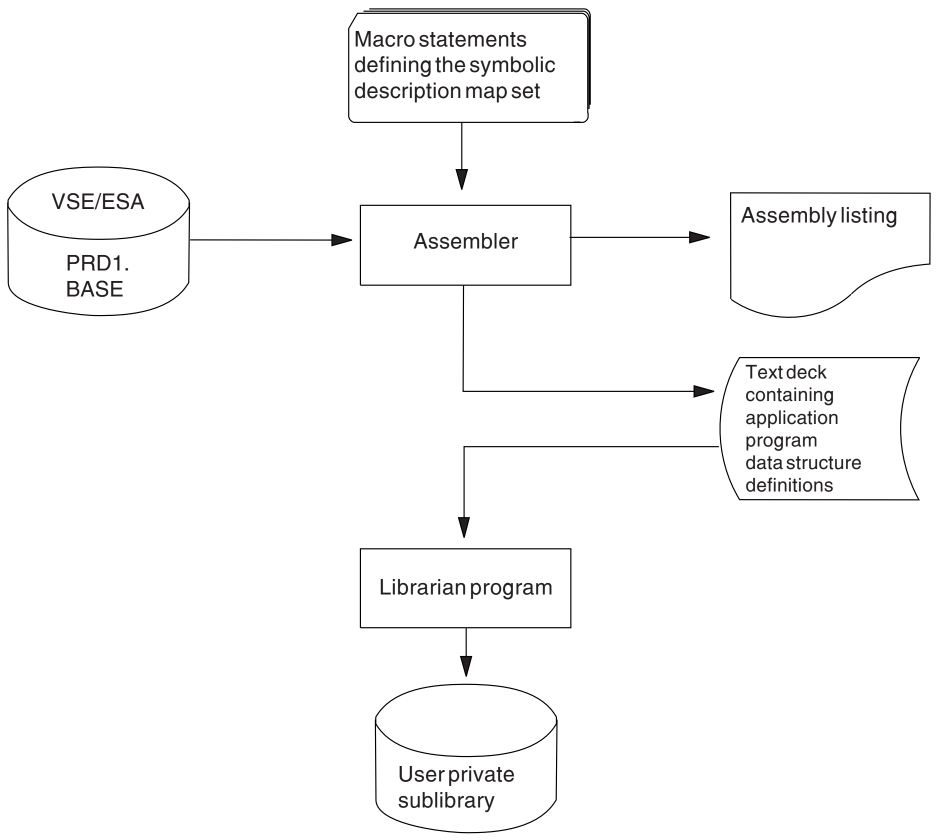

Macro statements defining the symbolic description map set
VSE/ESA
PRD1. BASE
Assembler
Assembly listing
Text deck containing application program data structure definitions
Librarian program
User private sublibrary

Figure 6. Installing symbolic description mapsets

To incorporate a symbolic description mapset in your programs, use the appropriate copy statement for the language you are using. You need only one symbolic description mapset corresponding to all the different suffixed versions of the physical mapset.

For example, if you want to run the same application on terminals with two different screen sizes, you would:

- Define two mapsets each with the same fields, but positioned to suit the screen sizes. These two mapsets should have the same name but different suffixes, the suffix corresponding to the terminal.
- Assemble and link-edit the different physical mapsets separately, but create only one symbolic description mapset, because the symbolic description mapset is the same for both of the physical mapsets.

Assembling, link-editing and cataloging a symbolic description map

The example in Figure 7 on page 21 can be used to assemble and catalog a symbolic map. It applies to symbolic description mapsets for any supported programming language. It uses disk as intermediate storage between the assembly and the librarian step, to catalog the symbolic description in a suitable sublibrary.

Using SYSPCH and SYSIPT

```
// JOB BMSMAPSY
* ----------------------------------
* Assemble and catalog symbolic map
* ----------------------------------
// OPTION  DECK,SYSPARM='DSECT'               1
// DLBL CICSUCT,'usercat',,VSAM
// DLBL CICSUSR,'vse.cics.library',,VSAM,CAT=CICSUCT
// DLBL   IJSYSPH,'asm.workfile',0
// EXTENT SYSPCH,,1,0,rtrk,ntrk
ASSGN SYSPCH,DISK,VOL=volid,SHR
*  The library search chain
// LIBDEF  *,SEARCH=(cicsusr.source,PRD1.BASE)
* ------------------------------------------
* STEP 1: Assemble symbolic map description
* ------------------------------------------
// EXEC ASMA90,SIZE=ASMA90,PARM='SIZE(MAX-200K,ABOVE)'
   PUNCH 'ACCESS SUBL=cicsusr.cicslib'        2
   PUNCH 'CATALOG mapname.x  REPLACE=YES'     2  3
    .
    Map source statements for the symbolic map description
    .
/*
* -----------------------------------------
* STEP 2: Catalog symbolic map description
* -----------------------------------------
CLOSE SYSPCH,cuu
// DLBL   IJSYSIN,'asm.workfile',0
// EXTENT SYSIPT
ASSGN SYSIPT,DISK,VOL=volid,SHR
// DLBL CICSUCT,'usercat',,VSAM               4
// DLBL CICSUSR,'vse.cics.library',,VSAM,CAT=CICSUCT
// EXEC LIBR
CLOSE  SYSIPT,cuu
/*
/&
// JOB RESET
ASSGN SYSIPT,SYSRDR   IF 1A93D, CLOSE SYSIPT,SYSRDR
ASSGN SYSPCH,cuu      IF 1A93D, CLOSE SYSPCH,cuu
/&
```

Figure 7. Assembling and cataloging a symbolic description mapset

Notes for Figure 7:

1 Specify the SYSPARM=‘ADSECT’ instead of SYSPARM=‘DSECT’ if you want halfword-aligned length fields.

2 Librarian requires the assembler job step to output the ACCESS and CATALOG commands before anything else. They are then the first statements on the file when it is read as SYSIPT by librarian.

3 The member type specified in the CATALOG statement should correspond to the programming language specified in the symbolic description map definition. Replace x with one of the following:

A for assembler language
C for COBOL
H for C
P for PL/I

4 This example assumes that you are cataloging the symbolic map description in a VSAM-managed library, and therefore includes the VSAM catalog and library DLBL statements.

Using IESINSRT

Figure 8 on page 23 shows an alternative to using SYSPCH and SYSIPT on disk. It uses the IESINSRT utility program which allows the job to run in a VSE dynamic partition and in a static partition. Using IESINSRT is faster than using SYSPCH and SYSIPT and avoids operational difficulties caused by incorrectly resetting permanent ASSGN statements for SYSPCH and (or) SYSIPT.

The need to use an intermediate disk file between assembly and LIBR is eliminated by running **two** VSE jobs. The first job, BMSMAPSY, builds the second job, BMSMAPS2, on the POWER RDR queue by using the POWER DISP=I facility. The IESINSRT utility program is used to build header JCL for running LIBR. The ASSEMBLY step builds the symbolic map data. The IESINSRT utility is then used again to add the trailer JCL for LIBR.

Job BMSMAPS2 runs after job BMSMAPSY has completed.

```
* $$ JOB JNM=BMSMAPSY,CLASS=x
* $$ LST CLASS=y
* $$ PUN DISP=I,CLASS=x,JNM=BMSMAPS2  1
// JOB BMSMAPSY  Build stage 2 job on RDR
// ASSGN SYSIPT,SYSRDR
// ON $CANCEL OR $ABEND GOTO ABEND  2
// EXEC IESINSRT  3
$ $$ LST CLASS=y
// JOB BMSMAPS2  Run stage 2 LIBR catalog
* ----------------------------------------------------------------
* STEP 2: Catalog symbolic map description
* ----------------------------------------------------------------
// DLBL CICSUCT,'usercat',,VSAM
// DLBL CICSUSR,'vse.cics.library',,VSAM,CAT=CICSUCT
// EXEC LIBR
* $$ END
// LIBDEF *,SEARCH=(cicsusr.source,PRD1.BASE)
// EXEC ASMA90,SIZE=ASMA90,PARM='SIZE(MAX-200K,ABOVE)'
   PUNCH 'ACCESS SUBL=cicsusr.cicslib'
   PUNCH 'CATALOG mapname.x R=Y'
   .
   Map source statements for symbolic map description
   .
         END
/*
/. ABEND
// EXEC IESINSRT
/+
/*
#&
$ $$ EOJ
* $$ END
/*
/&
* $$ EOJ
```

Figure 8. Using IESINSRT to assemble and catalog symbolic description mapsets (Job 1)

Notes for Figure 8:

1 All cards punched by job BMSMAPSY are placed back on the RDR queue as class 'x' with the POWER job name of BMSMAPS2.

2 The ON statement ensures that the trailer JCL is written out even if an abend occurs.

3 The IESINSRT utility program reads cards from SYSIPT until it reads the * $$ END statement. It translates column 1 $ to *, # to /, and punches the card out to SYSPCH. Note that only POWER statements and the VSE /& statement need to be altered.

The resultant BMSMAPS2 job looks as shown in Figure 9 on page 24

```
* $$ JOB JMN=BMSMAPS2,CLASS=x
* $$ LST CLASS=y
// JOB BMSMAPS2  Run stage 2 LIBR catalog
* -----------------------------------------------------------------
* STEP 2: Symbolic map description
* -----------------------------------------------------------------
// DLBL CICSUCT,'usercat',,VSAM
// DLBL CICSUSR,'vse,cics.library',,VSAM,CAT=CICSUCT
// EXEC LIBR
ACCESS SUBL=cicsusr.cicslib
CATALOG mapname.x R=Y
    .
    Symbolic map description
    .
/+
/*
/&
* $$ EOJ
```

Figure 9. Using IESINSRT to assemble and catalog symbolic description mapsets (Job 2)

Installing physical and symbolic description mapsets together

You can assemble and catalog the physical mapset and the symbolic description mapset in the same job using SYSPARM in the // OPTION job control statements for the assembler execution steps.

If you require unaligned length fields, you must use SYSPARM=‘MAP’ to produce the physical mapset, and SYSPARM=‘DSECT’ to produce the symbolic description mapset.

If you require halfword-aligned length fields, you must use SYSPARM=‘AMAP’ to produce the physical mapset, and SYSPARM=‘ADSECT’ to produce the symbolic description mapset.

Using SYSPCH and SYSIPT (unaligned physical and symbolic mapsets)

The job stream in Figure 10 on page 25 is an example of the assembly of unaligned BMS physical and symbolic description mapsets in one job, using disk as intermediate storage for the symbolic description mapset.

```
// JOB CICSBMS
* ----------------------------------------------------------------
* Assemble and catalog physical and symbolic description mapset
*  STEP 1: Catalog source statements into sublibrary
* ----------------------------------------------------------------
// DLBL CICSUCT,'usercat',,VSAM
// DLBL CICSUSR,'vse.cics.library',,VSAM,CAT=CICSUCT
// EXEC LIBR
    ACCESS  SUBL=cicsusr.source                                   1
    CATALOG  DFH$AMA.A  REPLACE=YES
MAPSETA  DFHMSD TYPE=&SYSPARM,MODE=INOUT,CTRL=(FREEKB,FRSET),         *
               LANG=ASM,TIOAPFX=YES,EXTATT=MAPONLY,COLOR=BLUE
DFH$AGA  DFHMDI SIZE=(12,40)
         .                                                        2
         .
         DFHMSD TYPE=FINAL
         END
/+                                                                3
/*
* ----------------------------
* STEP 2: Assemble physical map
* ----------------------------
// OPTION  CATAL,NODECK,SYSPARM='MAP',ALIGN
   PHASE DFH$AGA,*
* The library search chain
// LIBDEF *,SEARCH=(cicsusr.source,PRD1.BASE)
// LIBDEF PHASE,CATALOG=cicsusr.cicslib
// EXEC ASMA90,SIZE=ASMA90,PARM='SIZE(MAX-200K,ABOVE)'
   COPY  DFH$AMA
         END
/*
* ------------------------------------------------
* STEP 3:   Link-edit and catalog the map PHASE
// EXEC LNKEDT
* STEP 4:   Assemble the symbolic description map
* ------------------------------------------------
// OPTION  DECK,SYSPARM='DSECT'
// DLBL CICSUCT,'usercat',,VSAM
// DLBL CICSUSR,'vse.cics.library',,VSAM,CAT=CICSUCT
// DLBL IJSYSPH,'asm.workfile',0
// EXTENT SYSPCH,,1,0,120,15
ASSGN  SYSPCH,DISK,VOL=volid,SHR
// EXEC  ASMA90,SIZE=ASMA90,PARM='SIZE(MAX-200K,ABOVE)'
   PUNCH ' ACCESS  SUBL=cicsusr.tables '                        4
   PUNCH ' CATALOG DFH$AGA.A  REPLACE=YES '                     4  5
   COPY  DFH$AMA
         END
/*
```

Figure 10 (Part 1 of 2). Assembling and link-editing a physical and a symbolic description mapset

```
* --------------------------------------
* STEP 5: Catalog symbolic map description
* --------------------------------------
CLOSE SYSPCH,cuu
// DLBL CICSUCT,'usercat',,VSAM
// DLBL CICSUSR,'vse.cics.library',,VSAM,CAT=CICSUCT
// DLBL IJSYSIN,'asm.workfile',0
// EXTENT SYSIPT
ASSGN SYSIPT,DISK,VOL=volid,SHR
// EXEC LIBR
CLOSE  SYSIPT,cuu
/*
/&
// JOB RESET
ASSGN SYSIPT,SYSRDR   IF 1A93D, CLOSE SYSIPT,SYSRDR
ASSGN SYSPCH,cuu      IF 1A93D, CLOSE SYSPCH,cuu
/&
```

Figure 10 (Part 2 of 2). Assembling and link-editing a physical and a symbolic description mapset

Notes for Figure 10:

1 To catalog map source statements in a suitable user sublibrary (using the VSE librarian), specify the sublibrary using the ACCESS command, and the copybook name using the CATALOG command.

Note that this example is based on VSAM-managed libraries; make sure you specify a suitable value on the VSE SIZE parameter, one that allows enough GETVIS space for the execution of librarian.

2 The first few DFHMSD map source macros in this sample job are taken from one of the maps used by the FILEA sample application programs. The full map is supplied in the VSE/ESA sublibrary, PRD1.BASE, as member DFH$AMA.A. Note that the TYPE operand does not specify either “MAP” or “DSECT”, because these control statements are supplied later during the assembly stage by the SYSPARM parameter in the // OPTION statements in job step 2 (Assemble physical map) and job step 4 (Assemble symbolic description map). The phase from the link-edit stage is cataloged as DFH$AGA.

3 Terminate the data to be cataloged in a sublibrary with a valid end-of-data record; the default is /+ as in this example.

4 The fourth step assembles the symbolic map description ready for cataloging in your sublibrary by librarian. Because librarian reads the output from the assembler, you must precede your assembler source with the statements that create the necessary librarian commands (ACCESS and CATALOG).

5 In this example, the DFHMSD macro map source statements are cataloged as member DFH$AMA, but after assembly the symbolic description is cataloged as member DFH$AGA, type A. This naming convention is followed for all the CICS-supplied FILEA sample program maps.

Using IESINSRT (unaligned physical and symbolic mapsets)

You can use the IESINSRT utility program as an alternative to using ASSGN statements for SYSIPT and SYSPCH.

```
* $$ JOB JNM=CICSBMS,CLASS=x
* $$ LST CLASS=y
* $$ PUN DISP=I,CLASS=x,JNM=CICSBMS2  1
// JOB CICSBMS
* ------------------------------------------------------------------
* Assemble and catalog physical and symbolic description mapset
* STEP 1 : catalog source statements into sublibrary
* ------------------------------------------------------------------
// DLBL CICSUCT,'usercat',,VSAM
// DLBL CICSUSR,'vse.cics.library',,VSAM,CAT=CICSUCT
// LIBDEF *,SEARCH=(cicsusr.source,PRD1.BASE)
// EXEC LIBR
  ACCESS SUBL=cicsusr.source
  CATALOG DFH$AMA REPLACE=YES
MAPSETA DFHMSD TYPE=&SYSPARM,MODE=INOUT,CTRL=(FREEKB,FRSET),   *
               LANG=ASM,TIOAPFX=YES,EXTATT=MAPONLY,COLOR=BLUE
DFH$AGA DFHMDI SIZE=(12,40)
        .
        .
        DFHMDI TYPE=FINAL
        END
/+
/*
// ASSGN SYSIPT,SYSRDR
// ON $CANCEL OR $ABEND GOTO ABEND  2
// EXEC IESINSRT  3
$ $$ LST CLASS=y
// JOB CICSBMS2  Run stage 2 LIBR catalog
* ------------------------------------------------------------------
* STEP 5 : Catalog symbolic map description
* ------------------------------------------------------------------
// DLBL CICSUCT,'usercat',,VSAM
// DLBL CICSUSR,'vse.cics.library',,VSAM,CAT=CICSUCT
// EXEC LIBR
* $$ END
* ------------------------------------------------------------------
* STEP 2 : Assemble symbolic map description
* ------------------------------------------------------------------
// OPTION CATAL,NODECK,SYSPARM='MAP',ALIGN
// LIBDEF PHASE,CATALOG=cicsusr.source
// EXEC ASMA90,SIZE=ASMA90,PARM='SIZE(MAX-200K,ABOVE)'
   COPY  DFH$AMA
         END
/*
```

Figure 11 (Part 1 of 2). Using IESINSRT to assemble unaligned physical and symbolic mapsets (Job 1)

```
* ------------------------------------------------------------------
* STEP 3 : Link-edit and catalog the map PHASE
* ------------------------------------------------------------------
// EXEC LNKEDT
* ------------------------------------------------------------------
* STEP 4 : Assemble symbolic map description
* ------------------------------------------------------------------
// OPTION DECK,SYSPARM='DSECT'
// EXEC ASMA90,SIZE=ASMA90,PARM='SIZE(MAX-200K,ABOVE)'
   PUNCH 'ACCESS SUBL=cicsusr.cicslib'
   PUNCH 'CATALOG mapname.x R=Y'
   COPY  DFH$AMA
         END
/*
/. ABEND
// EXEC IESINSRT
/+
/*
#&
$ $$ EOJ
* $$ END
/*
/&
* $$ EOJ
```

Figure 11 (Part 2 of 2). Using IESINSRT to assemble unaligned physical and symbolic mapsets (Job 1)

Notes for Figure 11 on page 27:

1 All cards punched by job CICSBMS are placed back on the RDR queue as class 'x' with the POWER job name of CICSBMS2.

2 The ON statement ensures that the trailer JCL is written out even if an abend occurs.

3 The IESINSRT utility program reads cards from SYSIPT until it reads * $$ END. It translates column 1 $ to *, # to / and punches the card out to SYSPCH. Note that only POWER statements and the VSE /& need to be altered.

The resultant CICSBMS2 job looks as shown in Figure 12 on page 29

```
* $$ JOB JNM=CICSBMS2,CLASS=x
* $$ LST CLASS=y
// JOB CICSBMS2  Run stage 2 LIBR catalog
* ------------------------------------------------------------------
* Step 5: Catalog symbolic map description
* ------------------------------------------------------------------
// DLBL CICSUCT,'usercat',,VSAM
// DLBL CICSUSR,'vse.cics.library',,VSAM,CAT=CICSUCT
// EXEC LIBR
ACCESS SUBL=cicsusr.cicslib
CATALOG mapname.x R=Y
    .
    Symbolic map description
    .
/+
/*
/&
* $$ EOJ
```

Figure 12. Using IESINSRT to assemble unaligned physical and symbolic mapsets (Job 2)

Installing partition sets

Some terminals, such as the IBM 8775 Display Terminal, support screen division by partitions. You can split the available display area into a set of related **logical screens** called partitions. You define the layout and properties of the set of partitions that can be simultaneously displayed on a terminal in a **partition set**. CICS supports the definition of partition sets by assembler macros. Partition sets are handled in the same way as physical mapsets. There is no concept of a symbolic description partition set.

For further guidance about defining and using partition sets, see the *CICS Application Programming Guide*.

The procedure for installing partition sets is illustrated in Figure 13. The procedure is the same as the one for installing physical mapsets.

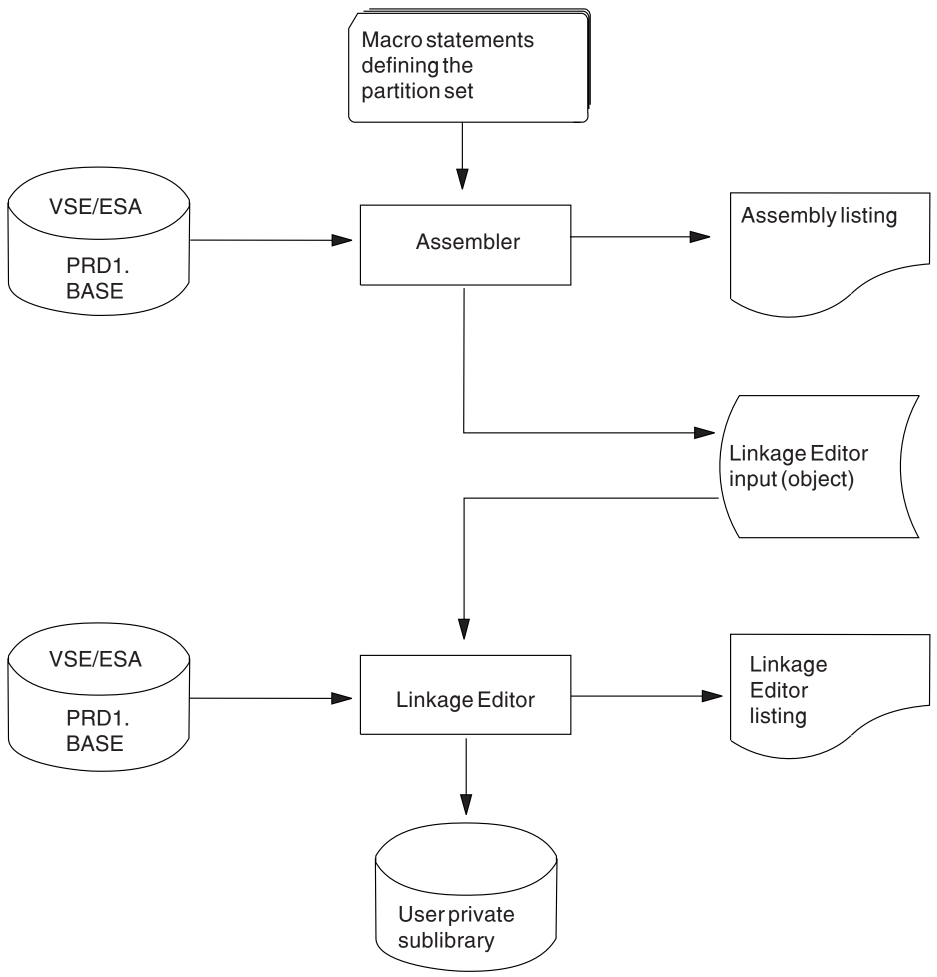


Figure 13. Installing partition sets

The job stream you need to install a partition set is similar to JOB BMSMAP, used to assemble and link-edit a physical mapset. If you want to install partition sets, use JOB BMSMAP, shown in Figure 5 on page 18, suitably modified for your purpose.

As with physical mapsets, you must provide a PHASE statement to specify the name of the partition set that you want BMS to load into storage. If the partition set is device-dependent, the phase name should be derived by appending the device suffix to the original partition set name (1–6 characters) used in the application program. The suffixes you can append for the various terminals supported by BMS depend on the value specified on the SUFFIX operand of the DFHPSD macro that defined the partition set.

For further guidance about BMS, see the *CICS Application Programming Guide*. For programming information about BMS and a complete list of partition set suffixes, see the *CICS Application Programming Reference* manual.

Entries for partition sets

Unless you are using autoinstall for programs, you must define each partitionset to CICS using the CEDA DEFINE PARTITIONSET command, the DFHCSDUP offline utility DEFINE PARTITIONSET command, or an EXEC CICS CREATE PARTITIONSET command.

For further guidance information about using the DEFINE PARTITIONSET command, see the *CICS Resource Definition Guide*. For more information about the EXEC CICS CREATE command, see the *CICS System Programming Reference*.

Chapter 4. Installing application programs

This chapter describes what you have to do to install an application program to run under CICS. **Application program** means any user program that uses the CICS command-level application programming interface (CICS API). Such programs use:

- SQL statements
- DLI requests
- Common programming interface (CPI) statements
- SAA Resource Recovery statements
- CICS front-end programming interface (FEPI) commands
- External CICS interface (EXCI) commands.

For information about writing CICS application programs, see the *CICS Application Programming Guide*.

Information can be found under the following headings:

CICS-supplied facilities for installing programs

This section describes the CICS-supplied translators, and interface modules that you can use to install application programs.

The CICS-supplied translators

Table 10 lists the translators supplied by CICS in the VSE/ESA sublibrary, PRD1.BASE.

Table 10. Translators supplied by CICS

Programming language	Translator name
Assembler	DFHEAP1$
C	DFHEDP1$
COBOL	DFHECP1$
PL/I	DFHEPP1$

The translators run in a VSE batch virtual partition of at least 256KB, depending on the size of the program you are translating.

You can use either disk or tape for intermediate storage for output from the translator.

The translators write out 81-byte records, but only 80 bytes are input to the compiler or assembler.

The DEBUG translator option is assumed by default. This allows the execution diagnostic facility (EDF) to display the line number of each command. Because the line number needs eight bytes of storage, you might want to suppress the line number by specifying NODEBUG when your application is fully tested.

For a description of the translation process, and information about the translator options that you can specify, see **The translation process** and **Specifying translator options** in the *CICS Application Programming Guide*.

The CICS-supplied interface modules

Each of your application programs to run under CICS must contain one or more interface modules (also known as **stubs**) to use the following CICS facilities:

- The EXEC interface
- The CPI Communications facility
- The SAA Resource Recovery facility

The EXEC interface modules

Each of your CICS application programs must contain an interface to CICS. This takes the form of an EXEC interface module, used by the CICS high-level programming interface. The module, found in the VSE/ESA sublibrary PRD1.BASE, must be link-edited with your application program to provide communication between your code and the EXEC interface program, DFHEIP.

The interface modules shown in Table 11 are required for assembler-language, C, COBOL, and PL/I.

Table 11. Programming language interface modules

Language	**Interface module name**
Assembler	DFHEAI and DFHEAI0
C/VSE, COBOL/VSE and PL/I VSE	DFHELII

The CPI Communications interface module

Each of your CICS application programs that uses the Common Programming Interface for Communications (CPI Communications) must contain an interface to CPI Communications. This takes the form of an interface module used by the CICS high-level programming interface that is common to all the programming languages. The module, DFHCPLC, is found in the VSE/ESA sublibrary PRD1.BASE, and must be link-edited with each application program that uses CPI Communications.

The SAA Resource Recovery interface module

Each of your CICS application programs that uses SAA Resource Recovery must contain an interface to SAA Resource Recovery. This takes the form of an interface module, used by the CICS high-level programming interface, common to all the programming languages. The module, DFHCPLRR, is found in the VSE/ESA sublibrary PRD1.BASE, and must be link-edited with each application program that uses the SAA Resource Recovery facility.

IBM Language Environment for VSE/ESA

Before you can run application programs compiled by an LE/VSE-conforming compiler under CICS, you must have installed LE/VSE.

CICS Transaction Server for VSE/ESA Release 1 supports IBM Language Environment® for VSE/ESA (LE/VSE) (program number 5686-094). CICS manages the interface between its program control component and the LE/VSE run-time environment. Command-level application programs can be written for any of the three LE/VSE-conforming compilers; COBOL/VSE, C/VSE, or PL/I VSE.

The benefits of using LE/VSE include:

- A common run-time library and environment.
- Centralized resources (including initialization, termination, storage management, message handling, condition handling, and math routines.
- Common dumps with debugging information in one place.
- Enhanced interlanguage communication.

  With LE/VSE, you can use the language best suited for particular tasks. For example, PL/I is good for complex calculations, while COBOL is better for report formatting and printing.
- Third-party applications are easily enabled.

  Because LE/VSE provides a well-defined common interface to the operating environment, vendors can write application packages in the language of their choice.

See the *IBM Language Environment for VSE/ESA Concepts Guide* for more information.

Installing LE/VSE

LE/VSE provides a run-time library that establishes a common execution environment for C, COBOL and PL/I application programs. To run programs compiled by an LE/VSE-conforming compiler under CICS, you must have LE/VSE support installed.

The CICS LE/VSE interface is enabled automatically if CICS can:

1. Load the LE/VSE interface module, CEECCICS
2. Successfully call the CEECCICS module to initialize the interface.

LE/VSE initialization takes place during CICS initialization. Before the start of second phase PLT processing the CEECCICS module is loaded and is followed by a partition initialization call to it. Initialization of LE/VSE will fail if the CEECCICS module cannot be loaded, or if something went wrong during the loading of a particular language routine.

Each language that is supported under LE/VSE has a unique LE/VSE interface module that provides the input point for LE/VSE into that language. The module names take the form CEEEV*nnn* where *nnn* represents a number between 000 and 255.

To run application programs compiled using any of the LE/VSE-conforming compilers under CICS, you must ensure that the necessary LE/VSE library routines are defined to CICS. One of the group lists specified on the GRPLIST system initialization parameter must include the group containing the LE/VSE library routines (group name CEE in the sample CSD definitions). LE/VSE supplies the necessary program definitions for the different languages as follows:

To include	Use member	In sublibrary
LE/VSE base	CEECCSD.Z	PRD2.SCEEBASE
COBOL	IGZCCSD.Z	PRD2.SCEECICS
PL/1	IBMCCSD.Z	PRD2.SCEEBASE
C	EDCCCSD.Z	PRD2.SCEEBASE

You can avoid the need to define the LE/VSE library routines to CICS if you exploit program autoinstall. However, if you do use program autoinstall, your program autoinstall control program *must* not itself be compiled using any of the LE/VSE-conforming compilers. If you wish to use a program autoinstall control program compiled with a LE/VSE-conforming compiler, the LE/VSE library routines must be defined to CICS as described above. For further information on the program autoinstall control program, see the *CICS Customization Guide*.

Storage requirements

You must allocate enough storage to run CICS and LE/VSE together. CEECCICS requires approximately 300KB of storage below the 16MB line. Refer to the *IBM Language Environment for VSE/ESA Installation and Customization Guide* for information about storage requirements.

Sample LIBDEF statement

Include the phase libraries containing the CICS LE/VSE run-time modules in the LIBDEF statement in the CICS startup JCL. An example is given in Figure 14. More information about defining your programs can be found in the *IBM Language Environment for VSE/ESA Installation and Customization Guide*.

```
// LIBDEF PHASE,SEARCH=(PRD2.SCEECICS,   1
                       PRD2.SCEEBASE,   2
                       cicslib.sublib,
                       .
                       .
                       .
                       )
```

Figure 14. Example statements to add LE/VSE to the CICS startup JCL

Notes for Figure 14:

1 LE/VSE run-time CICS specific phases. The CICS-specific phases must always precede the base phases.

2 LE/VSE run-time base phases.

Defining the CESE and CESO destinations

The LE/VSE transient data destination, CESE, stores the run-time output produced by LE/VSE, such as messages, dumps, and reports.

The LE/VSE transient destination, CESO, is required for C programs and stores ***stdout*** stream output.

CESE and CESO need to be defined in the destination control table (DCT). The copybook, DFH$DCTR, in the VSE/ESA sublibrary, PRD1.BASE, contains sample definitions for CESE and CESO.

For more information, see the *IBM Language Environment for VSE/ESA Installation and Customization Guide* and the *IBM Language Environment for VSE/ESA Programming Guide*

Preparing to install application programs

This section describes the following points that you should consider about application programs to be installed and the libraries that they are to be installed into:

- Using BMS mapsets in programs
- VSE residence and addressing mode
- Program eligibility for the shared virtual area (SVA)
- Installing programs in load library secondary extents.

This section also outlines the steps that you must complete to install application programs to run under CICS. (See “Installing application programs” on page 41.)

Using BMS mapsets in application programs

This section describes what you must do to use BMS mapsets in application programs.

Before you install an application program to run under CICS, you must:

- Create any BMS mapsets used by the program, as described in Chapter 3, “Installing mapsets and partition sets” on page 15.
- Include the physical mapsets (used by BMS in its formatting activities) in a library that is in the LIBDEF PHASE, SEARCH sublibrary chain for the CICS job.
- Either include the symbolic mapsets (copied into the application programs) in a user copy library or insert them directly into the application program source.

For more information about installing mapsets, see Chapter 3, “Installing mapsets and partition sets” on page 15. For information about writing programs to use BMS services, see the *CICS Application Programming Guide*.

VSE residence and addressing modes

This section describes the effect of the VSE residence and addressing modes on application programs, how you can change the modes, and how you can make application programs permanently resident. An application written to run on VSE/SP™ can run on an VSE/ESA system, if it is link-edited with the AMODE(24) and RMODE(24) options, which are the defaults.

A command-level program compiled with HLASM, C/VSE, COBOL/VSE, or PL/I VSE can reside above the 16MB line, and address areas above that line. The program can contain both CICS and EXEC DLI commands.

Establishing a program's addressing mode

Every program that executes in VSE/ESA is assigned two attributes: an addressing mode (AMODE), and a residency mode (RMODE). AMODE specifies the addressing mode in which your program is designed to receive control. Generally, your program is designed to execute in that mode, although you can switch modes in the program, and have different AMODE attributes for different entry points within a phase. The RMODE attribute indicates where in virtual storage your program can reside. Valid AMODE and RMODE specifications are:

AMODE=24	Specifies 24-bit addressing mode
AMODE=31	Specifies 31-bit addressing mode
AMODE=ANY	Specifies either 24- or 31-bit addressing mode
RMODE=24	Indicates that the module must reside in virtual storage below 16MB. You can specify RMODE=24 for 31-bit programs that have 24-bit dependencies
RMODE=ANY	Indicates that the module can reside anywhere in virtual storage.

If you do not specify any AMODE or RMODE attributes for your program, VSE assigns the system defaults AMODE=24 and RMODE=24. To override these defaults, you can specify AMODE and RMODE in one or more of the following places. Assignments in this list overwrite assignments later in the list.

1. On the linkage editor MODE control statement:
   ```
   MODE  AMODE(31),RMODE(ANY)
   ```
2. In the PARM string on the EXEC statement of the linkage editor job step:
   ```
   // EXEC LNKEDT,PARM='AMODE=31,RMODE=ANY'
   ```
3. On AMODE or RMODE statements within the source code of an assembler program. (You can also set these modes in COBOL for VSE and PL/I for VSE by means of the compiler options. For guidance information about the various compiler options, see the *IBM COBOL for VSE/ESA Programming Guide* and the *IBM PLI for VSE/ESA Programming Guide*.

For information about these modes and the rules that govern their use, see the *VSE/ESA Extended Addressability* manual.

LE/VSE considerations

The LE/VSE Default Run Time Option module for CICS, CEECOPT, contains the value ALL31(ON). If AMODE=24 programs run using LE/VSE, you must change this default to ALL31(OFF). You must also make a corresponding change to the STACK Run Time option to specify the BELOW suboption. You can find more details on LE/VSE Run Time options and how to change them in the *IBM Language Environment for VSE/ESA Installation and Customization Guide*.

CICS address space considerations

Table 12 gives the valid combinations of the AMODE and RMODE attributes and their effects:

Table 12. Valid AMODE and RMODE specifications and their effects

AMODE	RMODE	Residence	Addressing
24	24	Below 16MB line	24-bit mode
31	24	Below 16MB line	31-bit mode
ANY	24	Below 16MB line	31-bit mode
31	ANY	Above 16MB line	31-bit mode

Figure 15 shows linkage editor control statements for a program coded to 31-bit standards.

```
// OPTION CATAL
   PHASE anyname,*
   MODE AMODE(31),RMODE(ANY)
   INCLUDE anytext
   .
   .
   .
// EXEC LNKEDT
/*
```

Figure 15. Linkage editor control statements for 31-bit programs

Making programs permanently resident

If you define a program in the CSD with the resident attribute, RESIDENT(YES), it is loaded on first reference. This applies to programs link-edited with either RMODE(ANY) or RMODE(24). However, be aware that the storage compression algorithm that CICS uses does not remove resident programs.

If there is not enough storage for a task to load a program, the task is suspended until enough storage becomes available. If any of the DSAs get close to being short on storage, CICS frees the storage occupied by programs that are not in use. (For more information about the dynamic storage areas in CICS Transaction Server for VSE/ESA Release 1, see “Storage requirements for a CICS region” on page 300.)

Preparing applications to run in the SVA

Programs written in assembler language, C, COBOL, or PL/I can reside in the VSE shared virtual area (SVA). To do so, they must be read-only and have been link-edited with the SVA option on the PHASE card. Other requirements are as follows:

Assembler
Use the RENT assembler option to highlight obvious errors. However, it is still your responsibility to make sure that the code is fully re-entrant.

C Use the RENT compiler option.

COBOL
Use the RENT compiler option. (The CICS translator generates a CBL statement with the required compiler option RENT, unless you specify the translator option NOCBLCARD.)

PL/I
Do not overwrite STATIC storage. (The CICS translator inserts the required REENTRANT option into the PROCEDURE statement.)

If you want CICS to use modules that you have written to these standards, and installed in the SVA, you must specify USESVACOPY(YES) on the RDO PROGRAM resource definitions.

For information about installing CICS modules in the SVA see Chapter 5, “CICS programs in the VSE shared virtual area” on page 59.

Installing applications in the extended read-only DSA (ERDSA)

Programs that are eligible to reside above the 16MB boundary, and are read-only, can reside in the CICS extended read-only DSA (ERDSA). Therefore, to be eligible for the ERDSA, programs must be:

- Properly written to read-only standards
- Written to 31-bit addressing standards
- Link-edited with the SVA attribute and the RMODE(ANY) residency attribute.

Programs that are ***not*** eligible to reside above the 16MB boundary, and are read-only, can reside in the CICS read-only DSA (RDSA) below the 16MB boundary. Therefore, to be eligible for the RDSA, programs must be:

- Properly written to read-only standards
- Link-edited with the SVA attribute.

Note: When you are running CICS with RENTPGM=PROTECT specified as a system initialization parameter, the RDSAs are allocated from key-0 read-only storage.

Programs link-edited with the SVA and RMODE(ANY) attributes are automatically loaded by CICS into the ERDSA.

ERDSA requirements for the specific languages are described in the following sections.

Installing assembler-language programs in the ERDSA

For assembler-language programs that you want CICS to install in the ERDSA, you should assemble and link-edit them with the following options:

1. The RENT assembler option
2. The linkage-editor SVA attribute
3. The RMODE(ANY) residency mode.

Note: If you specify these options, you must ensure that the program is truly read-only (that is, does not modify itself in any way—for example, by writing to static storage), otherwise storage exceptions will occur. The program must also be written to 31-bit addressing standards. See the *CICS Problem Determination Guide* for some possible causes of storage protection exceptions in programs resident in the ERDSA.

Installing C programs in the ERDSA

For C programs that you want CICS to load into the ERDSA, you should compile and link-edit them with the following options:

1. The C/VSE RENT compiler option.
2. The linkage-editor SVA attribute.
3. The RMODE(ANY) residency mode.

Installing COBOL programs in the ERDSA

COBOL programs are automatically eligible for the ERDSA because:

- If you use the translator option, CBLCARD (the default), the required compiler option, RENT, is included automatically on the CBL statement generated by the CICS translator. If you use the translator option, NOCBLCARD, you must specify the RENT option by using the compiler customization macro, IGYCOPT.
- The COBOL/VSE compiler automatically generates code that conforms to read-only and 31-bit addressing standards.

However, you must specify the SVA and RMODE(ANY) attributes to the linkage-editor.

Installing PL/I programs in the ERDSA

PL/I programs are generally eligible for the ERDSA, provided they do not modify static storage. The following requirements are enforced, either by CICS or PL/I:

- The required REENTRANT option is included automatically, by the CICS translator, on the PL/I PROCEDURE statement.
- The PL/I VSE compiler automatically generates code that conforms to 31-bit addressing standards.

However, you must specify the SVA attribute to the linkage-editor.

Installing application programs

This section outlines the steps that you must complete to install application programs to run under CICS. If you want to use your own JCL to install application programs, see “Using your own job streams” on page 56.

1. Translate the program source code, turning CICS commands into code understood by the compiler.

- No translator step is needed for programs that do not use CICS commands and which are only invoked by a running transaction (and never directly by CICS task initiation).
- CICS command-level programs that access DL/I services through either the DL/I CALL or EXEC DLI interfaces must be translated.

2. Compile or assemble the translator output to produce object code.
3. Link-edit the object module to produce a phase which you store in an application library specified in the LIBDEF search chain in the CICS startup job stream.
4. Create RDO TRANSACTION resource definition entries for any transactions that invokes the program.
5. Do one of the following:
   - If you are using program autoinstall, ensure that the autoinstall user-replaceable module can correctly install a resource definition for the program.
   - If you are not using program autoinstall, create a RDO PROGRAM resource definition for the program.
6. Install resource definition entries. Either use the CEDA INSTALL COMMAND or add the appropriate resource definition groups to one of the group lists specified on the GRPLIST system initialization parameter.

Macro-level application programs

Macro-level programs are not supported in CICS Transaction Server for VSE/ESA Release 1. All such application programs must be recoded to command-level programming standards.

If you have macro-level application programs that you want to run on a CICS Transaction Server for VSE/ESA Release 1 system, you must convert them to command-level. Furthermore, references to the CSA or to the TCA are not allowed. You can specify YES for the system initialization parameter DISMACP to cause CICS to disable any transaction whose program invokes a CICS macro or references the CSA or the TCA.

IBM provides two tools, one to help you identify prohibited calls in application programs, and one to help you convert your macro-level applications to command-level:

- **DFHMSCAN**

  The macro-level scan utility, DFHMSCAN, scans a CICS library for macro calls and references to TCA and CSA. It lists any calls found in a report which you can use to help you locate the statements requiring change.

- **The Application Migration Aid (AMA)**

  The Application Migration Aid converts COBOL or assembler-language macro-level source programs to command-level. Most macro-level code is converted directly, and diagnostics are provided where the intent of the input program is not clear. AMA is supplied in VSE/ESA sublibrary PRD1.BASE, and is described in the *CICS Application Migration Aid Guide*.

Installing assembler-language application using SYSIPT and SYSPCH

The sample job in Figure 16 shows the job control statements needed for an assembler-language application program using disk as intermediate storage for output from the translator.

```
// JOB ASMPROG
* -------------------------------------------------
* Install an assembler-language application program
* -------------------------------------------------
// DLBL   IJSYSPH,'asm.translation',0
// EXTENT SYSPCH,,1,0,rtrk,ntrks                 1
ASSGN SYSPCH,DISK,VOL=volid,SHR
// DLBL cicsusr,'name.name',0,SD
// EXTENT,volid,1,0,rtrk,ntrks
*  The library search chain                      2
// LIBDEF  *,SEARCH=(cicsusr.source,PRD1.BASE)
// LIBDEF PHASE,CATALOG=cicsusr.programs
* --------------------------------
* STEP 1: Translate program source
* --------------------------------
// EXEC   DFHEAP1$,PARM='CICS,...,...,'        3
          .
          assembler-language source statements
          .
/*
* ----------------------------------
* STEP 2: Assemble translated source
* ----------------------------------
CLOSE SYSPCH,cuu
IF $RC > 4 THEN
GOTO EOJT
// DLBL   IJSYSIN,'asm.translation',0
// EXTENT SYSIPT
ASSGN SYSIPT,DISK,VOL=volid,SHR
// OPTION  CATAL,NODECK,SYM,ERRS
   PHASE programname,*
   INCLUDE DFHEAI                                4
// EXEC ASMA90,SIZE=ASMA90,PARM='SIZE(MAX-200K,ABOVE)'
CLOSE  SYSIPT,SYSRDR
IF $RC > 4 THEN
GOTO EOJT
* -----------------------------------
* STEP 3: Link-edit assembled program
* -----------------------------------
// EXEC LNKEDT
/. EOJT
/&
// JOB RESET
ASSGN SYSIPT,SYSRDR   IF 1A93D, CLOSE SYSIPT,SYSRDR
ASSGN SYSPCH,cuu      IF 1A93D, CLOSE SYSPCH,cuu
/&
```

Figure 16. Installing an assembler-language application using SYSPCH and SYSIPT

Notes for Figure 16 on page 43:

1 In JOB ASMPROG, `rtrk` is the relative track number for the start of the extent (or relative block number if you are using an FBA device). `ntrks` is the number of tracks allocated for the extent (or number of blocks allocated for the extent if you are using an FBA device).

2 Code the LIBDEF search chain to include all the library names needed to assemble your application program.

3 You can code the translator options (XOPTS) either on the PARM parameter of the EXEC job control statement as shown, or on an *ASM statement preceding the assembler-language source program.

See the *CICS Application Programming Guide* for further guidance information about the translator options.

4 The INCLUDE statement for DFHEAI must follow immediately after the PHASE statement and before the EXEC ASMA90 statement. DFHEAI and DFHEAI0 are supplied in the VSE/ESA sublibrary, PRD1.BASE.

Installing assembler-language applications using IESINSRT

You can use the IESINSRT utility program as an alternative to using ASSGN statements for SYSPCH and SYSIPT to install a command-level assembler-language application program, as shown in in Figure 17. This allows the job to run in both static and dynamic partitions.

```
* $$ JOB JNM=ASMPROG1,CLASS=x
* $$ LST CLASS=y
* $$ PUN DISP=I,CLASS=x,JNM=ASMPROG2                       1
// JOB ASMPROG1   Build stage 2 job on RDR
* ------------------------------------------------------------
* Install an Assembler language application program
* ------------------------------------------------------------
// ON $CANCEL OR $ABEND GOTO ABEND1                        2
// ASSGN SYSIPT,SYSRDR
// EXEC IESINSRT                                           3
$ $$ LST CLASS=y
// JOB ASMPROG2   Run stage 2 compile and lnkedt
* ------------------------------------------------------------
* Step 1: Compile translated source
* ------------------------------------------------------------
// DLBL CICSUSR,'name.name',0,SD
// EXTENT ,volid
// LIBDEF *,SEARCH=(cicsusr.source,PRD1.BASE)
// LIBDEF PHASE,CATALOG=cicsusr.programs
// OPTION CATAL
  PHASE programname,*
  INCLUDE DFHEAI
// EXEC ASMA90,SIZE=ASMA90,PARM='SIZE(MAX-200K,ABOVE)'
* $$ END
```

Figure 17 (Part 1 of 2). Installing an assembler-language application using IESINSRT - Job 1

```
* ------------------------------------------------------------
* Step 2: Translate program source
* ------------------------------------------------------------
// ON $CANCEL OR $ABEND GOTO ABEND2                           2
// DLBL CICSUSR,'name.name',0,SD
// EXTENT ,volid
// LIBDEF *,SEARCH=(cicsusr.source,PRD1.BASE)
// EXEC DFHEAP1$,PARM='CICS,....,....'
   .
   Assembler source statements
   .
/*
// IF $RC LE 4 THEN
GOTO TRANSOK
/. ABEND2
// EXEC IESINSRT                                              4
                  ***** TRANSLATE STEP FAILED *****
* $$ END
/. TRANSOK
// EXEC IESINSRT
/*
// IF $RC GT 4 THEN
GOTO EOJT
* ------------------------------------------------------------
* Step 3: Link-edit compiled program
* ------------------------------------------------------------
// EXEC LNKEDT
/. EOJT
* $$ END
/*
/. ABEND1
// EXEC IESINSRT
/*
#&
$ $$ EOJ
* $$ END
/*
/&
* $$ EOJ
```

Figure 17 (Part 2 of 2). Installing an assembler-language application using IESINSRT - Job 1

Notes for Figure 17 on page 44:

1 All cards punched by job ASMPROG1 are placed back on the RDR queue as class 'x' with the POWER job name of ASMPROG2.

2 The ON statement ensures that the trailer JCL is written out even if an abend occurs.

3 IESINSRT reads cards from SYSIPT until it reads * $$ END. It translates column 1 $ to *, # to / and punches the card out to SYSPCH. Note that only POWER statements and the VSE /& need to be altered.

4 If the translator fails, an erroneous language statement is punched out to force a compiler error.

The resultant ASMPROG2 job looks as shown in Figure 18 on page 46.

```
* $$ JOB JNM=ASMPROG2,CLASS=x
* $$ LST CLASS=y
// JOB ASMPROG2   Run stage 2 compile and lnkedt
* ---------------------------------------------------------
* Step 1: Compile translated source
* ---------------------------------------------------------
// DLBL CICSUSR,'name.name',0,SD
// EXTENT ,volid
// LIBDEF *,SEARCH=(cicsusr.source,PRD1.BASE)
// LIBDEF PHASE,CATALOG=cicsusr.programs
// OPTION CATAL
  PHASE programname,*
  INCLUDE DFHEAI
// EXEC ASMA90,SIZE=ASMA90,PARM='SIZE(MAX-200K,ABOVE)'
/*
// IF $RC GT 4 THEN
GOTO EOJT
* ---------------------------------------------------------
* Step 2: Link-edit compiled program
* ---------------------------------------------------------
// EXEC LNKEDT
/. EOJT
/*
/&
* $$ EOJ
```

Figure 18. Installing an assembler-language application using IESINSRT - Job 2

Installing COBOL application programs

CICS Transaction Server for VSE/ESA Release 1 supports LE/VSE, and the associated full 31-bit compiler and run-time support for COBOL.

For more information, see “IBM Language Environment for VSE/ESA” on page 35.

Installing a COBOL/VSE program using SYSPCH and SYSIPT

Figure 19 on page 47 shows an sample job to install a COBOL/VSE application program using ASSGN statements for SYSPCH and SYSIPT. Using a VSE virtual disk for IJSYSPCH will improve the performance of the job.

```
// JOB LECOBPRG
* ----------------------------------------------------------------------
* Install a COBOL/VSE application program
* ----------------------------------------------------------------------
// DLBL IJSYSPH,'lecobol.translation',0
// EXTENT SYSPCH,,1,0,rtrk,ntrks                            1
ASSGN SYSPCH,DISK,VOL=volid,SHR
// DLBL cicsusr,'name.name',0,SD
// EXTENT volid,1,0,rtrk,ntrks
* The library search chain
// LIBDEF *,SEARCH=(cicsusr.source,PRD2.PROD,
                PRD2.SCEEBASE,PRD1.BASE)                    2
// LIBDEF PHASE,CATALOG=cicsusr.programs
* --------------------------------
* STEP 1: Translate program source
* --------------------------------
// EXEC DFHECP1$,PARM='XOPTS'(COBOL3,CICS)'                 3
 CBL XOPTS(COBOL3,APOST),APOST                              4
   .
   COBOL source statements
   .
/*
* ---------------------------------
* STEP 2: Compile translated source
* ---------------------------------
CLOSE SYSPCH,cuu
IF $RC > 4 THEN
GOTO EOJT
// DLBL IJSYSIN,'lecobol.translation',0
// EXTENT SYSIPT
ASSGN SYSIPT,DISK,VOL=volid,SHR
// OPTION CATAL
  PHASE programname,*
  INCLUDE DFHELII                                           5
// EXEC IGYCRCTL,SIZE=IGYCRCTL,PARM='SZ(MAX),...'                     *
CLOSE SYSIPT,SYSRDR
IF $RC > 4 THEN
GOTO EOJT
* ----------------------------------
* STEP 3: Link-edit compiled program
* ----------------------------------
// EXEC LNKEDT                                              6
/. EOJT
/&
// JOB RESET
* IF THIS FAILS, CLOSE SYSIPT,SYSRDR
ASSGN SYSIPT,SYSRDR
* IF THIS FAILS, CLOSE SYSPCH,cuu
ASSGN SYSPCH,cuu
/&
* $$ EOJ
```

Figure 19. Installing a COBOL/VSE application using SYSIPT and SYSPCH

Notes for Figure 19:

1 `rtrk` is the relative track number for the start of the extent (or relative number of blocks if you are using an FBA device). `ntrks` is the number of tracks allocated for the extent (or number of blocks allocated for the extent if you are using an FBA device).

2 Code the LIBDEF search chain to include all the library names applicable to your installation that your COBOL compilations need. PRD2.PROD is the name of the library containing the COBOL/VSE compiler.

3 You can code the translator options (XOPTS) either on the PARM parameter of the EXEC job control statement as shown, or on a CBL statement preceding the COBOL source program. See the *CICS Application Programming Guide* for details of the translator options.

Specify the LIB option for the compilation step so that any COPY statements in the source (for example, symbolic description maps) can be processed correctly.

Note that the CICS Transaction Server for VSE/ESA translator does not generate COPY statements.

4 The APOST option is included in the CBL statement in this job for compatibility with the sample programs supplied with CICS/VSE®. (The supplied sample COBOL programs are written using the APOST option.)

5 An INCLUDE statement for DFHELII must follow immediately after the PHASE statement, and before the EXEC IGYCRCTL statement. DFHELII is supplied in the VSE/ESA sublibrary, PRD1.BASE.

6 You can ignore weak external references (WXTRN) unresolved by the linkage editor, and also the associated messages about unresolved address constants.

Installing a COBOL/VSE program using IESINSRT

As an alternative to using ASSGN statements for SYSPCH and SYSIPT to install a COBOL/VSE application, you can use the IESINSRT utility program as shown in Figure 20. The job may then run in a static or dynamic partition.

```
* $$ JOB JNM=LECOBPRG,CLASS=x
* $$ LST CLASS=y
* $$ PUN DISP=I,CLASS=x,JNM=LECOBPR2                          1
// JOB LECOBPRG   Build stage 2 job on RDR
* -------------------------------------------------------
* Install a COBOL/VSE application program
* -------------------------------------------------------
// ON $CANCEL OR $ABEND GOTO ABEND1                           2
// ASSGN SYSIPT,SYSRDR
// EXEC IESINSRT                                               3
$ $$ LST CLASS=y
// JOB LECOBPR2   Run stage 2 compile and lnkedt
* -------------------------------------------------------
* STEP 1: Compile translated source
* -------------------------------------------------------
// DLBL CICSUSR,'name.name',0,SD
// EXTENT ,volid
// LIBDEF *,SEARCH=(cicsusr.source,PRD2.PROD,
               PRD2.SCEEBASE,PRD1.BASE)
// LIBDEF PHASE,CATALOG=cicsusr.programs
// OPTION CATAL
  PHASE programname,*
  INCLUDE DFHELII
// EXEC IGYCRCTL,SIZE=IGYCRCTL,PARM='SZ(MAX),...'
* $$ END
* -------------------------------------------------------
* STEP 2: Translate program source
* -------------------------------------------------------
// ON $CANCEL OR $ABEND GOTO ABEND2                           2
// DLBL CICSUSR,'name.name',0,SD
// EXTENT ,volid
// LIBDEF *,SEARCH=(cicsusr.source,PRD2.COMP110,PRD2.PROD110,
               PRD2.SCEEBASE,PRD1.BASE)
// EXEC DFHECP1$,PARM='XOPTS(COBOL3,CICS)'
 CBL XOPTS(COBOL3,APOST),APOST
   .
   COBOL source statements
   .
/*
```

Figure 20 (Part 1 of 2). Installing a COBOL/VSE application using IESINSRT - Job 1

```
// IF $RC LE 4 THEN
GOTO TRANSOK
/. ABEND2
// EXEC IESINSRT                                          4
                  ***** TRANSLATE STEP FAILED *****
* $$ END
/*
/. TRANSOK
// EXEC IESINSRT
/*
// IF $RC GT 4 THEN
GOTO EOJT
* ----------------------------------------------------------
* STEP 3: Link-edit compiled program
* ----------------------------------------------------------
// EXEC LNKEDT
/. EOJT
* $$ END
/. ABEND1
// EXEC IESINSRT
/*
#&
$ $$ EOJ
* $$ END
/*
/&
* $$ EOJ
```

Figure 20 (Part 2 of 2). Installing a COBOL/VSE application using IESINSRT - Job 1

Notes for Figure 20 on page 49:

1 All cards punched by job LECOBPRG are placed back on the RDR queue as class 'x' with the POWER job name of LECOBPR2.

2 The ON statement ensures that the trailer JCL is written out even if an abend occurs.

3 IESINSRT reads cards from SYSIPT until it reads * $$ END. It translates column 1 $ to *, # to / and punches the card out to SYSPCH. Note that only POWER statements and the VSE /& need to be altered.

4 If the translator fails, an erroneous language statement is punched out to force a compiler error.

The resultant LECOBPR2 job looks as shown in Figure 21 on page 51.

```
* $$ JOB JNM=LECOBPR2,CLASS=X
* $$ LST CLASS=y
// JOB LECOBPR2   Run stage 2 compile and lnkedt
* -----------------------------------------------------------
* STEP 1: Compile translated source
* -----------------------------------------------------------
// DLBL CICSUSR,'name.name',0,SD
// EXTENT ,volid
// LIBDEF *,SEARCH=(cicsusr.source,PRD2.PROD,
               PRD2.SCEEBASE,PRD1.BASE)
// LIBDEF PHASE,CATALOG=cicsusr.programs
// OPTION CATAL
  PHASE programname,*
  INCLUDE DFHELII
// EXEC IGYCRCTL,SIZE=IGYCRCTL,PARM='SZ(MAX),...'
     .
     Translated source code
     .
/*
// IF $RC GT 4 THEN
GOTO EOJT
```

Figure 21 (Part 1 of 2). Installing a COBOL/VSE application using IESINSRT - Job 2

```
* -----------------------------------------------------------
* STEP 2: Link-edit compiled program
* -----------------------------------------------------------
// EXEC LNKEDT
/. EOJT
/*
/&
* $$ EOJ
```

Figure 21 (Part 2 of 2). Installing a COBOL/VSE application using IESINSRT - Job 2

Installing PL/I application programs

CICS Transaction Server for VSE/ESA Release 1 supports LE/VSE, and the associated full 31-bit compiler and run-time support for PL/I VSE.

For more information, see “IBM Language Environment for VSE/ESA” on page 35.

Installing a PL/I VSE program using SYSPCH and SYSIPT

Figure 22 shows an sample job to install a PL/I VSE application program using ASSGN statements for SYSPCH and SYSIPT. Using a VSE virtual disk for IJSYSPCH will improve the performance of the job.

```
// JOB LEPLIPRG
*-------------------------------------------------------------------
* Install a PL/I VSE application program
*-------------------------------------------------------------------
*  TRANSLATOR STEP
// DLBL IJSYSPH,'lepli.translation',0
// EXTENT SYSPCH,,1,0,rtrk,ntrks                               1
ASSGN SYSPCH,DISK,VOL=volid,SHR
// DLBL cicsusr,'name.name',0,SD
// EXTENT volid,1,0,rtrk,ntrks
* The library search chain
// LIBDEF *,SEARCH=(cicsusr.source,PRD2.PROD,
                PRD2.SCEEBASE,PRD1.BASE)                       2
// LIBDEF PHASE,CATALOG=cicsusr.programs
*----------------------------------
* STEP 1: Translate program source
*----------------------------------
// EXEC DFHEPP1$,PARM='CICS,....,.....'                        3
*PROCESS INCLUDE;                                               4
   .
   PL/I source statements
   .
/*
*-----------------------------------
* STEP 2: Compile translated source
*-----------------------------------
CLOSE SYSPCH,cuu
IF $RC > 4 THEN
GOTO EOJT
// DLBL IJSYSIN,'lepli.translation',0
// EXTENT SYSIPT
ASSGN SYSIPT,DISK,VOL=volid,SHR
// OPTION CATAL
// OPTION NODUMP
  PHASE programname,*
  INCLUDE DFHELII                                              5
// EXEC IEL1AA,SIZE=500K,PARM='LIST'
CLOSE SYSIPT,SYSRDR
IF $RC > 4 THEN
GOTO EOJT
*------------------------------------
* STEP 3: Link-edit compiled program
*------------------------------------
// EXEC LNKEDT                                                 6
/. EOJT
/*
/&
// JOB RESET
* IF THIS FAILS, CLOSE SYSIPT,SYSRDR
ASSGN SYSIPT,SYSRDR
* IF THIS FAILS, CLOSE SYSPCH,cuu
ASSGN SYSPCH,cuu
/&
* $$ EOJ
```

Figure 22. Installing a PL/I VSE application using SYSPCH and SYSIPT

Notes for Figure 22 on page 52:

1 `rtrk` is the relative track number for the start of the extent (or relative number of blocks if you are using an FBA device). `ntrks` is the number of tracks allocated for the extent (or number of blocks allocated for the extent if you are using an FBA device).

2 Code the LIBDEF search chain to include all the library names that your PL/I compilation needs. PRD2.PROD is the name of the library containing the PL/I VSE compiler.

3 You can code the translator options (XOPTS) either on the PARM parameter of the EXEC job control statement as shown, or on a *PROCESS statement preceding the PL/I source program.

See the *CICS Application Programming Guide* for details of the translator options.

4 You must specify the INCLUDE or MACRO option for the compilation step so that any %INCLUDE statements in the source (for example, symbolic description maps) are processed correctly. Note that the PL/I translator in CICS Transaction Server for VSE/ESA does not generate %INCLUDE statements.

5 The INCLUDE statement for DFHELII must follow immediately after the PHASE statement and before the EXEC IEL1AA statement. DFHELII is supplied in the VSE/ESA sublibrary, PRD1.BASE.

6 You can ignore weak external references (WXTRN) unresolved by the linkage editor, and also the associated messages about unresolved address constants.

Note that the CICS translator no longer generates PL/I CALLs to prompt messages from the PL/I compiler warning of argument lists that are too short.

The declarations of the CICS entry-point names are no longer required or generated by the translator. This change affects existing application programs only if they consist of included segments that do not start with valid PROCEDURE statements. If such a program is retranslated, each included segment must also be retranslated. This is because previously translated included segments refer to the entry-point names that were previously declared in the outer-level procedure.

You do not have to take any action unless you retranslate the outer-level procedure.

Installing a PL/I VSE program using IESINSRT

As an alternative to using ASSGN statements for SYSPCH and SYSIPT, you can use the IESINSRT utility program as shown in Figure 23 on page 54. The job may then run in a static or dynamic partition.

```
* $$ JOB JNM=LEPLI1,CLASS=x
* $$ LST CLASS=y
* $$ PUN DISP=I,CLASS=x,JNM=LEPLI2                          1
// JOB LEPLI1   Build stage 2 job on RDR
* -------------------------------------------------------
* Install a PL/I VSE application program
* -------------------------------------------------------
// ON $CANCEL OR $ABEND GOTO ABEND1                         2
// ASSGN SYSIPT,SYSRDR
// EXEC IESINSRT                                            3
$ $$ LST CLASS=y
// JOB LEPLI2   Run stage 2 compile and lnkedt
* -------------------------------------------------------
* Step 1: Compile translated source
* -------------------------------------------------------
// DLBL cicsusr,'name.name',0,SD
// EXTENT ,volid
// LIBDEF *,SEARCH=(cicsusr.source,PRD2.PROD,
               PRD2.SCEEBASE,PRD1.BASE)
// LIBDEF PHASE,CATALOG=cicsusr.programs
// OPTION CATAL
  PHASE programname,*
  INCLUDE DFHELII
// EXEC IEL1AA,SIZE=500K,PARM='LIST'
* $$ END
* -------------------------------------------------------
* Step 2: Translate program source
* -------------------------------------------------------
// ON $CANCEL OR $ABEND GOTO ABEND2                         2
// DLBL cicsusr,'name.name',0,SD
// EXTENT ,volid
// LIBDEF *,SEARCH=(cicsusr.source,PRD2.PROD,
               PRD2.SCEEBASE,PRD1.BASE)
// EXEC DFHEPP1$,PARM='CICS,....,....'
*PROCESS INCLUDE;
   .
   PL/I source statements
   .
/*
// IF $RC LE 4 THEN
GOTO TRANSOK
/. ABEND2
// EXEC IESINSRT                                            4
                  ***** TRANSLATE STEP FAILED *****
* $$ END
/*
/. TRANSOK
// EXEC IESINSRT
/*
// IF $RC GT 4 THEN
GOTO EOJT
```

Figure 23 (Part 1 of 2). Installing a PL/I VSE application using IESINSRT - Job 1

```
* -----------------------------------------------------------
* Step 3: Link-edit compiled program
* -----------------------------------------------------------
// EXEC LNKEDT
/. EOJT
* $$ END
/. ABEND1
// EXEC IESINSRT
/*
#&
$ $$ EOJ
* $$ END
/*
/&
* $$ EOJ
```

Figure 23 (Part 2 of 2). Installing a PL/I VSE application using IESINSRT - Job 1

Notes for Figure 23 on page 54:

1 All cards punched by job LEPLI1 are placed back on the RDR queue as class 'x' with the Power jobname of LEPLI2.

2 The ON statement ensures that the trailer cards are written out even if an abend occurs.

3 IESINSRT reads cards from SYSIPT until it reads * $$ END. It translates column 1 $ to *, # to / and punches the card out to SYSPCH. Note that only POWER statements and the VSE /& need to be altered.

4 If the translator fails, an erroneous language statement is punched out to force a compiler error.

The resultant LEPLI2 job looks as shown in Figure 24 on page 56.

```
* $$ JOB JNM=LEPLI2,CLASS=x
* $$ LST CLASS=y
// JOB LEPLI2   Run stage 2 compile and lnkedt
* -------------------------------------------------------------
* Step 1: Compile translated source
* -------------------------------------------------------------
// DLBL cicsusr,'name.name',0,SD
// EXTENT ,volid
// LIBDEF *,SEARCH=(cicsusr.source,PRD2.PROD,
               PRD2.SCEEBASE,PRD1.BASE)
// LIBDEF PHASE,CATALOG=cicsusr.programs
// OPTION CATAL
  PHASE programname,*
  INCLUDE DFHELII
// EXEC IEL1AA,SIZE=500K,PARM='LIST'
     .
     Translated source code
     .
/*
// IF $RC GT 4 THEN
GOTO EOJT
* -------------------------------------------------------------
* Step 2: Link-edit compiled program
* -------------------------------------------------------------
// EXEC LNKEDT
/. EOJT
/*
/&
* $$ EOJ
```

Figure 24. Installing a PL/I VSE application using IESINSRT - Job 2

Using your own job streams

This section summarizes the important points about the translator and each of the main categories of program.

Translator requirements

The CICS translator requires a minimum of 256KB of virtual storage. You may need to use the translator options CICS and DLI. (See the following sections.) For COBOL/VSE programs, use the COBOL3 translator option.

Online programs that use EXEC CICS or EXEC DLI commands

1. Always use the translator option CICS. If the program issues EXEC DLI commands, use the translator option DLI.
2. The linkage editor input must include the correct interface module ***before*** the object deck. Therefore, place an INCLUDE statement for the interface module before the object deck. Also put an ENTRY statement after all the INCLUDE statements. The interface modules are:

DFHEAI	Assembler
DFHELII	C/VSE, COBOL/VSE, and PL/I VSE

   Note: Assembler programs require an additional module, DFHEAI0. Inclusion of DFHEAI generates a reference to DFHEAI0, and it is automatically included.

3. Place the load module output from the linkage editor in a user-defined application program library.

Online programs that use the CALL DLI interface

1. Specify the translator option CICS, but not the translator option DLI.

   Note: For a program that does not use CICS commands and is only invoked by a running transaction (and never directly by CICS task initiation), no translator step is needed.

2. Place the load module output from the linkage editor in a user-defined application program library.

Defining programs, mapsets, and partition sets to CICS

To be able to use a program that you have installed in one of the load libraries specified in your CICS startup JCL, the program, and any mapsets and partition sets that it uses, must be defined to CICS. To do this, CICS uses the resource definitions MAPSET (for mapsets), PARTITIONSET (for partition sets), and PROGRAM (for programs). You can create and install such resource definitions in any of the following ways:

- CICS can dynamically create, install, and catalog a definition for the program, mapset, or partition set when it is first loaded, by using the autoinstall for programs function.
- You can create a specific resource definition for the program, mapset, or partition set and install that resource definition in your CICS region. You can install resource definitions in either of the following ways:
  – At CICS initialization, by including the resource definition group in a group list specified on the GRPLIST system initialization parameter
  – While CICS is running, by the CEDA INSTALL command.

For information about defining programs to CICS, see the *CICS Resource Definition Guide*.

Chapter 5. CICS programs in the VSE shared virtual area

This chapter looks at the VSE shared virtual area (SVA), and the programs that you can place in it.

Information can be found under the following headings:

- "Benefits of using the SVA"
- "Modules that must reside in the SVA"
- "Modules that are SVA-eligible" on page 60
- "Programs that are not SVA-eligible" on page 61
- "Installing and using the SVA" on page 61.

Benefits of using the SVA

The benefits you derive from using the SVA are:

- **Integrity**

  The SVA is key-0 protected, and so all modules in the SVA are automatically protected from being overwritten by other programs, such as CICS applications. This integrity feature is equally applicable to a single CICS system within the processor.

- **Performance**

  Performance can be improved and the demand for real storage reduced if you use the SVA for program modules. If more than one copy of the same release of CICS is running in multiple address spaces of the same processor, each address space requires access to the CICS nucleus modules. These modules may either be loaded into each of the address spaces or shared in the SVA. If they are shared in the SVA, this can reduce the working set and, therefore, the demand for real storage (paging).

Modules that must reside in the SVA

Some modules **must** reside in the SVA. Table 13 summarizes these modules.

Table 13 (Page 1 of 2). Mandatory SVA modules

Module	**Description**
DFHCDDAN	The CPC dead data anchor block DFHCDDAN must be loaded into the SVA in a dual-CPC XRF environment. Although DFHCDDAN is not read-only, it must still be placed in the SVA.
DFHCSEOT	CICS EOJ cleanup routine. Performs cleanup for interregion communication and shared data tables. CICS invokes DFHCSEOT during its end-of-job processing. If it cannot find DFHCSEOT in the SVA, cleanup does not take place and the interregion and shared data tables environments become unusable.
DFHCSVC	CICS SVC 1 2
DFHDSPEX	The CICS post exit stub 1

Table 13 (Page 2 of 2). Mandatory SVA modules

Module	Description
DFHDTSAN	Shared data tables anchor block.
DFHDTSVC	Shared data tables SVC module.
DFHIRP	Interregion communication program. If you are using MRO, DFHIRP must be used from the SVA for integrity reasons.
DFHSCTE	Subsystem control table extension. If you are using MRO, DFHSCTE provides an anchor for the interregion communication control block structure. Although DFHSCTE is not read-only, it must still be placed in the SVA.

Notes for Table 13 on page 59:

1 Can only be used from the SVA, and must be installed into the SVA before CICS can be started.

2 To communicate using MRO, all CICS regions in the same VSE image must use the latest levels of the modules DFHCSVC and DFHIRP in the SVA. If, when opening interregion communication, a CICS Transaction Server for VSE/ESA Release 1 region detects that the level of DFHIRP is infact a lower level version, it issues message DFHIR3799 and interregion communication fails to open.

Running different levels of CICS

To avoid problems that can arise when using different levels of CICS code within the same VSE system, it is recommended that only the **required** CICS Transaction Server for VSE/ESA Release 1 SVA-eligible modules are loaded into the SVA.

Modules that are SVA-eligible

Besides those CICS modules that must reside in the SVA, other CICS modules and user application program modules can also be used from the SVA.

CICS modules

A CICS module optionally installed in the SVA (that is, a module that is not mandatory for inclusion in the SVA) can be used only by the release of CICS to which it relates.

Those CICS modules that reside above the 16MB line (for example, the CICS initialization program, DFHSIP31) are loaded above the line. Such modules can also be installed in the extended SVA (ESVA).

CICS modules eligible to be used from the SVA are listed in Appendix B, “CICS modules eligible for the VSE Shared Virtual Area (SVA)” on page 315. Use this appendix to help you choose those CICS modules you want to install in the SVA.

For more information on LE/VSE modules to be installed in the SVA, see the *IBM Language Environment for VSE/ESA Installation and Customization Guide*.

User application programs

User application programs can be used from the SVA if they are read-only and:

- Written in COBOL, do not overwrite WORKING STORAGE, and are compiled using COBOL/VSE. (The CICS translator generates a CBL statement with the required compiler option, RENT.)
- Written in PL/I, do not overwrite STATIC storage, and compiled using PL/I VSE. (The CICS translator inserts the required REENTRANT option into the PROCEDURE statement.)
- Written in C, compiled using C/VSE, using the RENT option.
- Written in assembler language and assembled with the RENT option.

Programs that are not SVA-eligible

The following programs are not SVA-eligible:

- BMS mapsets and programs defined with RELOAD=YES
- Non-reentrant programs

System initialization parameters

Code YES or NO on the SVA system initialization parameter to indicate whether any CICS management modules can be used from the SVA.

Installing and using the SVA

The following sections describe how to install and use the SVA. It considers subjects such as space requirements, how to load the SVA, how to use the SVA during CICS execution, and how to update phases in the SVA.

Allocating space in the SVA

Allocate space in the SVA for CICS modules and their system directory list (SDL) entries at IPL time using the VSE SVA command. See the *VSE/ESA System Control Statements* manual for more information about this command. This space is additional to any required for your own phases. See Appendix B, “CICS modules eligible for the VSE Shared Virtual Area (SVA)” on page 315 for the approximate sizes of the SVA-eligible CICS modules. These sizes are taken from a directory listing of the pregenerated system. For more accurate space estimates, consult the library listings at the time of installation, and verify the space estimates against actual space layout when you have selected the modules for residency in the SVA.

When calculating the space required, note that VSE loads phases into the SVA (starting at the low end) contiguously on double-word boundaries, in alphanumeric sequence regardless of any ordering within a SET SDL command.

For example, if your current SVA command is as follows:

```
SVA SDL=300,GETVIS=768K,PSIZE=(256K,2000K)
```

and you want to add 80 phases with a space requirement of 256K of SVA-24 storage and 512K of SVA-31 storage *without* having to use existing free space, your new SVA command should specify the following values:

```
SVA SDL=380,GETVIS=768K,PSIZE=(512K,2512K)
```

Loading the SVA

Before you can load a module into the SVA, it must be link-edited as SVA-eligible; that is, with the SVA operand added to the linkage-editor PHASE statement. All CICS modules in the pregenerated system considered to be SVA-eligible are cataloged in this way.

See "Modules eligible for SVA residence" on page 315 for a list of CICS modules eligible for SVA residence.

To load SVA-eligible modules into the SVA, issue a SET SDL command from the background (BG) partition, naming the selected modules. You can issue this command at any time after IPL, but you must issue it ***before*** bringing up any CICS system that uses modules from the SVA.

VSE loads SVA phases from the libraries of the BG partition library search chain. You must specify those libraries that contain SVA-eligible modules in the SEARCH operand of the LIBDEF statement for the BG partition.

Figure 25 is an example of a CICS SVA load list.

```
// JOB CICS SVALOAD
*  CICS for VSE/ESA SVA LOAD LIST
* LIBDEF statement for the CICS system sublibrary
SET SDL
DFHCDDAN,SVA
DFHCSEOT,SVA
DFHCSVC,SVA
DFHDSPEX,SVA
DFHDTSAN,SVA
DFHDTSVC,SVA
DFHIRP,SVA
DFHIRW10,SVA
DFHSCTE,SVA
 .
 .
/*
/&
```

Figure 25. Example of a CICS SVA load operation

Alternatively, you can create an SVA load list module by using the VSE SVALLIST macro. (See the *VSE/ESA Guide to System Functions* for more information about the SVALLIST macro.)

CICS supplies a pregenerated SVA load list module called $SVACICS, in the VSE/ESA sublibrary PRD1.BASE. $SVACICS lists all those modules that **must** reside in the SVA. See "Modules that must reside in the SVA" on page 59 for more information. The source of $SVACICS is called DFH$SVA and is supplied in the VSE/ESA sublibrary PRD1.BASE.

Figure 26 on page 63 shows you how to load the SVA using $SVACICS.

```
// JOB CICS SVALOAD
*
*
SET SDL
LIST=$SVACICS
/*
/&
```

Figure 26. Sample JCL to load the SVA using $SVACICS

Using the SVA during CICS execution

The use of modules installed in the SVA is controlled by CICS system initialization parameters and resource definitions.

Using modules from the SVA

To use any of the CICS modules installed in the SVA, you must:

- Specify the system initialization parameter SVA=YES. CICS then uses the following search order:
  1. VSE System Virtual Area (SVA)
  2. LIBDEF search chain.
- Specify USESVACOPY(YES) in the associated RDO PROGRAM resource definition for any non-nucleus CICS module or user application program.

For each CICS-supplied SVA-eligible module that requires USESVACOPY(YES) specified in its associated RDO PROGRAM resource definition, you must create your own resource definition with USESVACOPY(YES) specified, and use it instead of the CICS-supplied resource definition. This is because you cannot modify the CICS-supplied resource definitions.

For example:

1. Use the DFHCSDUP program to: copy the CICS-supplied resource groups that contain the module definitions to a set of new resource groups.
2. For each module that needs USESVACOPY(YES), change the RDO PROGRAM resource definition in the new copy of the resource groups so that USESVACOPY(YES) is specified.
3. Add the new resource groups to a new group list, at the start of the list.
4. Append the CICS-supplied group list DFHLIST (or your own equivalent of that list) to the end of your new group list.
5. Remove the original copies of the CICS-supplied groups which have been copied and updated.
6. Specify the new group list on the GRPLIST system initialization parameter.
7. Reinitialize the CICS catalogs if you have been using modules which are not in the SVA and which you now wish to use from the SVA.

For more information on the DFHCSDUP off-line utility and examples of its use, see the *CICS Operations and Utilities Guide*.

Note

In the above example, instead of steps 3 and 4, CEDA may be used to:

- Copy the CICS-supplied group list to create a new group list;
- Add the new USESVACOPY(YES) groups to the new group list **in the same place as** the original groups.

Note that:

1. CICS uses eligible modules from the SVA if:
   - You have **not** specified the name of the module on the CICS system initialization parameter PRVMOD;
   - The module has not already been loaded from the LIBDEF search chain concatenation.
2. If CICS cannot find an eligible module in the SVA, it loads the private (non-shared) version from the LIBDEF search chain concatenation into the CICS address space, after first issuing the DFHLD0107I message to warn you that the module is not in the SVA (see “The module-not-found warning message” on page 66 for more information about this message);
3. Program List Tables (PLTs) must be placed in the LIBDEF search chain concatenation. However, before RDO PROGRAM resource definitions for phase one PLTPI programs and PLTSD programs are installed (for example, early in CICS initialization), CICS scans the SVA for those programs, and will issue a DFHLD0107I message for each program it cannot find there.
4. Before RDO PROGRAM resource definitions for global and task-related user exit programs are installed (for example, early in CICS initialization), CICS scans the SVA for those programs, and issues a DFHLD0107I message for each program it cannot find there.

Specifying USESVACOPY(YES)

For every non-nucleus CICS module or user application that you have moved to the SVA (that is, that you have removed from the LIBDEF search chain concatenation), you must ensure that you have specified USERSVACOPY(YES) on the associated RDO PROGRAM resource definition. If you do not do so CICS will be unable to find the module, and may thus fail to start up correctly.

Preventing CICS from using modules installed in the SVA

You can prevent CICS from using modules installed in the SVA by either:

- Specifying the SVA=NO system initialization parameter.

  This prevents CICS from using any modules installed into the SVA; instead, it tries to load them from the LIBDEF search chain concatenation.

  This option is useful if you need to test many SVA-eligible modules before installing them in the SVA. Once you have verified the use of these modules from the SVA, specify the SVA=YES system initialization parameter.

- Specifying the name of the module on the PRVMOD system initialization parameter:

  ```
  PRVMOD={name|(name1,name2,...)}
  ```

This prevents CICS from using the specified modules from the SVA **only** during the run of CICS for which the PRVMOD parameter is specified. This option is useful if you need to test a new version of a module before installing it in place of the copy that is already in the SVA.

You must specify the full name of the module on the PRVMOD parameter, including any suffix (for example, DFHMCP1$). If only one module is named, the parentheses are optional; if there is more than one module name, you must include the parentheses.

The PRVMOD parameter may span input lines, but you must not split module names across input lines because CICS initialization adds a comma at the end of each line that does not already end in a comma. The only validity checking performed on module names is to ensure that they do not exceed eight characters.

You cannot code the PRVMOD parameter in the DFHSIT module; instead, it should be specified on the PARM parameter, in the SYSIPT data stream, or from the system console.

- Specifying USESVACOPY(NO) for a non-nucleus CICS module or user application program on the associated RDO PROGRAM resource definition. This is the default.

  This option allows a more permanent exclusion of an SVA-resident module than the PRVMOD option, which only affects the single run of CICS where PRVMOD is specified.

Verifying modules for the SVA

While verifying new versions of modules to be installed into the SVA, you can instruct a CICS region to use the new versions from the LIBDEF search chain concatenation by using any of the following options:

- The SVA=NO system initialization parameter
- The PRVMOD system initialization parameter
- The USESVACOPY(NO) option of the associated RDO PROGRAM resource definition

For further information about these options, see “Preventing CICS from using modules installed in the SVA” on page 64.

In all cases, you must install the new versions of the modules into a library that is in the LIBDEF search chain concatenation.

After you have verified the use of the modules from the SVA and installed them, you should remove the versions of the modules which are in the LIBDEF search chain concatenation of your CICS startup job.

You may determine whether CICS is loading modules from the SVA or from the LIBDEF search chain concatenation by reviewing the loader section of a system dump (search for “===LD”). The ‘program repertoire’ section shows you a ‘Type’ of ‘SVA’ for modules loaded from the SVA.

The module-not-found warning message

CICS issues a DFHLD0107I message if it searches the SVA for a module which is supposed to be installed there, but fails to find it.

If you encounter this message, check if you have specified USESVACOPY(YES) on the associated RDO PROGRAM resource definition, where applicable.

Updating phases in the SVA

You can update phases in the SVA without having to re-IPL the system. In BG, issue the SET SDL command followed by the phases that you want to reload.

You can only update phases in the SVA if there is enough free space. To check the amount of free space available, issue the following LIBR command:

```
LD SDL
```

You must not, under any circumstances, update the DFHSCTE module in the SVA while any CICS interregion communication is in progress because DFHSCTE is an anchor block for the ISC control block structure.

Updating nucleus modules in the SVA

When you update phases in the SVA, the earlier versions remain in the SVA, but they are no longer addressable from the SDL. However, you should be aware that a running CICS system continues to use the original versions of nucleus modules; that is, those that were in the SVA at the time when the system was initialized.

Updating PPT modules in the SVA

You can issue a CEMT or an EXEC CICS SET PROGRAM NEWCOPY command to replace a PPT module that has been updated in the SVA. The CICS system initialization parameters and resource defintions determine whether the updated version of the module in the SVA is used.

Chapter 6. Accessing DL/I databases using CICS online applications

This chapter describes how to access DL/I VSE databases using CICS online applications.

Information can be found under the following headings:

- “Adding DL/I support” outlines the process for adding DL/I support to your CICS system.
- “The CICS-DL/I installation tester, DFHTDLI” on page 68 describes the CICS-DL/I installation tester program, DFHTDLI.

Adding DL/I support

The process for adding DL/I support to CICS is as follows:

1. Define DL/I databases and application programs during the preparation of CICS tables:
   a. Define a file control table (FCT), and include an FCT entry for each database descriptor (DBD) corresponding to a physical, logical, and index database.
   b. Define the applications that access DL/I databases in the CSD.

   Note

   Note that RDO **cannot** be used to define DL/I databases.

2. Specify DL/I and CICS system table macros for DL/I support as follows:
   a. Define the DL/I VSE application control table (ACT). The ACT is needed to associate online application programs with one or more DL/I databases.
   b. If program isolation is active (or if emergency restart or dynamic transaction backout is to be used with DL/I tasks) assign the DL/I VSE log to the CICS system log. The DL/I VSE log is assigned using the VSE UPSI byte information.

      See page 229 for more information about the DBP system initialization parameter.
   c. If you are using the execution diagnostic facility (EDF) with application programs containing EXEC DLI commands, the DL/I language definition table (DLZHLPI) must be known to CICS. DLZHLPI is a module provided with DL/I VSE.

      DLZHLPI is defined in the CSD in group DFHEDF (which is included in group list DFHLIST).
   d. If you want to capture DL/I run and buffer statistics, you must include the assembler-language DL/I module, DLZSTTL, in your resource definitions. The run and buffer statistics function captures online DL/I system statistics and writes them to the extrapartition destination, CSSL. This data is automatically printed during CICS shutdown only if DLZSTTL is included in the PLT.

Alternatively, printing can be invoked by the CSDE transaction.

Define the assembler-language program DLZSTTL to your CICS system in the CSD using the CEDA DEFINE PROGRAM command (or the DFHCSDUP offline utility DEFINE PROGRAM command).

Define the CSDE transaction in the same group, and then install both DLZSTTL and the CSDE transaction.

Following this, run the CSDE transaction to get DL/I statistics included in the shutdown statistics.

Chapter 21, "DL/I definitions" on page 183 describes the runtime considerations for DL/I.

The CICS-DL/I installation tester, DFHTDLI

DFHTDLI is a command-level assembler-language program that you can use to test a CICS-DL/I installation. Before you can run DFHTDLI, you must generate a DL/I application control table (ACT) to include an entry for the test program. DFHTDLI requires a database to work on. For information about creating databases, see the *DL/I VSE Interactive Resource Definition and Utilities* manual. Further requirements are listed at the end of this description.

The transaction identifier is TDLI, and the sublibrary phase name is DFHTDLI.

DFHTDLI uses Basic Mapping Support to format a 1920-character 3270 display screen, and requests that you build a DL/I CALL on the screen. You can issue any of the DL/I calls, using up to five qualified or unqualified segment search arguments. System calls and calls that include command codes are not supported. DFHTDLI only supports DL/I segments up to a length of 512 bytes. Attempts to access larger segments give unpredictable results.

Invoking DFHTDLI

Invoke DFHTDLI by typing "TDLI" and pressing ENTER. The first panel you see requests the entry of a PSB name. You can enter any PSB name defined in the ACT specified for use by the program. If you don't specify a PSB name (that is, you simply press the ENTER key), DFHTDLI uses the first default PSB defined in the ACT. If the program cannot find the PSB you name, an error message is issued, displaying, in hexadecimal notation, the DL/I error codes returned in TCAFCTR and TCAFCTR+1.

The ENTER panel

The ENTER panel is the first panel returned. It allows a DL/I call to be built to your requirements.

Calls are executed using the DBD name of the first program communication block (PCB) in the PSB unless you specify the name of a DBD on the panel. The program searches 1 through 8 PCBs for the DBD name entered and, if found, uses that PCB for the call. Some syntax editing is carried out on the input data, and any errors found are indicated by a message displayed at the bottom of the screen asking you to correct the error and press the ENTER key.

If you don't enter a function code, the program assumes that a GN call is required on the last PCB used. Pressing the CLEAR key terminates the program at any point.

The GOUT (good output) panel

If the call is successful, the segment requested is returned using the GOUT panel.

The GOUT panel displays the parameters used for the call, the PCB key feedback fully-concatenated key of the path, and the segment requested. If another call using the same PCB is required, for example GN or REPL, type the call into the function code and press the ENTER key. The program executes that call. To build a new call, go to the ENTER panel again by simply pressing the ENTER key.

The ERMP (error) panel

If DL/I finds an error in the call, the program displays the parameters used in the call, the PCB information, and up to 256 bytes of the PCB key feedback field. Press the ENTER key to return to the ENTER panel, or the CLEAR key to end the program.

Further requirements

Further requirements for DFHTDLI include:

TRANSACTION definition

TRANSACTION(TDCI)
PROGRAM(DFHTDLI)
TWASIZE(1200)
(long conversational)

PROGRAM definition PROGRAM(DFHTDLI)

A PROGRAM definition is not needed for the PSB for DL/I; the PSB name is specified in the DL/I ACT entry for DFHTDLI.

MAPSET Definition MAPSET(TDUMS)

FCT entries Specify each database in the CICS file control table that you use for the DL/I sample.

Note

1. DFHTDLI issues **any** call entered on the panel except system or command code calls.
2. If you use a PSB with a PROCOPT of other than G, an intent scheduling conflict in DL/I may occur, causing other tasks to wait for access to the database being used. This program is a long conversation which keeps its PSB scheduled from the start until terminated. Refer to the DL/I manuals for more details on intent scheduling.
3. You must ***not*** use this program to read or write segments greater than 512 bytes long, or unpredictable results may occur.

Source code for DFHTDLI

The source code for DFHTDLI and the BMS mapset TDLIMS can be found in the following PRD1.BASE sublibrary members:

```
DFHTDLI.A     Program
TDLIMS.A      Symbolic description map (copy code)
DFHXTDLM.A    BMS mapset source code for TDLIMS
```

Chapter 7. Defining terminal resources

This chapter describes how to define to CICS the terminals (and logical units) that it is to use, and how it is to use them. You define terminals to CICS in one of two ways, depending on the type of terminal access method you are using:

1. VTAM terminals are defined in the CSD either explicitly, or by using model terminal definitions if you are using the CICS automatic installation facility (**autoinstall**). Using autoinstall, you leave it to CICS to install the terminal resource definition dynamically at logon time. CICS obtains the information needed to create a terminal control table terminal entry (TCTTE) from the RDO TERMINAL and TYPETERM definitions recorded in the CSD. For guidance information about this process, see the *CICS Resource Definition Guide*.

   You can add VTAM definitions to the CSD offline using the DEFINE command of the CICS utility program, DFHCSDUP, or online using the CEDA DEFINE command. If you want the terminal definitions installed during CICS initialization, you must add the names of the groups containing the definitions to a group list used during a cold start. Otherwise you can install a group of definitions using the CEDA INSTALL GROUP(groupname) command online. For details of the GRPLIST system initialization parameter, see Chapter 22, “CICS system initialization” on page 187.

   Each terminal must also be defined to VTAM in a VTAM definition statement.

2. Non-VTAM terminals are defined in a terminal control table (TCT) using DFHTCT macros.

   During CICS initialization, CICS loads the TCT specified by the TCT system initialization parameter, and those terminals defined in the TCT are installed as CICS resources. You must also make these terminals known to the operating system, and include a DLBL statement in the CICS startup job stream for each terminal.

If you are running CICS with XRF, see “XRF considerations” on page 81.

VTAM terminals

If your CICS system is to communicate with terminals or other systems using VTAM services, you must:

1. Define CICS to VTAM with an APPL statement.
2. Define to VTAM the terminal resources that CICS is to use. For more information about defining terminal resources to VTAM, see “Defining CICS terminal resources to VTAM” on page 72.
3. Define to CICS the terminal resources that it is to use. For more information about defining terminal resources to CICS, see “Defining terminal resources to CICS” on page 72.

Defining CICS terminal resources to VTAM

Each terminal, or each logical unit (LU) in the case of SNA terminals, that CICS is to use must be defined to VTAM. The terminals can be defined as local or remote.

Local VTAM terminals
can be SNA terminals connected to a channel-attached cluster or they can be non-SNA 3270 terminals connected via a local control unit.

Remote VTAM terminals
are attached to an SNA cluster controller, which is connected via an SDLC line with a channel-attached communications controller. The communications controller may also be loaded with code to enable remote terminals to be connected to it by a binary synchronous (BSC) line.

You define terminals, controllers, and lines in VTAM tables[4] as nodes in the network. Each terminal, or each logical unit (LU) in the case of SNA terminals, must be defined in the VTAM tables with a VTAM node name that is unique throughout the VTAM domain.

With VTAM 4.2 or later, you can define the AUTINSTMODEL name, printer, and alternate printer to VTAM by using VTAM MDLTAB and ASLTAB macros. These definitions are passed to CICS to select autoinstall models and printers.

For information about defining resources to VTAM, see the *VTAM Resource Definition Reference* manual.

Defining terminal resources to CICS

A given VTAM terminal (or logical unit) may be defined explicitly in the CICS system definition file (CSD), in which case it has a TERMINAL name, and a NETNAME (which is the same as the VTAM node name). Terminals defined in this way have terminal control table entries (TCTTEs) installed at CICS startup.

If a terminal does not have an explicit definition in the CSD, CICS can create and install a definition dynamically for the terminal when it logs on, using the CICS autoinstall facility. CICS can autoinstall terminals by reference to TYPETERM and model TERMINAL resource definitions created with the AUTINSTMODEL and AUTINSTNAME attributes. For information about **TYPETERM** and **TERMINAL** resource definitions, see the *CICS Resource Definition Guide*.

If you use autoinstall, you must ensure that the CICS resource definitions correctly match the VTAM resource definitions. For programming information about VTAM logmode definitions and their matching CICS autoinstall model definitions, see the *CICS Customization Guide*.

If you specify the system initialization parameter TCTUALOC=ANY, CICS stores the terminal control table user area (TCTUA) for VTAM terminals above the 16MB line if possible. (See page 191 for more information about the TCTUALOC parameter.)

[4] VTAM has tables describing the network of terminals with which it communicates. VTAM uses these tables to manage the flow of data between CICS and the terminals.

Defining the terminal shutdown time limit

You can specify a time limit within which all VTAM terminals used CICS must shut down, when CICS is shutting down. (This is to prevent a hung terminal stopping CICS shutting down.) You specify this time limit on the TCSWAIT system initialization parameter. You can also specify actions that CICS is to take, if the time limit is exceeded. You specify the actions on the TCSACTN system initialization parameter. More information about choosing appropriate values for TCSWAIT and TCSACTN is given in the following sections.

Choosing an appropriate value for TCSWAIT

The value that you specify on the TCSWAIT system initialization parameter should be large enough so that under normal circumstances all VTAM terminals and connections shutdown in an orderly fashion. To help choose this value, consider using a value slightly larger than the elapsed time between the following two CICS terminal control shutdown messages:

```
DFHZC2305 Termination of VTAM sessions beginning
DFHZC2316 VTAM ACB is closed
```

Note: If you *do not* want a time limit (that is, you assume that all terminals never hang), specify the TCSWAIT=NO system initialization parameter.

Specifying that CICS is only to report hung terminals

To report hung terminals and not attempt to force-close them specify the TCSWAIT=mm (with an appropriate time interval) and TCSACTN=NONE system initialization parameters.

Specifying that CICS is to force close all hung terminals

To attempt to force-close all hung terminals specify the TCSWAIT=mm (with an appropriate time interval) and TCSACTN=UNBIND system initialization parameters.

Specifying that CICS is to force close some hung terminals

To attempt to force-close some hung terminals, and only report others, specify the TCSWAIT=mm (with an appropriate time interval) and TCSACTN=NONE system initialization parameters, and code a DFHZNEP routine that selects the required terminals and sets TWAOCN on for them.

To attempt to force-close the CICS VTAM ACB if there are any hung terminals, specify the system initialization parameters TCSWAIT=***mm*** (with an appropriate time interval) and TCSACTN=FORCE.

Limitations of the terminal shutdown time limit facility

The following limitations apply to the terminal shutdown time limit:

- The terminal control shutdown time limit facility is only for VTAM terminals and VTAM intersystem connections.
- For all CICS-supported VTAM terminals, including LU Type 6.2 single-session APPC terminals (but excluding LU Type 6.1 connections and LU Type 6.2 parallel connections), the following facilities are provided:
  - The TCSWAIT-controlled shutdown timing mechanism.
  - The TCSACTN- and DFHZNEP-controlled, optional, force close mechanism.

  - The following messages:

    ```
    DFHZC2350 Threshold exceeded. Sessions still active: ...
    DFHZC2351 Terminal still active. Reason: ...
    ```

- For all VTAM intersystem connections, including both LU Type 6.1 connections and LU Type 6.2 parallel connections (but not LU Type 6.2 single-session APPC terminals), the following facilities are provided:
  - The TCSWAIT-controlled shutdown timing mechanism.
  - The message: `DFHZC2352 Connection still active`
- The force-close action on a hung terminal (no quiesce protocol, issue VTAM CLSDST, send UNBIND to the terminal) only ATTEMPTS to shutdown the terminal; there is no guarantee that all terminals will shutdown in all circumstances.

To guarantee that all VTAM terminals are shutdown, and the VTAM ACB is closed, you must specify TCSACTN=FORCE.

Defining sequential devices to CICS

You can use a pair of input and output sequential data sets to simulate a terminal to CICS. You can do this to test an application program before an intended terminal becomes available. There are two ways to define sequential devices:

1. You can code four macros:

   DFHTCT TYPE=SDSCI to describe the input sequential data set
   DFHTCT TYPE=SDSCI to describe the output sequential data set
   DFHTCT TYPE=LINE to describe the I/O combination
   DFHTCT TYPE=TERMINAL to describe the I/O combination.

2. You can code a single DFHTCT TYPE=GPENTRY macro.

In the sample tables, a card reader and a line printer are defined to simulate a terminal. The DEVADDR operands in the DFHTCT TYPE=SDSCI macros (or the GPSEQLU list in the DFHTCT TYPE=GPENTRY macro) specify the symbolic unit addresses of the devices. In the sample tables, they are specified as SYSRDR and SYSLST respectively.

This I/O combination simulates a “terminal” known to CICS by the TRMIDNT name, SAMA. You can submit input data to a CICS application program through the card reader (SYSRDR), and output to the “terminal” is sent to the line printer (SYSLST).

Using disk data sets as simulated terminals

As an alternative to using sequential data sets to simulate a terminal, you can use two disk data sets to simulate a terminal. To do this, code the necessary DLBL, EXTENT, and ASSGN job control statements for the disk files in your CICS startup job stream. The file names you specify in the DLBL statements must be the same as the DSCNAME operands in the DFHTCT TYPE=SDSCI macro (or the GPNAME list in the TYPE=GPENTRY macro). This method is illustrated in Figure 27 on page 75.

```
       DFHTCT TYPE=SDSCI,       The macro for input          *
              DEVADDR=SYS001,                                *
              DEVICE=DISK,                                   *
              DSCNAME=DISKIN
       DFHTCT TYPE=SDSCI,       The macro for output         *
              DEVADDR=SYS006,                                *
              DEVICE=DISK,                                   *
              DSCNAME=DISKOUT
       DFHTCT TYPE=LINE,        The macro for the I/O        *
              ACCMETH=SEQUENTIAL,   disk combination         *
              TRMTYPE=DASD,                                  *
              ISADSCN=DISKIN,                                *
              OSADSCN=DISKOUT,                               *
              INAREAL=80
       DFHTCT TYPE=TERMINAL,    The macro  for the terminal  *
              TRMIDNT=SAMB,                                  *
              TRMPRTY=11,                                    *
              TRMSTAT=(TRANSCEIVE,'OUT OF SERVICE')
```

Figure 27. Using two disk data sets to simulate a terminal

You should code the job control statements for the example in Figure 27 as follows:

```
// DLBL    DISKIN,'simulatd.terminal.in',0,SD
// EXTENT  SYS001,volid,,,rtrk,ntrks
// ASSGN   SYS001,cuu

// DLBL    DISKOUT,'simulatd.terminal.out',0,SD
// EXTENT  SYS006,volid,,,rtrk,ntrks
// ASSGN   SYS006,cuu
```

where:

- `volid` is the volume serial identifier of the DASD device.
- `cuu` is the physical address of the DASD device.
- `rtrk` is the relative track number of the start of the disk extent (relative block number in the case of an FBA device).
- `ntrks` is the number of tracks allocated for the disk extent (number of tracks in the case of an FBA device).

This I/O combination simulates a “terminal” known as SAMB to the CICS system. You submit input from this “terminal” from the DISKIN data set. Output to the “terminal” is sent to the DISKOUT data set.

Each statement in the input file (from SYSRDR or DISKIN in the examples used above), must end with a character representing `X'E0'`. The standard EBCDIC symbol for this end-of-data hexadecimal value is a backslash (\) character, and this is the character defined to CICS. You can redefine this for your installation on the EODI system initialization parameter; see Chapter 22, “CICS system initialization” on page 187 for details.

End-of-file does not terminate sequential input. You should use CESF GOODNIGHT as the last transaction, to close the device and terminate reading from the device. Otherwise, CICS invokes the terminal error program (DFHTEP), and issues the messages in Table 14 on page 76 at end-of-file on the sequential device:

Table 14. Warning messages if a sequential terminal is not closed

Message	***Destination***
DFHTC2507 *date time applid* Input event rejected return code *zz* {on line w/term|at term}*termid* {, trans}*tranid*{, rel line=} *rr,time*	CSMT
DFHTC2500 *date time applid* {Line|CU|Terminal} out of service {Term|W/Term} *termid*	CSMT

Use of CESF GOODNIGHT puts the sequential device into RECEIVE status and terminates reading from the device. However, if you close an input device in this way, the receive-only status is recorded in the warm keypoint at CICS shutdown. This means that the terminal is still in RECEIVE status in a subsequent warm start, and CICS does not then read the input file.

You can also use CESF LOGOFF to close the device and terminate reading from the device, but CICS still invokes DFHTEP to issue the messages in Table 14 at end-of-file. However, the device is left in TTI status, and is available for use when restarting CICS in a warm start.

If you want CICS to read from a sequential input data set, either during or following a warm start, you can choose one of the following methods:

- Close the input with CESF LOGOFF, and ignore the resultant messages. This leaves the terminal in TTI state, and CICS reads input automatically in the next startup.
- Do not close the input, and ignore the resultant messages. This leaves the terminal in TRANSCEIVE state, and CICS reads input automatically in the next startup.
- Close the input with CESF GOODNIGHT, and change the status from RECEIVE to TRANSCEIVE ***after*** CICS initialization by using the CEMT master terminal transaction (input through the console or a master terminal):

  ```
  CEMT SET TERMINAL(termid) TTI
  ```

- Code a user program, to be invoked from the program list table (PLT), to issue the appropriate EXEC CICS INQUIRE and EXEC CICS SET commands for each sequential device that is required to process input. For example, use the following statement to establish the state of a sequential terminal:

  ```
  EXEC CICS INQUIRE TERMINAL(termid) SERVSTATUS(cvda) TTISTATUS(cvda)
  ```

  For each terminal where SERVSTATUS returns DFHVALUE(INSERVICE) and TTISTATUS returns DFHVALUE(NOTTI), set the terminal to TRANSCEIVE with the following statement:

  ```
  EXEC CICS SET TERMINAL(termid) TTI
  ```

  For programming information about the use of EXEC CICS INQUIRE and EXEC CICS SET commands, see the *CICS System Programming Reference* manual. For programming information about writing post initialization-phase programs, see the *CICS Customization Guide*.

If you use SAM devices for testing purposes, the final transaction to close down CICS could be CEMT PERFORM SHUT.

Console devices

You can operate CICS from a **console device**. A console device may be:

- The VSE system console
- A console managed by the Interactive Interface (II)
- A console managed by the Distributed Workstation feature
- A console managed by the VM/VSE feature.

You can use a terminal as both a system console and a CICS terminal. To enable this, you must define the terminal as a console in the CSD. (It is no longer possible to define consoles using the DFHTCT macro.)

You can use each console device for normal operating system functions and to invoke CICS transactions. In particular, you can use the console device for CICS master terminal functions to control CICS terminals or to control several CICS regions in conjunction with multiregion operation. Consequently, you can be a master terminal operator for several CICS regions.

You can also use console devices to communicate with alternate CICS regions if you are using XRF. Communicating in this way is limited to use of the CICS-supplied transaction, CEBT.

For further guidance information about operating CICS from a console device, see the *CICS Operations and Utilities Guide*.

Defining console devices to CICS

You can define console devices to CICS by using either the DEFINE TERMINAL command of the DFHCSDUP utility, or the CEDA DEFINE TERMINAL command using RDO.

Defining VSE consoles to CICS

To use a VSE console as a CICS master terminal, you must define it to CICS by a terminal definition entry in the CSD. Each console that you define is identified by the CONSNAME parameter on the TERMINAL definition:

CONSNAME(name) is an 2-8-character console name, used to generate a specific TCTTE for each console that can accept CICS commands. The names correspond to the names defined to VSE for the console. For example, it could be:

- “SYS” for the system console;
- The userid for an Interactive Interface user with console access. (See the *VSE/ESA Administration* guide for details on how to define an Interactive Interface userid.)

For an example of the DEFINE command required to define a console, see Figure 28 on page 78.

For further guidance on defining consoles to VSE, see the *VSE/ESA Planning* manual.

```
//JOB    DEFTERM
//DLBL   usercat,'usercat',,VSAM
//DLBL   DFHCSD,'user.dfhcsd',,VSAM,CAT=usercat
//LIBDEF PHASE,SEARCH=PRD1.BASE
//EXEC   DFHCSDUP,SIZE=DFHCSDUP
*  Define a console for CICS
DEFINE TERMINAL(trmidnt)   GROUP(grpname)     TYPETERM(DFHCONS)
       CONSNAME(consname)  DESCRIPTION(VSE CONSOLE consname)
*
APPEND LIST(DFHLIST)  TO(yourlist)
*
ADD GROUP(grpname) LIST(yourlist)
*
LIST   LIST(yourlist)  OBJECTS
/*
/&
```

Figure 28. Defining consoles in the CSD using DFHCSDUP.

Note: You must substitute your own values for the operands that are shown in lowercase in the DEFTERM job shown in Figure 28.

GROUP(grpname)
A unique name for the group to which the console resource definition is to belong.

TERMINAL(trmidnt)
A unique 4-character terminal identifier as the name by which CICS is to know the console.

CONSNAME(consname)
This uniquely identifies the name of a console device (2-8 characters) within a VSE image. The names correspond to the names by which the console is defined to VSE. For example, to define the system console code CONSNAME(SYS).

Note: If the name DFHCONxx (where ***xx*** is any valid character) is specified, the terminal is used by CICS as a ***pool*** console for use when processing a modify command from a console with a name that is not defined to CICS.

USERID(userid) - optional
The CICS preset security userid to be used for all security at this console device. This userid is permanently signed on when the console device is installed. That is, the userid cannot be signed off, and the security for the console is checked only when it is installed. Ensure that only authorized users can install terminals with preset security. For further information, see the *CICS Security Guide*.

Note: Preset Security, if defined for pooled consoles (those with a CONSNAME of DFHCONxx) allows each pooled console to be assigned only once as each is then permanently signed-on. This means that, once used, each pool entry is not returned to the pool but is permanently assigned to the console name that 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.

TO(yourlist) and LIST(yourlist)
A unique name for *yourlist*. This new list of the groups of resource definitions to be installed at CICS startup must include all the CICS-supplied resources as well as your own.

Installing resource definitions for console devices

Having defined the console devices in the CSD, you must ensure that their resource definitions are installed in the running CICS region. You can install the definitions in one of two ways, as follows:

1. At CICS system initialization, add the definitions to a group list specified on the GRPLIST system initialization parameter.
2. During CICS execution, install the console device group by using the RDO command CEDA INSTALL GROUP(*groupname*), where *groupname* is the name of the resource group containing the console device definitions.

DFHLIST, the CICS-defined group list created when you intialize the CSD with the DFHCSDUP INITIALIZE command, does not include any resource definitions for console devices. Specify the CICS-supplied TYPETERM definition, DFHCONS, on the TYPETERM(*name*) parameter. This TYPETERM definition for console devices is generated in the group DFHTYPE when you initialize the CSD. For information about terminal definitions, see **TERMINAL** and **TYPETERM** in the *CICS Resource Definition Guide*.

VTAM persistent sessions considerations

VTAM persistent session support improves the availability of CICS. It benefits from VTAM persistent LU–LU session improvements to provide restart-in-place of a failed CICS without rebinding.

CICS support of persistent sessions includes the support of all LU–LU sessions except LU0 pipeline and LU6.1 sessions. CICS determines for how long the sessions should be retained from the PSDINT system initialization parameter. This is a user-defined time interval. If a failed CICS is restarted within this time, it can use the retained sessions immediately—there is no need for network flows to rebind them.

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

If CICS is terminated through a CEMT PERFORM SHUTDOWN IMMEDIATE command, or if CICS fails, its sessions are placed in “recovery pending” state.

During emergency restart, CICS restores those sessions pending recovery from the CICS global catalog and the CICS system log to an “in session” state. This happens when CICS opens its VTAM ACB.

Subsequent processing is LU dependent: cleanup and recovery for non-LU6 persistent sessions are similar to that for non-LU6 backup sessions under XRF. Cleanup and recovery for LU6.2 persistent sessions maintain the bound session when possible but there are cases where it is necessary to unbind and rebind the sessions, for example, where CICS fails during a session resynchronization.

The end user of a terminal sees different symptoms of a CICS failure following a restart, depending on whether VTAM persistent sessions, or XRF, are in use:

- If CICS is running without VTAM persistent sessions or XRF, and fails, the user sees the VTAM logon panel followed by the “good morning” message (if AUTOCONNECT(YES) is specified for the RDO TYPETERM resource definition).
- If CICS does have VTAM persistent session support and fails, the user perception is that CICS is “hanging.” The screen on display at the time of the failure remains until persistent session recovery is complete. After a successful CICS emergency restart, the recovery options defined for the terminals or sessions take effect. The recovery options are specified on the RECOVOPTION parameter of the RDO TYPETERM resource definition. If you specify RECOVOPTION(SYSDEFAULT), the terminal user can clear the screen and continue to enter CICS transids. If you specify RECOVNOTIFY(MESSAGE) as an attribute of the RDO TYPETERM resource definition, the user is notified of the successful recovery.

Unbinding sessions

Sessions held by VTAM in a recovery pending state are not always reestablished by CICS. CICS (or VTAM) unbinds recovery pending sessions in the following situations:

- If CICS does not restart within the specified persistent session delay interval
- If you perform a COLD start after a CICS failure
- If CICS restarts with XRF=YES (when the failed CICS was running with XRF=NO)
- If CICS cannot find a terminal control table terminal entry (TCTTE) for a session (for example, because the terminal was autoinstalled with AIRDELAY=0 specified)
- If a terminal or session is defined with the recovery option (RECOVOPT) set to UNCONDREL or NONE
- A connection is defined with the persistent session recovery option (PSRECOVERY) set to NONE.

In all these situations, the sessions are unbound, and the result is as if CICS has restarted following a failure without VTAM persistent session support.

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

Sessions not retained

There are some circumstances in which VTAM does not retain LU–LU sessions:

- VTAM does not retain sessions after a VTAM, VSE, or processor (CPC) failure.
- VTAM does not retain CICS sessions if you close VTAM with any of the following CICS commands:
  - SET VTAM FORCECLOSE
  - SET VTAM IMMCLOSE
  - SET VTAM CLOSED

- VTAM does not retain CICS sessions if you close the CICS node with the VTAM command VARY NET,INACT,ID=applid
- VTAM does not retain CICS sessions if you perform a normal CICS shutdown (with a PERFORM SHUTDOWN command).

Without persistent session support, all sessions existing on a CICS system are lost when that CICS system fails. In any subsequent restart of CICS, the rebinding of sessions that existed before the failure depends on the terminal's AUTOCONNECT option. If AUTOCONNECT is specified for a terminal, the user of that terminal waits until the GMTRAN transaction has run before being able to continue working. If AUTOCONNECT is not specified for a terminal, the user of that terminal has no way of knowing (unless told by support staff) when CICS is operational again unless the user tries to log on. In either case, users are disconnected from CICS and need to reestablish a session, or sessions, to regain their working environment.

XRF considerations

If you intend operating CICS with the extended recovery facility (XRF), there are some points to consider when setting up your terminal network.

Terminals

All terminals must have their sessions reestablished to the new active CICS system after a takeover. How you achieve this depends on the network and the XRF configuration.

You must consider how the terminals in your network are defined and configured. If active and alternate CICS systems are in different CPCs, you must make arrangements for physically switching the terminals to the other CPC after a takeover has started. This affects your network definitions, your network controllers, and your switching arrangements.

Applids and the propagation of user variable

VTAM users can log on using an applid that is not specific to either the active or the alternate CICS. This is known as the **generic** applid. The applids that are specific to the active CICS or the alternate CICS are known as the **specific** applids. Applids are defined to CICS on the APPLID system initialization parameter, or SIT override. The APPLID parameter has the following format:

```
APPLID=({DBDCCICS|name1}[,name2])
```

If you are running CICS without XRF, you must specify **name1** only. If you are running with XRF, you must code both name1 and name2. In this case, **name1** is the generic applid, and **name2** is the specific applid. See page 217 for more information about the APPLID system initialization parameter.

CICS issues a VSE MODIFY command to your VTAM system, to define the name of a user variable and assign an initial value. CICS issues this command twice during initialization; once immediately after opening terminal data sets, and the second just prior to the end of initialization. The alternate CICS issues the F NET USERVAR command when it takes over from the active system. The name of the user variable (USERVAR) is always the CICS generic applid. The value assigned to the user variable in the F NET USERVAR command is the name of the specific applid of the CICS system. (This must first have been defined, in an APPL statement, to the VTAM to which the command is addressed.)

The syntax of the command is as follows:

```
MODIFY vtamname,USERVAR,ID=generic-name,VALUE=specific-name,
                OPTION=UPDATE
```

VTAM terminals can log on to the active CICS in the usual way, using the specific applid. For more information about these and other VTAM considerations when running CICS with XRF, see the *CICS XRF Guide*.

Chapter 8. Enabling the CICS spooler interface and using the report controller

This chapter describes how to enable the CICS spooler interface before you can use it with VSE/POWER and the report controller. You must:

1. Have VSE/POWER initialized in one of your VSE partitions (usually by means of an automated system initialization (ASI) JCL procedure).
2. Activate the CICS system spool interface as described in "Enabling the CICS system spool interface."

   Note: If all you want to do is to allow CICS programs to create and acess spool files, you only need to perform steps 1 and 2.
3. Create the required report controller resource definitions as described in "Including report controller resource definitions in a CICS system" on page 84.
4. Use the escape facility to process nonstandard data streams as described in "Using escape programs" on page 87.
5. Set resource security levels for batch-created reports as described in "Resource security level for batch-created reports" on page 88.
6. Select parameters to start printers attached to CICS, and review the requirements for using fast system printers attached to CICS as described in "Fast printers attached to CICS" on page 88.
7. Define a default FCB if the CICS pseudo FCB is not suitable for your installation. See "Default FCB" on page 89.

Enabling the CICS system spool interface

Enable the CICS system spool interface coding SPOOL=YES in your system initialization table, or as a startup override. SPOOL=YES allows you to use the CICS interface with VSE/POWER and the report controller.

See page 263 for more information about the SPOOL system initialization parameter.

Error conditions

CICS returns an error condition if you:

- Have not specified SPOOL=YES.
- Attempt to use the report controller transactions, or any of the EXEC CICS SPOOL*xxx*... REPORT commands (where SPOOL*xxx* is one of SPOOLOPEN, SPOOLCLOSE, or SPOOLWRITE) when you do not have the report controller installed.

The error condition may be an AEY9 transaction abend, a NOSPOOL condition, or the message "SPOOLING SYSTEM IS NOT AVAILABLE".

Some instances of the AEY9 and NOSPOOL conditions are shown in Table 15 on page 84.

Table 15. Examples of AEY9 and NOSPOOL conditions

SIT parameter	**Report controller installed**
SPOOL=YES	EXEC CICS SPOOL*xxx* ... REPORT and EXEC CICS SPOOL*xxx* commands are valid
SPOOL=NO	SPOOL*xxx* commands give abend code AEY9

For details of the exception conditions that can occur, see the *CICS Problem Determination Guide*.

Including report controller resource definitions in a CICS system

To use the report controller, you must include the report controller resource definitions in your CICS system. There are three categories of resource definitions:

1. Program and transaction resource definitions
2. Printer resource definitions for terminal-attached printers
3. Destination control table entries for transient data queues.

Program and transaction resource definitions

You must add the report controller definitions to your CSD using the CICS-supplied utility program, DFHCSDUP, with the command UPGRADE USING(DFHCURCF).

If you are using the Report Controller for the first time, you will notice warning messages being produced by DFHCSDUP regarding missing definitions. These messages can safely be ignored.

See the *CICS Operations and Utilities Guide* for more information about using the DFHCSDUP UPGRADE command.

When DFHCSDUP adds the IBM-defined group DFHRCF to your CSD it does ***not*** add the group to the IBM-defined group list, DFHLIST. Before you specify SPOOL=YES as a system initialization parameter, you should include group DFHRCF in a group list you intend using for your CICS startup. You do this using the CEDA ADD command or the DFHCSDUP offline utility ADD command, as described in the *CICS Resource Definition Guide*.

Alternatively, you can install group DFHRCF **after** CICS is initialized, using the following command:

```
CEDA INSTALL GROUP(DFHRCF)
```

You can use this option the first time you initialize CICS with SPOOL=YES, and also add the group to a group list for subsequent initializations.

However, when installing the group DFHRCF using the CEDA transaction in this way you must use the CEMS transaction to initiate the CXPB transaction before you attempt to start a printer for use. The CXPB transaction is initiated automatically during subsequent CICS initialization if the DFHRCF group is added to a group list used by your CICS startup.

Sample transactions, ARPS and SREP

The DFHRCF definitions include the sample transactions ARPS and SREP, for use with the report controller sample application programs.

Before you can run the sample transaction ARPS, you must install the FILEA sample application group, DFH$AFLA using the following command:

```
CEDA INSTALL GROUP(DFH$AFLA)
```

DFH$AFLA contains the map resource definitions needed for the FILEA samples.

Because ARPS uses the page retrieval command, you must ensure the SIT parameter PGRET=P/ is specified.

Printer resource definitions

You should define to CICS any printers that you want to use with the report controller. You can do this in the same way as for other terminal definitions, using either RDO or DFHCSDUP DEFINE TERMINAL commands.

For more information about resource definitions for printers, see the *CICS Resource Definition Guide*.

Destination control table resource definitions

If you want to maintain a report controller audit trail, then you must define entries in your destination control table for the transient data queues called CSPA and CSPW. These are used by the report controller as follows:

CSPA To maintain a log of changes to the characteristics of reports and printers

CSPW To maintain a log of action messages for the writer task (CEPW), and of severe error messages.

If CSPA and CSPW are defined as transient data **intrapartition** data sets, you can use the CEMS transaction to create reports from these queues.

The copybook, DFH$DCTF, in the VSE/ESA sublibrary, PRD1.BASE, contains sample definitions for CSPA and CSPW.

The DCT macros shown in Figure 29 are an example of how to define the transient data queues CSPA and CSPW. In this example, the queues CSPA and CSPW are defined as intrapartition transient data sets.

```
         DFHDCT TYPE=INITIAL,SUFFIX=rc
               .

               .

               .

CSPW     DFHDCT TYPE=INTRA,DESTID=CSPW,DESTFAC=FILE,DESTRCV=PH
*
CSPA     DFHDCT TYPE=INTRA,DESTID=CSPA,DESTFAC=FILE,DESTRCV=PH
               .

               .

               .

         DFHDCT TYPE=FINAL
         END
```

Figure 29. Example DFHDCT source statements for the CSPA and CSPW queues

Sample programs for use with the report controller tutorial

The sample transaction SREP generates the set of reports used in the tutorial section of the *CICS Report Controller User's Guide*. The SREP transaction is provided as an aid to end-user training. If you make the transaction available, either on your production CICS system or on a test CICS system, the user can sign on to CICS and type in the transaction code.

SREP generates a set of reports with the user's user ID as the report controller destination. End users can process these reports as they work through the tutorial, without affecting other users of the report controller, or other users of the tutorial.

The program names and transaction identifiers for this function are shown in Table 16.

Table 16. Program names, and transaction identifiers for the report controller

Language	Program	Transid
Assembler language	DFH$ASRE	SREP
COBOL	DFH0CSRE	Not defined
PL/I	DFH$PSRE	Not defined

Note that there are report controller sample programs are not currently available for C.

Using the report controller sample programs

Before you make report controller sample programs available to your users, you should carry out the following checks:

1. The report controller sample program takes the user ID from the CICS signon as the destination identifier for the set of reports generated. If your users do not sign on to CICS using one of the signon transactions, the signon user ID is not available, so you must provide an alternative method for obtaining a report

destination. You can do this by modifying the appropriate program, either to pick up a user ID from some other source, or use some convention of your own to generate a destination identifier.

2. Make sure that there are no conflicts between the characteristics generated on the sample reports and your normal installation standards.

   The reports are generated with the following:

Class	The sample program does not generate a report class and therefore the report controller takes the default spool class that you have specified on the SPOOL system initialization parameter.
Report names	These are listed in the *CICS Report Controller User's Guide*.
Userdata	A variety of examples are used; see the source listings for details of the data.
Priority	This has effect within the destination only, but you may need to check this if you modify the way in which the destination identifier is derived (see item 1 above).
Forms type	The sample programs use either 'SAMP' or blank (the default).
Size of reports	These have been kept small, and should be accommodated on all standard paper sizes.

If the destination identifier and sample report characteristics are satisfactory for your installation, you can use the report controller sample program (which is supplied in source form in the VSE/ESA sublibrary, PRD1.BASE).

Before you use the sample program (in the language of your choice), you must edit the source, translate, compile (or assemble), and link-edit it. (Some guidance on modifying the sample programs is given in the comments at the beginning of the sample source members.)

If you install an amended version, you should check the resource definitions for the transaction and program. Note that it is only the assembler-language version that has a transaction code defined in the CSD by the report controller upgrade module DFHCURCF.

Note: If you change any of the report characteristics, remember to note the changes in the tutorial text before you make them available to any user of the report controller.

The tutorial advises users that they should delete the sample reports when they have finished.

Using escape programs

To handle nonstandard data streams of batch-generated reports, such as graphics, you need to specify the name of an **escape program** in the JCL for the report. The named escape program is invoked when the report is printed. For example, you can use this method for printing graphics, or graphics and text on the same page. However, these reports cannot be printed on a system printer, but they can be printed on CICS printers.

Specifying the name of an escape program

Specify the name of an escape program on the ESCAPE option of the SPOOLOPEN REPORT command. Ensure that the escape program is locatable at printing time.

When printing ESCAPE reports, CEMS or CEOS printer commands cannot be executed until control is passed back to CICS from your escape program. Control is passed back at end-of-copy and at end-of-report. This means that you cannot stop or pause a printer until the end of the copy or report is reached.

For more information about specifying the ESCAPE option on the SPOOLOPEN REPORT command, see the *CICS Report Controller Planning Guide*.

You can also specify the name of an escape program in a batch job. You use the POWER autostart DEFINE statement to specify your own keywords to be used on JECL statements in a batch job. The *CICS Report Controller Planning Guide* describes in detail how to do this.

When a batch-generated report is directed to the report controller print component, the escape program name is retrieved and the program invoked as if it had been specified on the SPOOLOPEN REPORT command.

Resource security level for batch-created reports

Note: Although CICS internal resource security has been removed from CICS TS release 1, for reasons of compatibility with previous releases the RSL attribute for batch reports is retained.

By default, CICS treats batch-created reports on the VSE/POWER queue as having a resource security level (RSL) of 1, but you can override this default and allocate your own RSL value in the RSL option in CICSDATA (or your defined keyword). You do this in the same way that you define and use escape programs. The *CICS Report Controller Planning Guide* describes in detail how to do this.

Fast printers attached to CICS

When you specify **SYSPRT as a logical destination, it is interpreted as referring to batch reports in the VSE/POWER queue that do not have a user ID defined; that is, in CICS terms, reports that do not have a destination name. You can start a CICS terminal printer to service VSE/POWER or CICS reports produced by the report controller (which may have been destined for a central system printer), by using either a POWER PSTART command, or one of the report controller transactions (CEMS or CEOS).

For example:

```
PSTART DEV,name,SYSCICSa,b,PARM='destname'
```

where:

- `name` is the TERMINAL name or TRMIDNT of the printer
- `SYSCICS` identifies the CICS system to POWER
- `a` is specified on the SPOOL parameter
- `b` is the class of reports that the printer is to print

- `destname` is the name of the destination.

For more information about printers and VSE/POWER, see the *CICS Report Controller Planning Guide*.

When you use the destination **SYSPRT, it not only signifies that you are specifying reports in the VSE/POWER queue that have no user ID (CICS destination name), but also indicates that the report controller should use a larger buffer for the transfer of report data between VSE/POWER and CICS. These larger buffers, used by CICS when you specify **SYSPRT, minimize the communication overhead between VSE/POWER and CICS. They are intended for use when you are printing large reports on fast printers attached to CICS.

Note: The report controller checks the format of the VSE system date to determine the format of the VSE/POWER queue records for report selection. For report controller record selection to work correctly, the reports on the VSE/POWER queue must be in the format of the VSE system date.

Default FCB

If either the report or the CICS printer specifies an FCB, the report controller uses a pseudo-FCB that specifies 12-inch paper at 6 lines per inch. If this is not acceptable for your installation, you must catalog a user default FCB with the the phase name, DFHPSFCB, in a library that is accessible by CICS. Do **not** specify this name on an RDO PROGRAM definition.

For further guidance information about adding support for the report controller, see the *CICS Report Controller Planning Guide*, and the *CICS Report Controller User's Guide*.

Part 2. Defining data sets

Having defined your resources and installed your application programs (as described Part 1, “Installing resource definitions” on page 1), you must now set up data sets and data definition statements.

Table 17. Defining data sets road map

If you want to know about ...	Refer to...
The required CICS data sets	Chapter 9, “Preparing to set up CICS data sets” on page 93
Temporary storage data sets	Chapter 10, “Defining the temporary storage data set” on page 101
Transient data queues	Chapter 11, “Defining transient data destination data sets” on page 105
Journaling and archiving data sets	Chapter 12, “Defining data sets for journaling and archiving” on page 113
DMF data sets	Chapter 13, “Defining data sets for the Data Management Facility (DMF)” on page 125
The CSD	Chapter 14, “Defining the CICS system definition data set” on page 129
The restart data set	Chapter 15, “Defining and initializing the restart data set” on page 147
The new CICS catalogs	Chapter 16, “Defining and using catalog data sets” on page 151
Auxiliary trace data sets	Chapter 17, “Defining and using auxiliary trace data sets” on page 159
Dump data sets	Chapter 18, “Defining dump data sets” on page 163
CAVM data sets	Chapter 19, “Defining the CICS availability manager data sets” on page 169
User files	Chapter 20, “Defining user files” on page 175
DL/I data sets	Chapter 21, “DL/I definitions” on page 183

Chapter 9. Preparing to set up CICS data sets

This chapter shows you how to define the data sets you need to run CICS. Some of these data sets are mandatory, whilst others are needed only if you are using the corresponding facilities.

You need:

- Disk label (DLBL) statements for all your VSAM application files and DL/I databases
- ASSGN job control statements for terminals other than VTAM terminals.
- DLBL, EXTENT and ASSGN statements for non-VSAM files
- TLBL and ASSGN statements for labeled tape files
- ASSGN statements only for unlabeled tapes.

Space calculations are given so that you can calculate how much space to allocate to the data sets, and the data definition statements to define them to the running CICS region.

CICS utility programs provided for postprocessing of the data sets are described in the *CICS Operations and Utilities Guide*.

Overview of setting up CICS data sets

Before you start setting up your CICS data sets, review the CICS programs you need, and their data set requirements.

You then have to:

- Set up the data sets and libraries that are used by the CICS programs during execution.
- If necessary, initialize or preformat the data sets for use during CICS execution. Protect the data sets to suit your security requirements.
- Include in the CICS startup job stream DLBL statements for the required data sets, but note that DLBL statements are ***not*** needed for user files which you are using CICS dynamic allocation facilities for.

  For more information about user file and DL/I file definitions, see Chapter 20, “Defining user files” on page 175 and Chapter 21, “DL/I definitions” on page 183.

Table 18 on page 94 summarizes the CICS data sets and their characteristics.

Data set naming conventions

There are no restrictions on the data set names you choose for CICS data sets, other than VSE constraints. In the examples in this book, CICS410 is used as the high-level qualifier, and the DLBL file name as the lowest level. If you are running multiple CICS regions, and especially if you are running CICS with XRF, you can use the CICS APPLID as a second level qualifier.

If the data set is shared between an active CICS region and an alternate CICS region, use the generic APPLID, but if the data set is unique to either the active or the alternate CICS region, use the specific APPLID. For information about actively and passively shared data sets, see “Data set considerations when running CICS with XRF” on page 97.

Table 18 (Page 1 of 2). Summary of CICS data sets

Data set	Filename used by CICS	Block or control interval size (bytes)	Record format	Data set organization	Other comments
AUXILIARY TRACE (See page 159)	DFHAUXT DFHBUXT	4096	F	Sequential	
CAVM CONTROL (See page 169)	DFHXCTL	4096 minimum	**1**	VSAM ESDS	Required if running CICS with XRF.
CAVM MESSAGE (See page 169)	DFHXMSG	4096 minimum	**1**	VSAM ESDS	Required if running CICS with XRF.
CATALOGS (See page 151)	DFHGCD DFHLCD	8192 & 2048	VB	VSAM KSDS	Both data sets must be initialized before use (**2**).
CSD (See page 129)	DFHCSD	8192	VB	VSAM KSDS	---
DUMP (See page 163)	DFHDMPA DFHDMPB	32 760 (tape) or 1 track (DASD)	V	Sequential	For CICS transaction dumps only; see page 97 for information about system dumps.
JOURNAL (See page 113)	DFHJnnx where: nn=id of journal x=A or B	Defined in JCT as BUFSIZE= parameter (**4**)	U	Sequential	Defined by CICS as U, but record structure same as VB. May be processed as VB, if RECFM=VB is defined in JCL. These data sets must be formatted before they are used for the first time.
JOURNAL ARCHIVE CONTROL (See page 113)	DFHJACD	512	**1**	VSAM RRDS	Data set holds 199 records.
JOURNAL ARCHIVE sublibrary (See page 113)			FB	Library	Members of this sublibrary are named in the ARCHJCL parameter of the JCT entries.

Table 18 (Page 2 of 2). Summary of CICS data sets

Data set	Filename used by CICS	Block or control interval size (bytes)	Record format	Data set organization	Other comments
TRANSIENT DATA INITIALIZATION TERMINATION MESSAGES (see page 110)	DFHCXRF	136	V	Sequential	Used by CICS startup and shutdown when transient data not available.
MONOTORING (see page 125)	DFHDMF	---	1	VSAM ESDS	Must be formatted before use.
RESTART (See page 147)	DFHRSD	2048 (data: maximum 2000, average 400)	V	VSAM KSDS	Must be initialized before use with a dummy record. Initialized with one record; key is ‘ACTL 0001’ followed by 13 blanks.
TEMPORARY STORAGE (See page 101)	DFHTEMP	See page 103	1	VSAM ESDS	---
TRANSIENT DATA EXTRA-PARTITION (See page 105)	From DSCNAME operand of DFHDCT TYPE=EXTRA macro	From BLKSIZE operand of DFHDCT TYPE=SDSCI macro	From RECFORM operand of DFHDCT TYPE=SDSCI macro	Sequential	The TYPE=SDSCI macro referred to has the same DSCNAME parameter as the TYPE=EXTRA macro.
TRANSIENT DATA INTRA-PARTITION (See page 105)	DFHNTRA	See page 107.	1	VSAM ESDS	---

Notes for Table 18 on page 94:

1 These data sets use control interval (CI) processing and therefore the record format is not relevant.

2 DFHGCD is the CICS global catalog data set, and in an XRF environment it is passively shared between the active and the alternate CICS regions. DFHLCD is the CICS local catalog data set, and this is a unique data set; each CICS region must have its own local catalog. See “Data set considerations when running CICS with XRF” on page 97 for an explanation of actively and passively shared data sets in an XRF environment.

3 The CICS utility program, DFHTU410, prints and formats auxiliary trace data. For information about this CICS utility program, see the *CICS Operations and Utilities Guide*.

4 The maximum block size on journal tapes is 32 760 bytes, and the maximum block size of 32 760 bytes for disk journal data sets is defined in its DTF (define the file) in the journal formatting program, DFHJCJFP. You cannot override this block size by specifying a BLKSIZE parameter in the DLBL statement when you are formatting the journal data sets. You limit the actual size of the journal blocks by coding a BUFSIZE operand in the JCT entry for the journal. However, the actual size of the blocks written to journals by CICS can vary widely, up to the limit set by the BUFSIZE operand.

Multiple extents and multiple volumes

You can define a temporary storage data set or a transient data destination data set as a single extent defined on a single volume. That data set must be big enough to hold all your data. Instead of defining one data set, which might have to be much larger than your average needs to cater for exceptional cases, you can define:

- Multiple extents on one volume
- One extent on each of multiple volumes
- Multiple extents on multiple volumes.

When you define more than one extent, CICS uses the extra extents only when the primary extent is full. You could make your primary extent large enough to meet average demand, and then have smaller secondary extents for overflow. In this way, you are saving space until it becomes necessary to use it. As each extra extent becomes full, VSAM creates another. VSAM continues to create extra extents when needed, up to a maximum of 123 extents for data sets defined with the default NOREUSE operand (or a maximum of 16 extents per volume if the NOREUSE operand is explicitly coded).

To allocate additional extents in the same volume, code a secondary extent operand on the RECORDS parameter:

```
RECORDS(primary,secondary)
```

To use single extents on multiple volumes, code:

```
RECORDS(primary) -
VOLUMES(volume1,volume2,volume3,.....)
```

For multiple extents on multiple volumes, combine both primary and secondary RECORDS operands with multiple VOLUMES operands:

```
RECORDS(primary,secondary) -
VOLUMES(volume1,volume2,volume3,.....)
```

If a particular volume causes performance bottlenecks, try single extents on multiple volumes.

Multiple extents over multiple volumes should be used if there is a probability that a volume will exhaust its free space before VSAM reaches its limit on extra extents. If this occurs, VSAM continues to create extra extents on the next volume in the list, but be aware that VSAM always acquires the primary allocation when switching to a new volume.

Performance considerations of TS and TD buffers

When specifying the number of buffers for temporary storage and transient data, you should consider the following possible performance impacts:

- Using a large number of buffers means that for non-recoverable queues all processing can be performed without going to VSAM. This improves CICS performance. However, at shutdown all the buffers have to be flushed sequentially which can take a long time.
- If you specify a large number of buffers, and the number of queues is small, CICS takes longer to search down the chain of buffers for a particular queue.
- You can still specify only up to 255 VSAM strings. This means that there is no need for CICS to wait for VSAM strings.

VSE system data sets used by CICS

Besides its own system data sets, summarized in Table 18 on page 94, CICS also uses some VSE data sets. These are listed in Table 19:

Table 19. VSE data sets used by CICS

Data set	Owned or used by	Other comments
SYSDUMP sublibraries	VSE SDUMPX macro	Used by CICS for system dumps via the VSE SDUMPX macro.

Recalculate the size of this system data set, taking into account the increased volumes of data that CICS generates. For example, for a VSE dump sublibrary, you typically need a minimum of 1200 tracks of a 3380 device, or the equivalent. For guidance information about calculating the size of the SYSDUMP library, see the *VSE/ESA Diagnosis Tools* manual.

The SYSDUMP data sets can become full with unwanted SDUMPs that precede ASRA, ASRB, and ASRD abends (after message DFHAP0001). To prevent this, suppress such SDUMPs as described on page 164.

Data set considerations when running CICS with XRF

There are some factors that you must consider regarding both the CICS system data sets and the user application data sets when you are running CICS with XRF. These considerations are about the type of data set sharing that takes place between the active and the alternate CICS systems, and about data set allocation and disposition.

Consider the following general points when running CICS in an XRF environment:

- An alternate CICS region can be started before an active CICS system terminates.
- The active and alternate CICS systems may be executing in different VSE images.

It follows that the status and location of the data sets used by CICS become very important. In particular, consider the following points:

- For a given **file name**, do the active and alternate CICS systems:
  - Refer to separate data sets?
  - Refer to the same data set?
- For a given data set, is it required by the alternate CICS system:
  - Before takeover occurs?
  - After takeover occurs?

The sharing of data sets is an important factor when running CICS with XRF. A shared data set, in XRF terms, means one that is required by both the active and alternate CICS systems, though not necessarily concurrently. In an XRF environment, CICS classifies data sets as follows:

- Actively shared
- Passively shared
- Unique

Concurrent OPENs of passively shared data sets would normally cause problems. However, when you run CICS with XRF, CICS always ensures (except for the CSD) that there is no conflicting concurrent use of data sets by an active CICS and its alternate.

Actively shared data sets

Actively shared data sets are required for use in both the active and alternate CICS systems. There are only two CICS system data sets in this category, called the CICS availability manager (CAVM) data sets. The active and the alternate CICS regions each open these data sets during CICS initialization, and the data sets are shared while both CICS systems are running. They are the:

- CAVM control data set (DFHXCTL)
- CAVM message data set (DFHXMSG)

It is not usual for user application data sets to be actively shared.

Passively shared data sets

These data sets are required by both the active and alternate CICS systems, but not at the same time. Initially, they are opened by the active CICS region, and are only opened by the alternate CICS region during or following takeover. Thus in an XRF environment, passively shared data sets are said to be “owned” by the active CICS system. The CICS system data sets in this category are the:

- CICS system definition data set (DFHCSD)
- CICS Global catalog data set (DFHGCD)
- CICS Restart data set (DFHRSD)
- CICS system log and other journal data sets (DFHJnnx)
- Automatic journal archiving control dataset (DFHJACD)
- Temporary storage data set (DFHTEMP)
- Transient data intrapartition data set (DFHNTRA).

User data sets managed by CICS file control, and DL/I data sets, are also passively shared.

Unique data sets

These data sets are unique to either the active ***or*** the alternate CICS region, and are not shared in any way. The CICS system data sets in this category are:

- CICS local catalog data set (DFHLCD)
- CICS Transaction dump data sets (DFHDMPA and DFHDMPB)
- Auxiliary trace data sets (DFHAUXT and DFHBUXT).

User application data sets are not usually unique.

VSE/VSAM Space Management for SAM feature

The following data sets are supported in VSAM data space managed by the VSE/VSAM Space Management for SAM feature of VSE:

- Auxiliary trace data sets
- CICS Transaction dump data sets
- Transient data extrapartition data sets.

For dump and auxiliary trace data sets, you can only use a primary allocation (using the BLOCKS, CYLINDERS, RECORDS, or TRACKS parameter); there is no support for secondary allocation when the primary allocation for a dump or auxiliary trace data set is full. For the other data sets supported by this feature, secondary extents are created by VSAM when the primary allocation is full.

Journal data sets are the only CICS sequential disk system files ***not*** supported in VSAM data space managed by the VSE/VSAM Space Management for SAM feature.

Chapter 10. Defining the temporary storage data set

This chapter describes how to define the temporary storage data set. CICS provides the **temporary storage** facility to enable application programs to hold data, created by one transaction, for use later by the same transaction or by a different transaction. You save data in temporary storage queues that are identified by symbolic names. Temporary storage queues can be either in main storage or in VSAM-managed auxiliary storage.

You would use main storage if:

- Data is needed for only short periods of time
- Data does not need to be recoverable
- Only small amounts of data are to be stored

You would use auxiliary temporary storage if:

- Large amounts of data are to be stored
- Data is to be kept for extended periods of time
- Data is to be maintained from one CICS run to the next

For background information about CICS temporary storage, see the *CICS Application Programming Guide*.

You define auxiliary temporary storage as a nonindexed VSAM data set. CICS uses control interval processing when storing or retrieving temporary storage records in this data set. A control interval usually contains several records. Temporary storage space within a control interval is reusable.

You do not need to format the data set. It may be redefined while CICS is inactive so that its CI size and space allocation can be changed.

Job control statements to define the temporary storage data set

To define a VSAM data set for auxiliary temporary storage, as a single extent data set on a single volume, you can use the sample job shown in Figure 30 on page 102.

Note: You must not define any extra associations for a temporary storage data set. (Do not, for example, define a PATH.) Doing so causes CICS startup to fail.

```
// JOB CREATETS CREATE AUXILIARY TEMPORARY STORAGE DATA SET
// EXEC IDCAMS,SIZE=AUTO
   DEFINE CLUSTER                          -
            (NAME(CICS410.applid.DFHTEMP)  -
            RECORDSIZE(16377,16377)        -  1
            RECORDS(144)                   -
            NONINDEXED                     -
            CONTROLINTERVALSIZE(16384)     -
            SHAREOPTIONS(2)                -
            VOLUMES(volid))                -
          DATA(NAME(CICS410.applid.DFHTEMP.DATA))   -
          CATALOG(usercat)
/*
/&
```

Figure 30. Coding a single-extent data set on a single volume

Notes for Figure 30:

1 You specify the record size and number of records required on the RECORDSIZE and RECORDS parameters when you define the temporary storage data set. VSAM uses this information and allocates enough disk space to hold at least this number of records. The value of the RECORDSIZE parameter must be 7 bytes less than the value on the CONTROLINTERVALSIZE parameter. See "The control interval size" on page 103 for information about calculating the control interval size.

Using multiple extents and multiple volumes

The job control statements in Figure 30 on page 102 are for a single-extent data set defined on a single volume. That data set must be big enough to hold all your data. Instead of defining one data set, which might have to be much larger than your average needs to cater for exceptional cases, you can define multiple extents and multiple volumes.

When you define more than one extent, CICS uses the extra extents when the primary extent is full. Therefore you can make your primary extent just large enough to meet average demand, and then have smaller secondary extents for overflow. In this way you are saving space until it becomes necessary to use it.

To allocate additional extents on the same volume, code a secondary space argument on the RECORDS parameter like this:

```
RECORDS(primary secondary)
```

where *primary* is the number of records for which space is to be allocated in the primary extent; and *secondary* is the number of records for which space is to be allocated in secondary extents.

To use single extents on multiple volumes, code:

```
RECORDS(primary)  -
VOLUMES(volid1,volid2,volid3,...)
```

For multiple extents on multiple volumes, combine both operands:

```
RECORDS(primary secondary)  -
VOLUMES(volid1,volid2,volid3,...)
```

If you are suffering from a performance bottleneck on a particular volume, single extents on multiple volumes could be the solution.

Space considerations

The amount of space allocated to temporary storage is expressed in two values that you must specify:

1. The control interval size
2. The number of control intervals in the data set.

The control interval size

You specify the control interval size with the CONTROLINTERVALSIZE parameter in the VSAM CLUSTER definition. Because a control interval contains one or more temporary storage records, take the temporary storage record size into account when choosing the control interval size. The following factors affect your choice:

- Each temporary storage record ***must*** have space for:
  - The data
  - 20 bytes (for the temporary storage header)
  - 32 bytes if the storage is defined as recoverable and is requested by an EXEC CICS START TRANSID(name) FROM (data-area) command.

  If you install BMS with 3270 support, the data length of the record is at least as large as the 3270 buffer size. For 3270 terminals with the alternate screen size facility, the data length is the larger of the two sizes.

  The total number of bytes allocated for a temporary storage record is rounded up to a multiple of 64 (for control interval sizes less than, or equal to, 16 384), or a multiple of 128 (for larger control interval sizes).

- The control interval size should be large enough to hold at least one (rounded up) temporary storage record, including 64 bytes of VSAM control information for control interval sizes less than, or equal to, 16 384, or 128 bytes of control information for larger control interval sizes. The maximum control interval size is 32 768 bytes.

  Choose a control interval size large enough to hold the largest normally occurring temporary storage record, together with the VSAM control information. Oversize records are split across control intervals, but this degrades performance.

  CICS statistics will show the number of records that are larger than the defined CI size.

Example

If you use BMS to write a 24 x 80 character screen to temporary storage, the data written occupies 1920 bytes. If you define the temporary storage queue as recoverable, you need a further 32 bytes, with another 20 bytes for the CICS temporary storage header, giving a total of 1972 bytes. Rounding this up to a multiple of 64 gives 1984 bytes. Finally, adding a further 64 bytes of VSAM control information gives a control interval size of 2048 bytes. Typically, the CI size is larger than this, to hold several records possibly differing in size.

Number of control intervals

VSAM uses the RECORDS and RECORDSIZE operands to allocate enough space for the data set to hold the number of records of the specified size. You must code the same value for the two operands of the RECORDSIZE parameter (the average and maximum record sizes), and this value must be 7 bytes less than the CONTROLINTERVALSIZE. In this way, the specified number of VSAM records matches the number of control intervals available to temporary storage management. You thus specify, indirectly, the number of control intervals in the temporary storage data set. (Note that the RECORDS and RECORDSIZE parameters do not correspond to the temporary storage records as seen at the CICS temporary storage interface.)

The number of control intervals to be allocated depends on user and system requirements for temporary storage, up to the maximum number permitted of 65 535. You must consider space for dynamic transaction backout, especially the dynamic logging of DL/I database changes if you are running with DL/I support.

Number of VSAM buffers and strings

You can use the TS system initialization parameter to specify the number of CICS temporary storage buffers up to the maximum of 32 767. The number of buffers that you specify may have an effect on CICS performance, as described in “Performance considerations of TS and TD buffers” on page 97. You should specify a value to suit your CICS region. If you specify **TS=(,0)**, requests for auxiliary temporary storage are executed using main storage.

Job control statements for CICS execution

The file name required by the temporary storage data set is DFHTEMP. For a CICS execution, you need a DLBL statement for DFHTEMP in a label information area, or in the startup job stream, such as:

```
// DLBL DFHTEMP,'CICS410.applid.DFHTEMP',,VSAM,CAT=CICSUCT
```

XRF considerations

The temporary storage data set is a passively shared data set, owned by the active CICS system, but allocated to both the active and alternate CICS systems.

Chapter 11. Defining transient data destination data sets

Data sets used for transient data destinations (queues) can be intrapartition or extrapartition. This chapter tells you how to define transient data destination data sets. The transient data intrapartition data set is a VSAM entry-sequenced data set (ESDS) used for queuing messages and data within the CICS region.

Transient data extrapartition data sets are sequential files, normally on disk or tape; each queue can be used either to send data outside the CICS region or to receive data from outside the region. For background information about intrapartition and extrapartition transient data, see the *CICS Application Programming Guide*.

Messages or other data are addressed to a symbolic queue which you in the destination control table (DCT) with the parameter TYPE=INTRA (for intrapartition queues) or TYPE=EXTRA (for extrapartition queues). The queues can be used as **indirect** destinations to route messages or data to other queues.

For information about coding destination control table entries, see the descriptions of the DFHDCT macro in the *CICS Resource Definition Guide*.

Queues used by CICS

System messages that CICS produces are commonly sent to transient data queues, either intrapartition or extrapartition. The following is a brief description of the queues used by CICS.

CADL	CEDA VTAM resource log, indirect to CSSL.
CAIL	Autoinstall terminal model resource log, indirect to CSSL.
CCPI	CPI Communications message log, indirect to CSSL.
CDUL	Dump message log, indirect to CSSL.
CESE	LE/VSE runtime output.
CESO	LE/VSE runtime output.
CMIG	Migration log to detect use of EXEC CICS ADDRESS CSA commands indirect to CSSL.
CRDI	RDO install log, indirect to CSSL.
CSCC	CICS Client log, idirect to CSSL.
CSCS	Signon/sign-off security log, indirect to CSSL.
CSDL	CEDA command log, indirect to CSSL.
CSFL	File allocation message log, indirect to CSSL.
CSKL	Transaction and profile resource log, indirect to CSSL.
CSML	Signon/sign-off message log, indirect to CSSL.
CSMT	Terminal error message and transaction abend message log, indirect to CSSL.
CSNE	ZNAC-produced messages log, indirect to CSSL.
CSPL	Program resource log, indirect to CSSL.
CSRL	Partner resource log, indirect to CSSL.
CSSL	Message log, direct to file name MSGUSR (all the other general CICS queues are defined as indirect queues to CSSL).
CSTL	Terminal I/O error log, indirect to CSSL.
CSZL	The queue used for Front End Programming Interface (FEPI) messages. You do not have to define this queue if you do not have FEPI installed. Indirect to CSSL.

CSZX The queue used for Front End Programming Interface (FEPI) processing, You do not have to define this queue if you do not have FEPI installed.

All the above queues are defined in sample DCT DFHDCT2$ except FEPI queues CSZL and CSZX which are defined in the copybook DFH0IZRQ.

You should include in your CICS region all of the queues that CICS uses. Although the omission of any of the queues does not cause a CICS failure, you lose important information about your CICS region if CICS cannot write its data to the required queue. The sample destination control table DFHDCT2$ contains definitions of all the queues that CICS uses, and you can use this as the basis of your own DCT. The sample table and the included copybooks are supplied in the VSE/ESA sublibrary PRD1.BASE. For information about the queues used by CICS, see the *CICS Resource Definition Guide*.

For information about the queues that CICS uses for RDO, see “RDO command logs” on page 142.

For a way of printing these system messages on a local printer as they occur, see the transient data write-to-terminal sample program, DFH$TDWT. This sample program is supplied pregenerated in the VSE/ESA sublibrary PRD1.BASE; the assembler source is also in PRD1.BASE. For programming information about DFH$TDWT, see the *CICS Customization Guide*.

Defining the intrapartition data set

To create a transient data intrapartition data set, you code the required DEFINE CLUSTER parameters, and run an IDCAMS job. Figure 31 gives an example of a job you could use, showing the data space defined as a number of records. Note that if you allocate space for records in this way, rather than as a number of tracks or cylinders, you must also code a RECORDSIZE parameter. The value of the RECORDSIZE parameter must be 7 bytes less than the value of the CONTROLINTERVALSIZE parameter.

```
// JOB DEFINTRA
* ---------------------------------------------
* Create transient data intrapartition data set
* ---------------------------------------------
// EXEC IDCAMS,SIZE=AUTO
 DEFINE CLUSTER                          -
          (NAME(CICS410.applid.DFHNTRA)     -
           RECORDSIZE(4089,4089)         -
           RECORDS(144)                  -
           NONINDEXED                    -
           CONTROLINTERVALSIZE(4096)     -
           VOLUMES(volid))               -
        DATA                             -
          (NAME(CICS410.applid.DFHNTRA.DATA))      -
        CATALOG(usercat)
/*
/&
```

Figure 31. Sample JCL to create a transient data intrapartition data set

You must not define any extra associations for a transient data intrapartition data set. (Do not, for example, define a PATH.) Doing so causes CICS startup to fail.

Using multiple extents and multiple volumes

The job control statements in Figure 31 on page 106 are for a single extent data set defined on a single volume. That data set must be big enough to hold all your data. Instead of defining one data set, which might have to be much larger than your average needs to cater for exceptional cases, you can define:

- Multiple extents on one volume
- One extent on several volumes
- Multiple extents on multiple volumes

For more information about defining these, see “Multiple extents and multiple volumes” on page 96.

Space considerations

Space is allocated to queues in units of a control interval. The first CI is reserved for CICS use, the remaining CIs are available to hold data. Data records are stored in CIs according to VSAM standards.

The CONTROLINTERVALSIZE parameter must be large enough to hold the longest record, plus the 32 bytes that CICS requires for its own purposes. The maximum control interval size is 32 768 bytes.

Size of the intrapartition data set

- Make the data set large enough to avoid a NOSPACE condition. If a queue has the reusable queue space option, a control interval allocated to the queue is released for reuse when all records stored in that control interval have been read by EXEC CICS READQ TD requests.

  If all available control intervals are currently allocated to queues, further EXEC CICS WRITEQ TD requests receive a NOSPACE response until control intervals are released by GET requests. Space is also released by an EXEC CICS DELETEQ TD request, regardless of whether the queue is defined as reusable.

- The intrapartition data set should hold at least two control intervals.

Intrapartition data set restriction

A transient data intrapartition data set should be associated with one, and only one, CICS region. (If you are running CICS with XRF, one region means a pair of active and alternate CICS regions.) The destination control table contains relative byte addresses (RBAs) of records written to an intrapartition data set, and care must be taken to preserve the RBAs during any VSAM export or import operation on the data set.

Data can be corrupted or lost if you:

- Start CICS with the wrong intrapartition data set; that is, a data set that contains data from another CICS region.

- Use VSAM IDCAMS services to increase the control interval size.

Number of VSAM buffers and strings

You can use the TD system initialization parameter to specify the number of CICS transient data buffers up to the maximum of 32 767. The number of buffers that you specify may have an effect on CICS performance, as described in “Performance considerations of TS and TD buffers” on page 97. You should specify a value to suit your CICS region.

Job control statements for CICS execution

The file name for the intrapartition data set is DFHNTRA, and the file-id operand must be the name of the VSAM entry-sequenced data set. For a CICS execution, you need a DLBL statement such as:

```
// DLBL DFHNTRA,'CICS410.applid.DFHNTRA',,VSAM,CAT=CICSUCT
```

XRF considerations

A transient data intrapartition data set is a passively shared data set, owned by the active CICS region, but allocated to both active and alternate CICS systems.

Defining extrapartition data sets

Extrapartition data sets are defined by DFHDCT TYPE=SDSCI entries in the Device Control Table. You can define each extrapartition data set either for input only or for output only, but not for both.

You should define transient data extrapartition data sets used as queues for CICS messages (for example, SDSCI for MSGUSER) with a RECSIZE of 132 bytes and a RECFORM of either

- VARUNB (in which case BLKSIZE=136), or
- VARBLK.

Unlike older releases of CICS/VSE, device independence for extrapartition data sets is provided. The SDSCI entries do not statically create VSE DTFs. Instead, the DTFs are built dynamically at run-time based on VSE JCL and data set characteristics (for example, BLKSIZE, RECSIZE and RECFORM) from the DCT.

Subject to certain constraints (see “Restrictions” on page 109), this effectively allows the user to change the DFHDCT TYPE=SDSCI macro DEVICE= operand at run time without requiring a reassembly of the DCT.

For example, the DCT could define a destination as a printer and, at run time, the JCL could be set up to write the records to any of the following:

A sequential disk - supply a DLBL, EXTENT and ASSGN to disk
A labelled tape - supply a TLBL and ASSGN to a tape
An unlabelled tape - supply an ASSGN to a tape
A printer - supply an ASSGN to a printer

With one exception (see “Restrictions” on page 109), other DCT-defined characteristics cannot be changed at run-time. In the above example relating to a printer, the records will still be unblocked and either variable or fixed format no matter which of the device types is selected. Therefore, do **not** attempt to assign a printer to a SDSCI entry that describes a data set with blocked records, as this will result in a failure to build the DTF, the data set will be forced into a closed state and subsequent opens will fail.

Operands that are not relevant for the selected device type will be ignored. For example, if the DCT entry defines a tape device, and at run time a disk is selected, the tape-related operands FILABL=, REWIND=, TPMARK= and TYPEFLE=RDBACK are ignored.

Apart from one restriction with DEVADDR (see "Restrictions"), the device type is selected by using the DSCNAME to search through the user label, partition or class standard label area and finally the standard label area:

- If a DLBL is found, a DTFSD is built.
- If a TLBL is found, a DTFMT for labelled tape is built.
- Otherwise, the DTF type is selected based on the device class that is assigned to the DCT-defined DEVADDR logical unit (ie. to select an unlabelled tape or spool device type)

Restrictions

The following restrictions apply to changing operands with the run time JCL

- Only the SDSCI DEVICE= operand can be changed at run time;

  Note: For a CKD or ECKD™ device, the DLBL BLKSIZE= operand can be used to change the BLKSIZE value coded in the DCT.
- If DEVADDR is not coded in the SDSCI entry, a DTFSD is always built, as only the DTFSD permits DEVADDR to be omitted.

Defining extrapartition data sets on disk

You can use the example shown in Figure 32 for the extrapartition data set entries in the DCT.

```
//  DLBL   filname,'dataset.name',0,SD 1 2
//  EXTENT SYSnnn,volid,1,0,rtrk,ntrks 3 4
//  ASSGN  SYSnnn,cuu                  5
```

Figure 32. Sample JCL for an extrapartition data set on disk

Notes for Figure 32:

1 File name *filname* in the DLBL statement must be the same as the name defined on the DSCNAME operand of the DFHDCT TYPE=SDSCI macro.

2 *dataset.name* is the name of the extrapartition data set.

3 *nnn* is the logical unit number.

4 *rtrk* is the relative track number for the start of the extent (or relative block number if you are using an FBA device), and *ntrks* is the number of tracks allocated for the extent (or number of blocks allocated for the extent if you are using an FBA device).

5 *cuu* is the unit address.

Defining extrapartition data sets on tape

For transient data sets on tape, you can use the same sample job statements shown in Figure 32 on page 109. However, the ASSGN statement should be as follows:

```
//  ASSGN  SYS009,cuu
```

where the DFHDCT TYPE=SDSCI macro **must** specify DEVADDR=SYS009. If you require a labelled tape, a TLBL job control statement is also required.

The DFHCXRF data set

Besides any extrapartition data sets that you might define using DFHDCT macros, there is a special extrapartition queue that CICS creates dynamically. This has the destination identifier CXRF, and is created by CICS early in the initialization process. The file name for this extrapartition data set is DFHCXRF. You cannot define CXRF or DFHCXRF as operands in the DCT. If you code DFHCXRF as a DSCNAME, or code CXRF as a DESTID, the DFHDCT macro generates an MNOTE 8 error message.

Although this data set has special significance in an alternate CICS system when you are operating CICS with XRF, it is also available in an active CICS system, and in CICS systems running with XRF=NO.

DFHCXRF data set in active CICS systems

CICS uses the CXRF queue during:

- CICS startup, for any CICS components that need to write to transient data queues before transient data initialization has ended. Requests to write to any CICS transient data queue (DESTID=Cxxx) before the queue is ready are redirected to DFHCXRF. Requests from CICS components to write CICS transient data before even CXRF has been created fail with a QIDERR condition.
- CICS shutdown, after DFHSTP has flushed the intrapartition transient data buffers. Requests to any CICS intrapartition transient data queue after this point are redirected to DFHCXRF. This is because once the buffers have been flushed to DFHNTRA, the DCTEs for the intrapartition queues must not be modified before they are warm-keypointed by DFHWKP later in the shutdown process.'

If you want to take advantage of the special CXRF queue, you must include a DLBL statement for DFHCXRF. (For example, see Figure 33 on page 111.) If you omit the DLBL statement, transient data write requests redirected to CXRF fail with a NOTOPEN condition.

DFHCXRF data set in alternate CICS systems

DFHCXRF has special significance for alternate CICS systems, because transient data initialization is suspended while the alternate CICS system is in standby mode, and is not completed until a takeover occurs. If you want to capture messages written to transient data by the alternate CICS region before takeover, you must include a DLBL statement defining the DFHCXRF data set. These messages include information about terminals that have been installed and logged on to the active CICS region. The alternate CICS region takes this information from the message data set and stores it in the CICS-generated extrapartition transient data queue, CXRF.

DLBL statements for the DFHCXRF data set

You can define the DFHCXRF data set to VSE in the same way as other transient data extrapartition data sets, as a sequential data set on disk, tape or printer. For example, you could use the DLBL statements shown in Figure 33 in a startup job stream for an alternate CICS system for disk or tape, or those shown in Figure 34 for a printer.

```
// DLBL DFHCXRF,'CICS410.applid.DFHCXRF',0,SD
// EXTENT SYS020,SYSWK1,1,0,2000,5
// ASSGN SYS020,DISK,VOL=SYSWK1,SHR
```

Figure 33. Sample DLBL statements for DFHCXRF for disk or tape

```
// DLBL DFHCXRF,'CICS410.applid.DFHCXRF',0,SD
// EXTENT SYS020,SYSWK1,1,0,2000,5
// ASSGN SYS020,SYSLIST
```

Figure 34. Sample DLBL statements for DFHCXRF on a printer

Before takeover occurs, the alternate CICS system assumes that the transient data queues are defined as indirect, and pointing to CXRF. CXRF is associated with the data set that has the data set name DFHCXRF.

XRF considerations

Except for DFHCXRF, an alternate CICS system does not open any extrapartition data sets before takeover. (See “The DFHCXRF data set” on page 110.)

Normally, when data sets are defined for output, you should have separate data sets for the active and alternate CICS systems; that is, they are unique data sets in CICS terms.

Data written by the active CICS system is lost when the alternate CICS system takes over and opens the data set.

Transient data extrapartition data sets as VSAM-managed SAM files

CICS extrapartition data sets are also supported as SAM files managed in VSAM space, using the VSE/VSAM Space Management for SAM feature.

Job control statements for defining VSAM-managed SAM files

For examples of defining CICS data sets that are supported in VSAM-managed space, see the descriptions given for the dump data sets in “Dump data sets as VSAM-managed SAM files” on page 166. An example of the DLBL statement for cataloging a parameterized procedure is given in Figure 35 on page 112.

There are also examples of the DLBL statements for VSAM-managed SAM files shown in the example of the startup job stream in “Sample startup job stream” on page 290.

```
// DLBL MSGUSR,'%user.msgusr',0,VSAM,CAT=CICSUCT
               RECORDS=(25,25),RECSIZE=2048
```

Figure 35. Sample DLBL statement for cataloging a parameterized procedure.

Chapter 12. Defining data sets for journaling and archiving

This chapter describes how to create and initialize the following data sets, which CICS uses for journaling in a running CICS region:

System log data sets	CICS uses these to record changes to recoverable resources
User journal data sets	These are used by your own applications, and for auto-journaling file control, terminal control, and forward recovery logs in file control
A journal archive data set	For use by the CICS automatic journal archiving facility
A VSE sublibrary	For the members containing the skeletal JCL used by the archiving job submission program

If you want CICS to write journal records, you must create and initialize the journal data sets you need. Each journal data set (or each journal tape) must also be defined with an operating system data definition statement in the CICS startup job stream. The DLBL statements defining data sets for CICS journals have the form DFHJnnA or DFHJnnB, where “nn ” is the journal indentifier. For all journals except the CICS system log, the journal identifier in CICS journal DLBL statements corresponds to the JFILEID operand that you specify in the journal definition entries in the journal control table (JCT).

For the system log you must specify JFILEID=SYSTEM in the JCT, and this corresponds to a journal identification of 01 in the DLBL statement. For information about how to define journals in the JCT, see the *CICS Resource Definition Guide*.

The CICS system log can be used to return recoverable resources to their committed state after an abnormal termination of CICS. For information about this use of the CICS system log, see the *CICS Recovery and Restart Guide*.

Defining and formatting CICS journals on disk

You define disk journals on sequential data sets that are reused for output when full. This is true whether a disk journal is defined in the JCT as one or two data sets (JTYPE=DISK1 or JTYPE=DISK2). You must also format the required disk data sets for journal output before CICS can use them for the first time. Once they have been formatted, data sets are reused for successive CICS executions. During initialization, CICS determines which data sets to open for journaling, and where to begin journaling within the data sets, according to:

- The options defined in the JOUROPT parameter in the JCT
- The type of start CICS is performing
- The status of the journal data sets, in the CICS global catalog.

For information about where CICS begins recording on journal data sets, see the *CICS Recovery and Restart Guide*.

The CICS system log is sensitive to the physical characteristics of the DASD it resides on, and cannot be moved to another type of DASD. If you move your CICS data sets from one type of DASD to another (for example, from 3380 devices to 3390 devices), you must:

- Preformat new copies of your journal data sets (on the new DASD) by using the journal formatting utility, DFHJCJFP. (See “Preformatting journal data sets.”)
- Perform a cold start of CICS.

Preformatting journal data sets

You preformat each data set by running the DFHJCJFP utility program. If you do not preformat a journal data set, CICS may not be able to open it. In the unlikely event that CICS does open an unformatted journal data set, the wrong data may appear in the journal.

Attention! Do not reformat journal data sets used by CICS

Except for the CICS system log emergency data set, do not format journal data sets again after they have been used by CICS, unless you intend to use them for another purpose, or have a particularly good reason for needing to reformat them. Reformatting destroys the contents of a journal data set, and if you reformat the system log, CICS cannot use it in an emergency restart.

However, you should format any journals that have been used in a previous run of CICS with a date/time stamp greater than the current date before they are used with the current (lower) date/time. This is to ensure that no residual log records from earlier runs of CICS are left with a higher date/time stamp, which may cause any subsequent emergency restart to fail.

Resetting the journal status

CICS maintains a record of the status of disk journal data sets in one of two ways. These are:

1. If you are using the CICS automatic journal archiving facility, the status (open, ready for use, or not ready for use) is maintained in the journal archive control data set (DFHJACD).
2. If you are not using automatic journal archiving, the status is maintained in the CICS global catalog.

If you want to change back from using automatic journal archiving to manual archiving you must ensure that, when the change takes place, the appropriate journal data sets have already been archived using the automatic archiving JCL. This ensures that the status on the JACD data set is set to “READY FOR USE”, and that no unexpected automatic archive jobs will be submitted during the next CICS startup.

If you are managing your own archiving of disk journals, and you specify JOUROPT=PAUSE, CICS issues message DFHJC4583 when a journal disk data set is full and ready for copying. If you fail to reply to DFHJC4583 messages, it is possible for both data sets of a journal to have a status of “NOT READY” at CICS initialization. If this occurs, you can reset the journal status by specifying the system initialization parameter, JSTATUS=RESET. However, you must ensure that ***all*** journal data sets have been copied before you specify the JSTATUS parameter. Specifying RESET causes the status of all journal data sets to be set to “READY

FOR USE”, and the journals repositioned according to the rules in the *CICS Recovery and Restart Guide*.

Job control statements for formatting journals on disk

The sample job in Figure 36 formats the journal data sets of a two-data-set system log on disk.

```
// JOB CICSFDSK TO FORMAT JOURNAL DISK EXTENTS
// DLBL JOURNAL,CICS410.applid.DFHJ01A,0,SD   1 2
// EXTENT SYSnnn,volid,1,0,rtrk,ntrks
// ASSGN SYSnnn,cuu
// LIBDEF PHASE,SEARCH=PRD1.BASE
// EXEC DFHJCJFP,SIZE=DFHJCJFP
/*
// DLBL JOURNAL,CICS410.applid.DFHJ01B,0,SD   1 2
// EXTENT SYSnnn,volid,1,0,rtrk,ntrks
// ASSGN SYSnnn,cuu
// EXEC DFHJCJFP,SIZE=DFHJCJFP
/&
```

Figure 36. Formatting a journal data set

Notes for Figure 36:

1 The DLBL name for the journal data definition statement must be JOURNAL.

2 For FBA devices, you can specify a CISIZE operand on the DLBL statement. If you specify a CISIZE for the journal, it must be at least 7 bytes greater than the value specified on the BUFSIZE operand of the DFHJCT TYPE=ENTRY macro. Furthermore, a CISIZE value must be either a multiple of 512 bytes when ≤8192 bytes, or a multiple of 2048 bytes when >8192 bytes.

Note that, if you specify a CISIZE on a DLBL statement when formatting a journal, and you specify a CISIZE in your CICS startup job stream, these values must be the same. For more information about the CISIZE operand, see the *VSE/ESA System Control Statements* manual.

General notes:

- The system log is defined on 2 data sets (JTYPE=DISK2).
- A user journal is defined with journal identification 2 (JFILEID=2) and on a single data set (JTYPE=DISK1).
- Change the space allocations to suit your installation’s needs, but you must allocate at least 3 tracks.
- The maximum block size for disk journal data sets is 32760 bytes, and is set in its DTF by the DFHJCJFP program. Therefore, do ***not*** code the BLKSIZE operand on the DLBL statements to specify a block size for the journal data sets. (Specifying the BLKSIZE operand can lead to unpredictable errors when the journal is used subsequently, such as in an emergency restart.) You limit the actual size of the journal blocks by coding a BUFSIZE operand in the JCT entry for the journal. However, the actual size of the blocks written to journals by CICS can vary widely, up to the limit set by the BUFSIZE operand. For information about specifying the BUFSIZE parameter, see the *CICS Resource Definition Guide*.

Space considerations

If you choose to have the system log on disk, you must allocate enough space for each data set. For a successful restart, the system log needs to be large enough to hold, at any one time, at least one complete keypoint. It must also hold all data logged from the start of the task (or logical unit of work) to the moment of failure, for each task that does logging.

Minimum space allocation

You must preallocate at least 3 tracks of disk space for each data set used for journaling, and all of the disk space within each data set must be contiguous. If you do not allocate at least three tracks, the formatting job fails with message DFHJC4598 (insufficient space).

In practice, allocate enough space in each data set to hold at least two to three times the amount of data logged during the processing of the longest task. Also specify the activity keypoint frequency (AKPFREQ) as a system initialization parameter so that CICS records at least two or three keypoints while writing to each system log data set. You may decide to allocate enough space on the combined log data sets to avoid having to switch more than once during a CICS run.

Variables that affect the system log's space requirements include:

- The loading on the system log. This, in turn, depends on parameters chosen during CICS table preparation, such as recoverable destination control table destinations and user journaling.
- The duration of the longest-running task.
- The frequency at which keypoints are written.
- The buffer size of the system log.
- The frequency of WAIT requests.
- The track capacity of the device on which the system log resides.

Job control statements for CICS execution

If you catalog and format the journal data sets on disk as shown in Figure 36 on page 115, the data definition statements for the CICS execution are as shown in Figure 37.

```
// DLBL DFHJ01A,'CICS410.applid.DFHJ01A',0,SD  1  2  3
// EXTENT SYSnnn,volid,1,0,rtrk,ntrks             4
// ASSGN SYSnnn,cuu                               5
// DLBL DFHJ01B,'CICS410.applid.DFHJ01B',0,SD  1  2  3
// EXTENT SYSnnn,volid,1,0,rtrk,ntrks             4
// ASSGN SYSnnn,cuu                               5
```

Figure 37. Job control statements for CICS execution

Notes for Figure 37:

1 The file name on the DLBL statement must be of the form DFHJnnx, where "nn" is the journal identifier for a particular JCT entry and "x" is A or B. For the CICS system log, the journal identifier is 01.

2 You must provide either one DLBL statement with file name DFHJnnA, or two DLBL statements with file names DFHJnnA and DFHJnnB, for each JCT entry, depending on whether the JCT entry specifies JTYPE=DISK1 or JTYPE=DISK2.

3 For FBA devices, you can specify a CISIZE operand on the DLBL statement. If you specify a CISIZE, it must be at least 7 bytes greater than the value in the BUFSIZE operand in DFHJCT TYPE=ENTRY macro. Furthermore, a CISIZE value must be either a multiple of 512 bytes when ≤8192 bytes, or a multiple of 2048 bytes when >8192 bytes. It must also be the same CISIZE value specified, if any, to format the journal when running the journal formatting utility (DFHJCJFP).

4 and **5** Although the DEVADDR operand on the DFHJCT TYPE=ENTRY macro allows you to specify a logical unit name for the journal data set, you can override this in the EXTENT job control statement. Thus, the logical unit name specified in the DEVADDR operand of the JCT can be different from that specified in the ASSGN job control statement.

General note:

Any disk logs that you may want to use for forward recovery **must** be on a different disk device from your recoverable data sets.

Defining and formatting CICS journals on tape

Tape journals generally consist of a series of physical reels (tape volumes) used in sequence. The volumes are mounted on either one or two tape drives, according to the JTYPE parameter in the JCT and the corresponding DLBL statements.

Because only unlabeled tapes may be used, it is necessary to label each tape used with an external (paper) label and to record the identity and sequence of tape volumes constituting the journal separately.

Include corresponding job control statements in your CICS start up job stream. The statements must have the file names in the form DFHJnnx, where “nn” is the journal identifier, and “x” may be either A or B. For tape journals, A and B correspond to whichever pair of tape drives you are defining. If your journal consists of a single data set, “x” has the value A. If the journal you are defining is the system log, “nn” has the value 01.

For information about how to define CICS journals on tape, see the **JCT chapter** in the *CICS Resource Definition Guide*.

During emergency restart, the CICS recovery utility program, DFHRUP, must find the last record written. It needs to have this record to process the system log from the point where it was positioned when the system terminated abnormally. If new tapes are used in system logs and an abnormal termination causes no end-of-file mark to be written, difficulties occur in finding the last record written; for example, the tape may run off the end of the reel.

To avoid this, use the CICS-provided utility program, DFHFTAP, to preformat the tapes before they are used as CICS journals for the first time. Do not run the DFHFTAP utility program on tapes if they have been formatted before and contain journal records that you might need.

Job control statements for formatting tapes for journal use

If you are using standard-labeled tapes, you can use the sample job shown in Figure 38 to format a system log or journal data set on tape.

```
// JOB CICSFTAP FORMAT JOURNAL TAPE
// ASSGN SYS010,cuu  1
// MTC WTM,SYS010    2
// MTC REW,SYS010
// LIBDEF PHASE,SEARCH=PRD1.BASE
// EXEC DFHFTAP
/*
/&
```

Figure 38. Sample to format a system log or journal data set on tape

Notes for Figure 38:

1 The logical unit name must be SYS010 for program DFHFTAP.

2 If the tape you are formatting is a new (unused) tape, make sure that a tape mark is written to the tape before it is processed by DFHFTAP. You can use either the // MTC WTM,SYS010 job control statement, or a utility program such as DITTO, to write the tape mark."

Job control statements for CICS execution

If you activate journaling for a CICS execution, you have to assign I/O devices for each entry of the JCT selected.

For example, if the system log has been specified as:

```
JTYPE=TAPE2,DEVADDR=(SYS004,SYS005)
```

the job control statements would be:

```
// ASSGN   SYS004,cuu
// ASSGN   SYS005,cuu
```

Note that the logical unit names in the ASSGN statements for tape journals ***must*** be the logical unit names specified in the DEVADDR operand.

Writing an EOF mark on a journal or system log tape

You can use the end-of-file utility program, DFHTEOF, offline to reposition and correctly close a journal tape after an uncontrolled shutdown.

CICS also uses the DFHTEOF utility program during a CICS restart. In an emergency restart, CICS checks the closed status of the system log recorded in the global catalog. If the global catalog shows that the system log was successfully closed following the uncontrolled shutdown, the DFHTEOF utility program does not need to search for the end of the log.

However, if the global catalog does not show that the log was closed, the DFHTEOF utility program writes an end-of-file (EOF) mark after the last valid record written before the failure. It does this by reading the tape forward from the load point and checking the label record in each journal block.

If the system log (or any other CICS-created tape journal) is to be used by DL/I or by a user-written program that does not include such label-checking logic, you must write an EOF mark after the last valid record. Do this even when a partition ABEND caused the abnormal termination, in case the ABEND routine itself failed to close the tapes.

A sample job stream to run the DFHTEOF utlity program is shown in Figure 39.

```
// JOB CICSTEOF WRITE EOF ON JOURNAL TAPE
// ASSGN SYS010,cuu  1
// LIBDEF PHASE,SEARCH=PRD1.BASE
// EXEC DFHTEOF
/*
/&
```

Figure 39. Sample JCL to run DFHTEOF (stand-alone)

Note for Figure 39:

1 The logical unit name on the ASSGN statement for the tape must be SYS010. Ensure that the correct volume (that is, the last volume that was mounted for system log output during the previous execution) is mounted and positioned at its load point.

User journals using DMF

CICS user journals that are for output only (but not the system log) can optionally record data on DMF data sets using the SMF 110 type record format. This facility is an alternative to writing journal data to CICS disk or tape journal. To do this, you must create the necessary JCT entries by specifying FORMAT=SMF and JTYPE=SMF on the DFHJCT TYPE=ENTRY macro.

For programming information about SMF format, see the *CICS Customization* guide.

XRF considerations

When running CICS with XRF=YES, you must define the system log as a disk journal with two data sets; otherwise CICS abends.

The active and alternate CICS regions must refer to the same system log data sets because the alternate uses information in the log to effect its emergency restart. While the alternate is in standby mode it does not require the system log, but opens the log during emergency restart following a takeover.

Your user journals may be on disk or tape. The user journals are opened in the usual way during the emergency restart that forms the takeover process.

Although your user journals can be on disk or tape, if your CICS region depends heavily on user journals, consider the following points:

Disk journals
The use of disk journals provides faster switching, but unless you are using automatic journal archiving (JOUROPT=AUTOARCH) it could be at the expense of operator control over archiving.

With disk journaling, you should archive each data set before reuse to preserve journal data that might be required for batch backout or forward recovery purposes. This archive must be done either by the operator, or automatically by the CICS automatic archiving program. (See “Journal archive data sets” on page 120.) To allow for a fast switch at takeover time, you are strongly recommended to define two data sets for each disk journal.

(If you define a single data set for use as a disk journal, the data set must always be archived when it becomes full, before the alternate can takeover and reuse it. The time taken to archive the data set creates an unacceptable delay during the takeover period.)

The data sets are allocated at job step initiation. The need to archive journals means that this is the normal status.

Tape journals
Tape journals provide implicit archiving at the expense of slower switching. Although tape journaling covers the question of archiving, there is additional operator involvement and time required to make the tape journal available to the alternate. There is also the manual switching of tape units, and the movement of the journal tapes to consider. For these reasons tape journaling is probably unacceptable where you want a fast takeover.

Journal archive data sets

You can specify that you want CICS to archive a disk journal data set automatically when it is closed for output, but only when you have defined the journal in the JCT with two data sets (JTYPE=DISK2). The automatic journal archiving facility caters for all the data sets defined for a journal.

You request automatic journal archiving using the DFHJCT TYPE=ENTRY macro, by specifying the AUTOARCH option on the JOUROPT parameter.

The automatic journal archiving facility does not normally require any action by an operator, except to mount tapes if the archive is to tape. The only time that an operator needs to intervene is if an automatic journal archiving job has failed for some reason, and as a result a journal data set is not ready for use when CICS tries to open it for output. If this occurs, CICS prompts the operator with message DFHJC4542D or DFHJC4543D, which may be followed by DFHJC4544D, depending on the reply to the other message. For information about how to respond to these messages, see the *VSE/ESA Messages and Codes Volume 3* manual.

If you specify JOUROPT=AUTOARCH, CICS requires the following data sets:

- A **journal archive control data set** (DFHJACD), which must be defined as a VSAM relative-record data set (RRDS)
- A **journal archive JCL sublibrary**, each member of which contains skeletal JCL for use by the automatic archive job submission program

Defining the journal archive control data set (JACD)

You can use the sample job in Figure 40 on page 122 to create a JACD.

CICS uses the JACD to store control information about the journal data sets. The JACD must have the following attributes:

- **Share option 4**

  This allows both CICS and the batch archive job to update the data set, and also ensures that the data set buffers are refreshed for each access. This data set is read only infrequently, and refreshing the buffers each time is not a performance consideration, but it is important for ensuring data integrity.

- **Control interval (CI) size of 512 bytes**

  This ensures that each JACD record is also a physical block.

- **Record size of 505 bytes**

  This record size is the CI size minus 7 (the 7 bytes required for VSAM CI information), and ensures that the control interval contains only one JACD record.

- **Number of records as 198**.

  This allows one record for each of the possible disk journal data sets that can be defined to CICS.

You do not have to initialize the JACD; CICS initializes it for you. For each JCT entry defined with JOUROPT=AUTOARCH, records are added or updated as follows:

- If there is already an entry for the journal data set on the JACD, and the journal data set name and the ARCHJCL name is the same as in the JCT, CICS does not change the record.
- If there is an entry on the DFHJACD, but the journal data set name or the ARCHJCL name has changed, CICS updates the record and issues message DFHJC4539.
- If there is no entry on the JACD for the journal data set, CICS adds a new record.

Note: All added and updated records are given a status of READY.

CICS ***never*** deletes records from the JACD.

```
// JOB DEFJACD
/* DEFINE THE JOURNAL ARCHIVE DATA SET
// EXEC IDCAMS,SIZE=AUTO
        DEFINE CLUSTER                        -
              (NAME(CICS410.applid.DFHJACD)      -
               NUMBERED                        -
               RECORDS(198)                    -
               RECORDSIZE(505 505)             -
               CONTROLINTERVALSIZE(512)        -
               SHAREOPTIONS(4)                 -
               VOLUMES(CIC410))                -
               DATA                            -
                (NAME(CICS410.applid.DFHJACD.DATA) ) -
                 CATALOG(usercat)
   /*
   /&
```

Figure 40. Sample job to define a journal archive data set

Block size when archiving

Do not try to change the block size of journal data sets when you copy them. CICS journal control writes a sequence number at the start of each block, and this is essential for the operation of the batch backout operation. If you reblock the journal data, the sequence number is not at the start of each block as before.

Defining a journal archive JCL sublibrary

The journal archive JCL sublibrary contains skeletal JCL for use by the journal archive submission program, DFHJASP. DFHJASP uses the appropriate member for each journal, obtaining the member name from the ARCHJCL parameter in the JCT and where the member type must be DFHJASP.

You can use the sample job shown in Figure 41 to create a suitable sublibrary and also copy into it the CICS supplied sample JCL DFH$ARCH from the VSE/ESA sublibrary PRD1.BASE.

```
// JOB CRTARCH
// EXEC LIBR
   DEFINE SUBLIB=CICS410.ARCHJCL REUSE=IMM
   CONNECT S=PRD1.BASE:CICS410.ARCHJCL
   COPY    DFH$ARCH.J:DFH$ARCH.DFHJASP
/*
/&
```

Figure 41. Sample job to copy CICS-supplied skeletal JCL

You can see from the sample skeletal JCL in DFH$ARCH, that the JCL you write for use by the automatic journal archive submission program, DFHJASP, should be written using symbolic parameters (for example, those defined with the percent symbol, like %JJ in the sample). These parameters enable you to use a single skeletal JCL source file to archive all the journals defined to your CICS regions. DFHJASP resolves the symbolic parameters into the correct values for the journal it is archiving at the time it submits the job to POWER.

For information about the sample skeletal JCL, the symbolic parameters that are available to you, and the journal archive utility program, DFHJACDU, see the *CICS Operations and Utilities Guide*.

Job control statements for CICS execution

If you specify JOUROPT=AUTOARCH, you must include a DLBL statement for the journal archive control data set (DFHJACD) in the CICS startup job stream.

For example:

```
// DLBL DFHJACD,'CICS410.applid.DFHJACD',,VSAM,CAT=CICSUCT
```

You also need to define the sublibrary containing the journal archive JCL.

For example:

```
// LIBDEF SOURCE,SEARCH=(archjcl.library)
```

Journal utility programs (DFHJUP and DFHJACDU)

CICS provides two journal utility programs:

DFHJUP
The CICS journal utility program. You can use the DFHJUP utility program to select, print, or copy data held on CICS journal data sets.

DFHJACDU
If you are using the CICS automatic archiving facility, the DFHJACDU utility program is used to check, and then update, the journal archive control data set (DFHJACD).

For information about these utility programs, see the *CICS Operations and Utilities Guide*.

Chapter 13. Defining data sets for the Data Management Facility (DMF)

In CICS Transaction Server for VSE/ESA Release 1 monitoring and statistics data are not handled by journal control; instead the data is written to CICS Data Management Facility (DMF) Data Handler data sets.

DMF provides similar data capture facilities and functions as the MVS/ESA™ System Management Facility (SMF).

DMF runs in its own VSE partition, collecting and monitoring data passed to it from any CICS system runnning under the same VSE image, and storing the data in a VSE data space.

DMF writes records from this data space to one of thirty six DMF data sets. This chapter describes how to create and initialize one of these DMF data sets

Attention!

There is no alternative data repository to DMF. If DMF is not active, monitoring and statistical data is lost.

Creating DMF data sets

You must define each DMF data set to VSAM:

- As an ESDS file of fixed allocation
- Allow for spanned records with a maximum length of 32 767 bytes
- Define each data set as REUSABLE.

You can choose the names for your DMF data sets, which are kept in an assembled table, DFHDMFSU, together with other default parameter values used to control DMF itself.

See the *CICS Operations and Utilities Guide* for information about the different parameters you can use to control DMF.

Sample job to create a DMF data set

Figure 42 on page 126 shows a sample job for creating a DMF data set.

```
// JOB CREATE A DMF DATA SET
// EXEC IDCAMS,SIZE=AUTO
   DEFINE CLUSTER                                -
            (NAME(CICS410.SYS1.MANZ)             -
            NONINDEXED                           -
            CYLINDERS(10)                        -
            REUSE                                -
            RECORDSIZE(125,32767)                -
            SPANNED                              -
            CONTROLINTERVALSIZE(4096)            -
            SHAREOPTIONS(2)                      -
            VOLUME(volid))                       -
          DATA(NAME(CICS410.SYS1.MANZ.DATA))     -
          CATALOG(usercat)
/*
/&
```

Figure 42. Sample job to create a DMF data set

Formatting DMF data sets

To record DMF data, you must define the DMF data sets. A minimum of two data sets should be available for use by DMF. You can allocate up to 36 DMF data sets. DMF routines fill these data sets one at a time. While DMF is writing data to one data set, it can write out or clear others. When the current data set becomes full, DMF is able to continue writing records for as long as it can find an empty inactive data set.

Job control statements to format DMF data sets

To preformat DMF data sets, use the JCL shown in Figure 43 on page 126. If you do not preformat your data sets, DMF cannot use them. DMF preformats the data sets with dummy records.

```
// JOB FORMAT
// DLBL NEWDS,'CICS410.SYS1.MANZ',,VSAM,CAT=usercat
// EXEC DFHDFOU
   INDD(NEWDS,OPTIONS(CLEAR))
/*
```

Figure 43. JCL to preformat DMF data sets

Space considerations

CICS can write records to DMF of up to 32 767 bytes, resulting in DMF writing spanned records to the DMF data sets.

For a more efficient use of DASD, you should consider creating the DMF data sets to be used by CICS with a control interval (CI) size of either 16 384 bytes (16KB) or 8192 bytes (8KB). For example, a CI size of 18KB on a 3390 gives 98% track usage.

If you use other CI sizes, you must consider the trade off between the efficient use of DASD, DMF data set I/O performance and the possibility of data being lost owing to insufficient DMF buffers.

For more information about CICS statistics and monitoring records and their sizes, see the *CICS Performance Guide*.

Chapter 14. Defining the CICS system definition data set

This chapter describes how to define and initialize a system definition data set (CSD) that CICS needs to store definitions of the resources that it uses.

This chapter also discusses some considerations regarding the use of the CEDA transaction, particularly when a CSD is being shared by more than one CICS region.

A CSD is mandatory for some resource definitions. If you are creating a CSD for the first time, go through the steps listed below under “Summary of steps to create a CSD”: The remainder of this chapter describes these steps in more detail.

If you are already using a CSD with a previous version of CICS, upgrade your CSD to include CICS resource definitions new in CICS Transaction Server for VSE/ESA Release 1. For information about upgrading your CSD, see the *CICS Operations and Utilities Guide*.

You can run the DFHCSDUP offline utility as a batch job to read from and write to the CSD.

You should give update access to the CSD ***only*** to those users who are permitted to use the DFHCSDUP utility.

Summary of steps to create a CSD

If you do not already have a CSD in use at your installation:

1. Choose a data set name for your CSD. In the samples that follow, replace *CICS410.applid.DFHCSD* with a name for the CSD to suit your own naming conventions; but whatever you use must be the same name in:
   - The NAME parameter of the VSAM CLUSTER definition
   - The file identifier in the DFHCSD DLBL statement for the DFHCSDUP utility
   - The file identifier in the DFHCSD DLBL statement for CICS start up.
2. Decide how much disk space you require.
3. Define and initialize the CSD.
4. Decide what CICS file processing attributes you want for your CSD, and code these in as system initialization parameters.
5. Decide what backup and recovery procedures you require for your CSD. If you decide to make the CSD a CICS recoverable resource, code the CSDRECOV system intitialization parameter accordingly. You must also specify support for dynamic transaction backout, by coding the DBP system initialization parameter.
6. Decide if you want to use command logs for RDO; see “RDO command logs” on page 142 for details of the CADL, CAIL, CRDI, CSDL, CSFL, CSKL, CSPL, and CSRL destinations that CICS uses for RDO command logs.
7. Make the CSD available to CICS by including the necessary DLBL statement in the CICS startup job stream.

When you have started CICS, test the RDO transactions CEDA, CEDB, and CEDC. For information about these transactions, see the *CICS Resource Definition Guide*.

8. Finally, if you want to restrict access to particular CICS-supplied transactions by applying security, create a new group list (to use instead of the IBM-defined list, DFHLIST). This list should contain the resource definitions for the transactions CEMT, CEST, CECI, CEDA, CEDB (and any others that you think need to be secure) with the transaction security values appropriate to your installation. For information about how to do this, see the *CICS Resource Definition Guide*.

When you are sure that RDO is installed correctly, you can plan to migrate your CICS tables to the CSD. The PCT, the PPT, and the TCT for VSAM resources, are obsolete in CICS Transaction Server for VSE/ESA Release 1, and these must be migrated to your CSD.

For information about migrating CICS control tables to your CSD using the MIGRATE command, see the *CICS Operations and Utilities Guide*.

You define file control resource definitions for the CSD by specifying CSDxxxxx system initialization parameters, which are described in Chapter 22, “CICS system initialization” on page 187.

Calculating disk space

Before you can create the CSD, you must calculate the amount of space you need in your CSD for definition records. Use the following information:

- Each resource definition (for example each program, transaction and terminal) needs one record; the maximum sizes of these definition records are:

Resource	**Definition record size**
Program	169 bytes
Transaction	309 bytes
Profile	164 bytes
Partition set	135 bytes
Map set	135 bytes
Terminal	314 bytes
Typeterm	325 bytes
Connection	200 bytes
Partner	210 bytes
Tranclass	134 bytes
Session	240 bytes
File	277 bytes
LSR pool	224 bytes

- Each group requires two 122-byte records.
- Each group list requires two 122-byte records.
- Each group name within a list requires one 66-byte record.

In your calculation, allow for approximately 708 CICS-supplied resource definitions of various types, which are loaded into the CSD when you initialize the CSD with the utility program, DFHCSDUP. Finally, add a suitable contingency (approximately 25%), and use your calculated figure when you define the VSAM cluster for the CSD. (See the sample job in Figure 44 on page 131.)

Alternatively, you may decide to use the TRACKS or CYLINDERS space allocation parameters, instead of RECORDS as shown in Figure 44 on page 131. Whichever method you use, ensure that you allow for expansion by specifying a secondary extent value also.

Defining and initializing the CICS system definition

Before you can use the CSD, you must define it as a VSAM KSDS data set, and initialize it using the DFHCSDUP utility program as shown in Figure 44.

The INITIALIZE command initializes your CSD with definitions of the CICS-supplied resources. After initialization, you can migrate resource definitions from your CICS control tables, and begin defining your resources interactively with CEDA. You use INITIALIZE only once in the lifetime of the CSD.

The command LIST ALL OBJECTS lists the CICS-supplied resources that are now in the CSD.

```
* This job DEFINES and INITIALIZES a new CSD
* for CICS Transaction Server for VSE/ESA Release 1
// JOB DEFCSD
// DLBL CICSUCT,'usercat',,VSAM
// DLBL DFHCSD,'CICS410.applid.DFHCSD',,VSAM,CAT=CICSUCT      1
// EXEC IDCAMS,SIZE=AUTO
   DEFINE CLUSTER -
            (NAME(CICS410.applid.DFHCSD) -
            VOLUMES(volid) -
            KEYS(22 0) -                                  2
            INDEXED -
            RECORDS(n1 n2) -                              3
            RECORDSIZE(120 500) -                         4
            FREESPACE(10 10) -
            SHAREOPTIONS(2)) -                            5
          DATA -
            (NAME(CICS410.applid.DFHCSD.DATA)     -
             CONTROLINTERVALSIZE(8192)) -                 6
          INDEX -
            (NAME(CICS410.applid.DFHCSD.INDEX))     -
          CATALOG(usercat)
/*
* INITIALIZE THE CSD
// LIBDEF PHASE,SEARCH=(PRD1.BASE,PRD2.SCEEBASE)     7
// EXEC DFHCSDUP
   INITIALIZE
   UPGRADE USING(DFHCURCF)                            8
   UPGRADE USING(DFHCUICF)                            9
   LIST ALL OBJECTS
/*
/&
```

Figure 44. Sample job to define and initialize the CSD

Notes for Figure 44:

1 The file name in the DLBL statement for the CSD must be “DFHCSD”.

2 The KEYS (22 0) parameter in this sample job stream is mandatory and you must ***not*** vary it.

3 See "Calculating disk space" on page 130 for guidance about how much space to allow for your CSD, and allocate values for `n1` and `n2` appropriate to your installation.

4 The average record size of 120 bytes is calculated for a CSD that contains only the IBM-supplied resource definitions (generated by the INITIALIZE and UPGRADE commands). If you create a larger proportion of terminal resource definition entries than defined in the initial CSD, the average record size is higher because of the large size of terminal-type entries. (See the terminal and typeterm definition record sizes in "Calculating disk space" on page 130.)

5 Code the VSAM SHAREOPTIONS parameter as shown.

6 You can vary the CONTROLINTERVALSIZE (CI size) from the recommended value of 8192. Changing the CI size may affect the time taken to start CICS with a cold start, but should not affect a warm start, or CICS shutdown times.

If you run the job stream as shown, VSAM assumes a BUFFERSPACE value of 16 896 bytes, but you can code an explicit value if you want to use larger buffers. Coding a BUFFERSPACE parameter larger than the default may give some improvement on a cold start, but at the expense of an increase in the virtual storage required for the VSAM buffers.

7 The LIBDEF SEARCH chain must include the library containing the LE/VSE base code; usually called PRD2.SCEEBASE.

8 If you require the report controller resource definitions, you must specify the UPGRADE command shown to include the required report controller resource definitions. They are ***not*** generated by the INITIALIZE command, nor are they included in group DFHLIST.

9 If you require the ICCF resource definitions, include the UPGRADE command as shown. The definitions are ***not*** generated automatically by the INITIALIZE command.

Creating a larger CSD

To avoid the CSD filling while CICS is running, ensure that you define the data set with primary and secondary space parameters, and that there is sufficient DASD space available for secondary extents. If your CSD fills up while you are running a CEDA transaction (or the offline utility), define a larger data set and use an IDCAMS command, such as REPRO, to recover the contents of the CSD. If your CSD was dynamically allocated, you can close it, delete it, and redefine it as a larger data set. If your CSD was not dynamically allocated, you must shut down CICS to create a larger data set.

For a description of the commands that you can use for copying files, see the *VSE/VSAM Commands* manual.

File processing attributes for the CSD

File processing attributes for the CSD must be supplied as the following system initialization parameters:

CSDACC	The type of access allowed
CSDBUFND	The number of buffers for CSD data
CSDBUFNI	The number of buffers for the CSD index
CSDFRLOG	A forward recovery journal identifier
CSDJID	An identifier for automatic journaling
CSDLSRNO	A VSAM local shared resource pool
CSDRECOV	Whether or not the CSD is recoverable
CSDSTRNO	The number of strings for concurrent requests.

These parameters are described in greater detail in Chapter 22, “CICS system initialization” on page 187.

Sharing and availability of the CSD

This section describes considerations that affect how you can implement the sharing of a CSD. Sharing the CSD by several CICS regions enables those regions to use the same definitions, and means there is no need for duplicate data sets.

To optimize the sharing of a CSD, you should observe the following:

Shared user access from the same CICS region

- Several users in a CICS region can access the CSD at the same time
- If you have specified read/write access for the CSD, all the CEDA and CEDB users in a CICS region can perform read and write functions. CICS file control manages concurrent access for multiple users within a region, using the attributes specified in the CSDACC system initialization parameter.

For more information, see “Multiple users of the CSD within a CICS region” on page 135.

Shared user access from several CICS regions

- Several users in different CICS regions can access the CSD at the same time
- Only one CICS region should be given read/write access to the CSD (CSDACC=READWRITE system initialization parameter). That CICS region should be at the latest level, to ensure that obsolete resource attributes from earlier release can still be updated safely. Other CICS regions should be given only read access to the CSD (CSDACC=READONLY system initialization parameter). This ensures that the CSD integrity is preserved for CICS regions in the same VSE image or different VSE images.
- If you update your shared CSD from one region only, and use CEDA in all the other regions just to install into the required region, specify read/write access for the updating region and read-only for all other regions.

- If you want to update the CSD from several CICS regions, you can use the CICS transaction routing facility, and MRO or ISC, to enable read-only CICS regions to update the CSD. The procedure to follow is:
  1. Select one region that is to own the CSD (the CSD-owning region), and only in this region specify read/write access for the CSD.
  2. Define the CSD as read-only to other CICS regions.
  3. For all regions other than the CSD-owning region:
     a. Redefine the CEDB transaction as a remote transaction (to be run in the CSD-owning region).
     b. Install the definition and add the group to your group list for these regions.

     You may then use the CEDB transaction from any region to change the contents of the CSD, and use CEDA to INSTALL into the invoking region. You cannot use CEDA to change the CSD in region(s) that do not own the CSD.

     If the CSD-owning region fails, the CSD is not available through the CEDB transaction until emergency restart of the CSD-owning region has completed (when any backout processing on the CSD is done). If you try to install a CSD GROUP or LIST that is the target of backout processing, before emergency restart, you are warned that the GROUP or LIST is internally locked to another user. Do not run an offline VERIFY in this situation, because backout processing removes the internal lock when emergency restart is invoked in the CSD-owning region.

  If you do not want to use the above method, but still want the CSD to be defined as a recoverable resource, then integrity of the CSD cannot be guaranteed.
- You can define several CICS regions with read/write access to the CSD, but this should only be considered if all the CICS regions run in the same VSE image, and all are at the latest CICS level.
- If you give several CICS regions read/write access to the same CSD, and those regions are in the same VSE image, integrity of the CSD is maintained by the SHAREOPTIONS(2) operand of the VSAM definition, as shown in the sample job stream on page 131.
- If you give several CICS regions read/write access to the same CSD, and those regions are in different VSE images, the VSAM SHAREOPTIONS(2) operand does not provide CSD integrity, because the VSAMs for those VSE images do not know about each other.

For more information about shared CSD access within one VSE image, see “Sharing a CSD by CICS regions within a single VSE image” on page 135. For more information about shared CSD access across several VSE images, see “Sharing a CSD in a multi-VSE environment” on page 137. For information about sharing the CSD between CICS Transaction Server for VSE/ESA Release 1 and earlier versions, see page 138.

Shared access from CICS regions and DFHCSDUP

If you want to use the DFHCSDUP utility program in read/write mode to update the CSD, you must ensure that no CICS users are using any of the CEDA, CEDB, or CEDC transactions.

For information about other factors that can restrict access to a CSD, see "Other factors restricting CSD access" on page 139.

For information about the system initialization parameters for controlling access to the CSD, see page 225.

Multiple users of the CSD within a CICS region

If you have specified read/write access for the CSD, all the CEDA users in a CICS region can perform read and write functions. CICS file control manages concurrent access for multiple users within a region, using the attributes specified in the CSDACC system initialization parameter.

CICS protects individual resource definitions against concurrent updates by a series of internal locks on the CSD. CICS applies these locks at the group level. While CICS is executing a command that updates any element in a group, it uses the internal lock to prevent other RDO transactions within the region from updating the same group. CICS removes the lock record when the updating command completes execution. Operations on lists are also protected in this way.

The number of concurrent requests that may be processed against the CSD is defined by the CSDSTRNO system initialization parameter. Each user of CEDA (or CEDB or CEDC) requires two strings, so calculate the CSDSTRNO value by first estimating the number of users that may require concurrent access to the CSD, and then multiply the number by two.

The CEDx transactions issue a diagnostic message if the CSDSTRNO value is too small to satisfy the instantaneous demand on the CSD for concurrent requests. A subsequent attempt to reissue the command succeeds if the conflict has disappeared. If conflicts continue to occur, increase the CSDSTRNO value.

Sharing a CSD by CICS regions within a single VSE image

The CSD may be shared by a number of CICS regions within the same VSE image. You can maintain the integrity of the CSD in this situation by coding SHAREOPTIONS(2) on the VSAM definition, as shown in the sample job stream on page 131. The CICS attributes of the CSD, as viewed by a given region, are defined in the system initialization parameters for that region.

You should consider defining:

- One CICS region with read/write access (CSDACC=READWRITE) to the CSD. That region can use all the functions of CEDA, CEDB, and CEDC.
- Other CICS regions with only read access (CSDACC=READONLY) to the CSD. Such CICS regions can use the CEDC transaction, and those functions of CEDA and CEDB that do not require write access to the CSD (for example, they can use INSTALL, EXPAND, and VIEW, but not DEFINE). You can enable such CICS regions to update the CSD, by using the procedure described on page 134.

Note: Read integrity is not guaranteed in a CICS region that has read-only access to a shared CSD. For example, if one CICS region that has full read/write access updates a shared CSD with new or changed definitions, another CICS region with read-only access might not obtain the updated information. This could happen if a control interval (CI) already held by a read-only region (before an update by a read/write region) is the same CI needed by the read-only region to obtain the updated definitions. In this situation, VSAM does not reread the data set, because it already holds the CI. However, you can minimize this VSAM restriction by specifying CSDLSRNO=NONE, and the minimum values for CSDBUFNI and CSDBUFND, but at the expense of degraded performance.

If you define several CICS regions with read/write access to the CSD, those regions should all be at the latest level. Only one CICS region with read/write access can use a CEDA, or CEDB transaction to access the CSD, because the VSAM SHAREOPTIONS(2) definition prevents other regions from opening the CSD.

If you are running CICS with the CSD defined as a recoverable resource (CSDRECOV=ALL), see “Planning for backup and recovery” on page 140 for some special considerations.

You can use CEMT to change the file access attributes of the CSD, or you can use the EXEC CICS SET FILE command in an application program, However, ensure that the resulting attributes are at least equivalent to those defined either by CSDACC=READWRITE or CSDACC=READONLY. These system initialization parameters allow the following operations on the CSD:

CSDACC operand	***Operations***
READONLY	Read and browse.
READWRITE	Add, delete, update, read and browse.

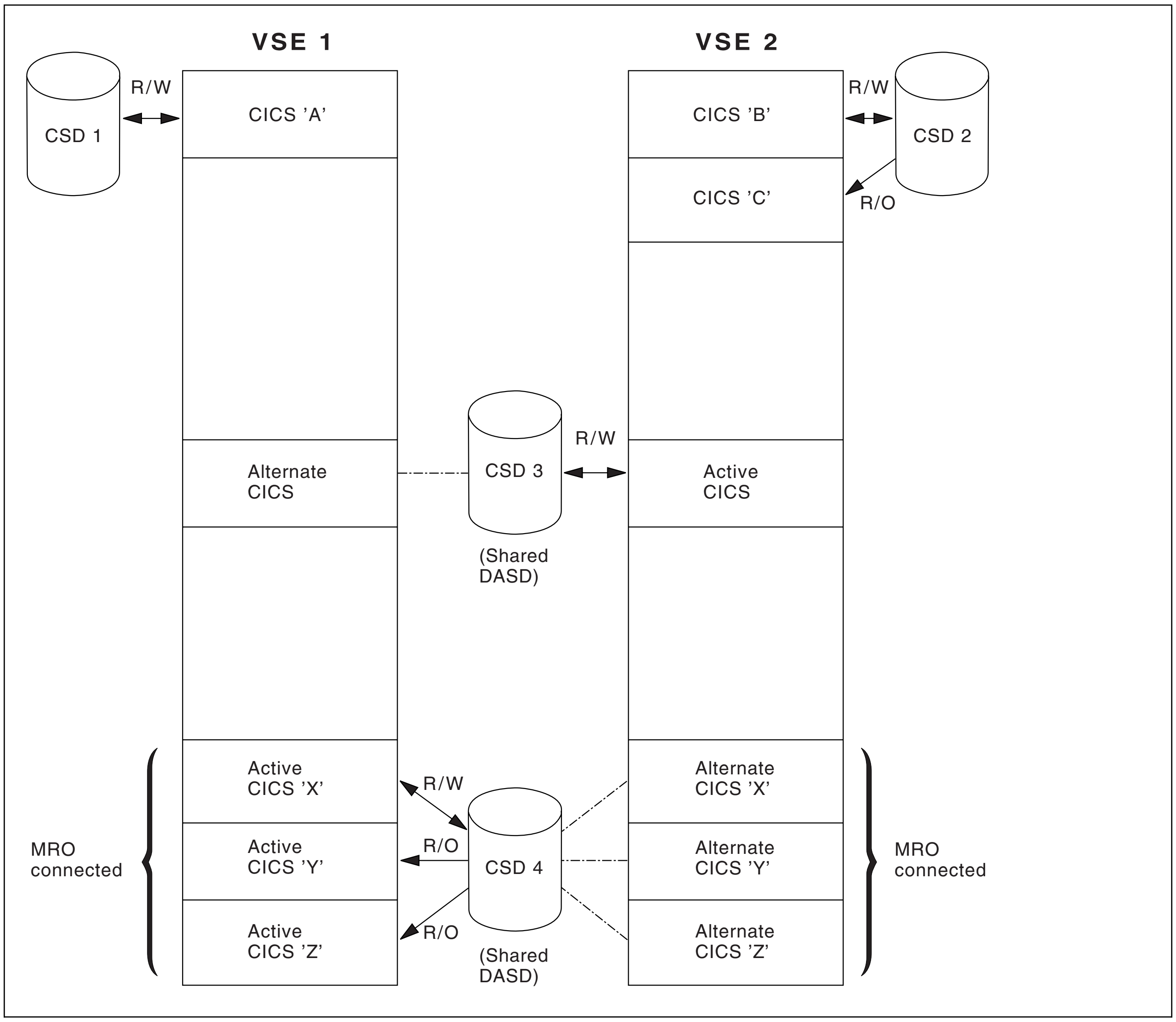


Figure 45. Some examples of CSD usage in a multi-VSE environment

Sharing a CSD in a multi-VSE environment

If you need to share a CSD between CICS regions that are running in different VSE images, you should ensure that only one region has read/write access.

The VSAM SHAREOPTIONS(2) provides full integrity only if the CSD is on a volume that was defined as shared at IPL time.

These multi-VSE restrictions also apply to running the offline utility, DFHCSDUP. See Figure 45 for examples of CSD usage in a multi-CPC environment.

Multiple users of one CSD across CICS or batch partitions

The types of access needed in the four situations where the CSD may be used are shown in Table 20 on page 138.

Table 20. CSD access

	Type of Activity	Access
1	CICS region performing initialization (cold start)	Read-only
2	CICS region running one or more CEDA, CEDB, or CEDC transactions	Read/write or read-only (as specified in the CSDACC system initialization parameter)
3	Batch region running utility program DFHCSDUP	Read/write or read only, depending on PARM parameter
4	CICS regions performing emergency restart, and file backout is required	Read/write

Note the following limitations when the activities listed in Table 20 are attempted concurrently:

1. You cannot run DFHCSDUP in read/write mode in a batch partition if any CICS partition using the same CSD is running one of the CEDA, CEDB, or CEDC transactions. (The exception is when the CEDx transactions accessing the CSD are in a partition (or partitions) for which the CSD is defined as read-only.)
2. None of the CEDx transactions runs if the CSD to be used is being accessed by the DFHCSDUP utility program in read/write mode. (This restriction does not apply if the transaction is run in a region for which the CSD is defined as read-only.)
3. None of the CEDx transactions runs in a CICS region whose CSD is defined for read-write access if any of the CEDx transactions are running in another CICS partition that has the CSD defined for read-write access.

A CICS partition starting with a cold start opens the CSD for read access only, regardless of the CSDACC operand. This enables a CICS partition to be initialized even if a user on another partition or the DFHCSDUP utility program is updating the CSD at the same time.

On a warm or emergency start, the CSD is not opened at all during CICS initialization if CSDRECOV=NONE is coded as a system initialization parameter. However, if CSDRECOV=ALL is coded, and backout processing is pending on the CSD, the CSD is opened during CICS initialization on an emergency start.

Sharing the CSD between CICS Transaction Server for VSE/ESA Release 1 and earlier releases

Resource attributes become obsolete when they have no relevance for a new release of CICS. In CICS Transaction Server for VSE/ESA Release 1, CICS continues to display these on CEDx panels, but they are displayed as protected fields, indicating that they are not supported by CICS Transaction Server for VSE/ESA Release 1. Using the ALTER command on definitions that specify obsolete attributes does not cause the loss of these attributes in CICS Transaction Server for VSE/ESA Release 1, so you can safely update resource definitions from a CICS Transaction Server for VSE/ESA Release 1 region. If you are sharing the CSD between a CICS Transaction Server for VSE/ESA Release 1 region and a

CICS/VSE Version 2 system, you can update the unsupported fields by using the PF2 function key to remove the protection when in ALTER mode. (PF2 is designated as the “compatibility” key (COM) on the CEDA or CEDB display panels.) Pressing PF2 converts protected fields to unprotected fields that can you can modify. If you want to use this facility to enable you to share common resource definitions, the rule for sharing between different release levels of CICS is that you ***must*** update the CSD from the higher level region, that is, CICS Transaction
# Server for VSE/ESA Release 1. Do not add resource definitions created at the
# higher level region to a CSD which has not been created at or upgraded to the
# higher level.

For information about using the CEDA and CEDB ALTER commands to update resource definitions in compatibility mode, see the *CICS Resource Definition Guide*.

You can also use the CICS Transaction Server for VSE/ESA Release 1 CSD utility program, DFHCSDUP, to update resources that specify obsolete attributes. A compatibility option is added for this purpose, which you must specify on the PARM parameter on the EXEC DFHCSDUP statement. You indicate the compatibility option by specifying COMPAT or NOCOMPAT. The default is NOCOMPAT, which means that you cannot update obsolete attributes.

For information about sharing the CSD between CICS releases, see the *CICS Migration Guide*.

CICS supplied compatibility groups

If you are sharing the CSD between CICS Transaction Server for VSE/ESA Release 1 and CICS for VSE/ESA Version 2.3, you must ensure that the group list you specify (on the GRPLIST system initialization parameter) contains all the CICS-required standard definitions. When you upgrade the CSD to the CICS Transaction Server for VSE/ESA Release 1 level, some of the IBM groups referenced by your group list are deleted, and the contents transferred to the compatibility group, DFHCOMP1. To ensure that these continue to be available to CICS for VSE/ESA Version 2.3 regions, add the compatibility group *after* all the other CICS-supplied definitions.

A separate compatibility group is created for the report controller (DFHCOMP2).

For information about upgrading your CSD, and about the compatibility groups in CICS Transaction Server for VSE/ESA Release 1, see the *CICS Resource Definition Guide*.

Other factors restricting CSD access

Access to the CSD may also be restricted if it is left open after abnormal termination of a CEDA, CEDB, or CEDC transaction. If the CSD is left open with write access, this prevents other address spaces from subsequently opening it for write access. This situation can be remedied by using CEMT to correct the status of the CSD.

Access to the CSD is not released until the RDO transaction using it is ended, so users of CEDA, CEDB, and CEDC should ensure that a terminal running any of these transactions is not left unattended. Always end the transaction with PF3 as soon as possible. Otherwise, users in other regions are unable to open the CSD.

There may be times when you cannot create definitions in a group or list. This situation arises if an internal lock record exists for the group or list you are trying to update. If you are running the DFHCSDUP utility program (or a CEDA transaction) when this occurs, CICS issues a message indicating that the group or list is locked. As described under “Multiple users of the CSD within a CICS region” on page 135, this is normally a transient situation while another user within the same region is updating the same group or list. However, if a failure occurs, preventing a CEDA transaction from completing successfully, and CSDRECOV=NONE is coded, the internal lock is not removed and is left in force. (If CSDRECOV=ALL is coded, the CSD is recoverable and file backout occurs and frees the lock.) This could happen, for example, if a system failure occurs while a CEDA transaction is running; it could also happen if the CSD becomes full. You can remedy this situation by running the DFHCSDUP utility program with the VERIFY command.

However, if you have coded CSDRECOV=ALL, make sure no backout processing is pending on the CSD before you run an offline VERIFY. The effect of coding CSDRECOV=ALL is discussed more fully under “Planning for backup and recovery.”

Planning for backup and recovery

To guard against system failures that affect your CSD, take a backup copy of the CSD at regular intervals. Then, if the CSD is corrupted for any reason, you can restore it to its state at the last backup.

If you code CSDRECOV=ALL as a system initialization parameter, all changes (after images) made by CICS to the CSD are logged in the CICS journal defined by the CSDFRLOG system initialization parameter. If you code CSDRECOV=ALL, but omit CSDFRLOG, CICS uses the system log as the journal for CSD recovery. Using the latest backup copy and the after images from the relevant CICS journal, you can recover all the changes made by running a forward recovery utility. After performing forward recovery, you must reenter the CEDA transactions that were running at the time of failure, as these are effectively backed out by the forward recovery process. You can find details of these in the CSDL transient data destination, which is the log for copies of all CEDA commands. See “RDO command logs” on page 142 for more information.

The CSDRECOV and CSDFRLOG system initialization parameters interact according to how they are specified. Table 21 and Table 22 on page 141 summarize their effects when the SIT is assembled and during CICS override processing, respectively.

Table 21. Interactions between CSD system initialization parameters at SIT assembly time

CSDRECOV	CSDFRLOG	Result
ALL	FRLOG from 01 through 99.	OK
ALL	NO	CSDFRLOG defaults to 01 — the system log.
BACKOUTONLY or NONE	NO	OK
BACKOUTONLY or NONE	FRLOG from 01 through 99.	SIT assembly warning MNOTE stating that CSDFRLOG requires CSDRECOV=ALL.

Table 22. Interactions between CSD system initialization parameters

CSDRECOV	CSDFRLOG	Result
ALL	FRLOG from 01 through 99.	OK
ALL	NO	CSDFRLOG defaults to 01 — the system log.
BACKOUTONLY or NONE	NO	OK
BACKOUTONLY or NONE	FRLOG from 01 through 99.	Processing continues and message DFHPA1930 stating that CSDFRLOG has been ignored is issued.

Write and test procedures for backing up and recovering your CSD before beginning to operate a production CICS region.

Forward recovery of the CSD is not possible if CSD updates are made outside CICS (for example, by using DFHCSDUP) To enable recovery of the updates made outside CICS, you need to use a backup copy. If you update the CSD from outside CICS, do not use CEDA to update the CSD until a backup has been taken, for example, by using an IDCAMS REPRO command.

Transaction backout during emergency restart

If you define the CSD as a recoverable resource, by coding the CSDRECOV system initialization parameter, the same rules apply to the CSD as to any other CICS recoverable resource. If you code CSDRECOV=ALL (or BACKOUTONLY) as a system initialization parameter, and have to perform an emergency restart following a failure, CICS backs out any incomplete RDO transactions that were in-flight at the time of failure.

Dynamic backout for transactions

CICS performs dynamic transaction backout for any RDO transaction abends. You cannot decide whether you want dynamic transaction backout by coding an attribute on transaction definitions in the CSD; CICS assumes this requirement for all transactions. This means that you must include support for dynamic transaction backout by coding the DBP system initialization parameter. To do this, code:

DBP=1$, if you have ***not*** installed DL/I support
DBP=2$, if you have installed DL/I support.

Other recovery considerations

When you are deciding what to code on the CSDRECOV parameter, consider the following factors:

- CEDA command syncpoint criteria
- Sharing the CSD with another CICS region
- Accessing the CSD by the offline utility program DFHCSDUP.

For information about CEDA command syncpoint criteria, see “CEDA command syncpoint criteria” on page 142: For information about sharing the CSD between CICS regions, see “Sharing and availability of the CSD” on page 133: For information about using the DFHCSDUP utility to access the CSD, see “Accessing the CSD by the offline utility program, DFHCSDUP” on page 142.

CEDA command syncpoint criteria

You can issue CEDA in two ways:

1. A single command entered on the command line
2. A series of single commands from within an EXPAND or DISPLAY panel.

Commands that change the contents of the CSD commit or back out changes at the single command level. The exception to this rule is a generic ALTER command. A generic ALTER command is committed or backed out at the single resource level.

The replacement of an existing resource definition by an INSTALL command only occurs if the resource is not in use. If any of the resources in the group being installed are in use, the install will fail.

Changes made to the following resource definitions by an INSTALL command are committed at the resource level and are not backed out if the install fails:

AUTOINSTALL MODEL, FILE, LSRPOOL, MAPSET, PARTITIONSET, PARTNER, PROFILE, PROGRAM, TRANSACTION and TRANCLASS.

Changes made to the following resource definitions by an INSTALL command are committed at the group level and are backed out if the install fails:

CONNECTION, SESSION, TERMINAL, and TYPETERM

Accessing the CSD by the offline utility program, DFHCSDUP

Changes made to the CSD by the offline utility program DFHCSDUP are not recoverable. Also consider the effects of using commands provided by this program before emergency restart of a failing CICS region, that:

1. Change the contents of lists or groups that are the target of backout processing.
2. Remove internal locks (by using VERIFY, for example).

These situations are analogous to the problems met when using multiple read/write regions, and are discussed above.

RDO command logs

If you want to record RDO commands, specify the DCT entries for the CADL, CAIL, CRDI, CSDL, CSFL, CSKL, CSPL, and CSRL extrapartition transient data destinations that CICS can use as the logs. These are used as follows:

CADL	Logs VTAM resources installed in the active CICS region. CICS records in this log all terminal entries installed in the TCT, entries deleted from the TCT, and dynamically installed entries that are discarded. This log includes autoinstalled terminal definitions, terminal definitions installed explicitly by the CEDA INSTALL command, and terminal definitions installed from a group list during system initialization.
CAIL	Logs autoinstall terminal model entries installed in the TCT, and entries deleted from the TCT.
CRDI	Logs installed resource definitions of programs, transactions, mapsets, profiles, partition sets, files, and LSR pools.

CSDL Logs RDO commands that affect the CSD.

CSFL Logs file resources installed in the active CICS region. That is, all file entries installed in the FCT, entries deleted from the FCT, dynamically installed entries that are discarded, and messages from dynamic allocation of data sets and from loading CICS data tables.

CSKL Logs transaction and profile resources installed in the active CICS region. That is, all transaction and profile entries installed in the PCT, entries deleted from the PCT, and dynamically installed entries that are discarded.

CSPL Logs program resources installed in the active CICS region. That is, all program entries installed in the PPT, entries deleted from the PPT, and dynamically installed entries that are discarded.

CSRL Logs changes to the set of partner resources installed in the active CICS region. That is, all operations that install or discard partner resources.

If you want these RDO command logs sent to the same destination (CSSL) as the messages, code DCT entries shown in Figure 46. (These definitions are included in the sample DCT copy member, DFH$DCTR, which is supplied in the VSE/ESA sublibrary, PRD1.BASE.) If you like, you can direct these logs to any other transient data queue, or define them as extrapartition data sets.

```
CADL   DFHDCT TYPE=INDIRECT,     CEDA VTAM RESOURCE LOGGING          X
             DESTID=CADL,                                            X
             INDDEST=CSSL
*
CAIL   DFHDCT TYPE=INDIRECT,     AUTOINSTALL TERMINAL MODEL LOGGING  X
             DESTID=CAIL,                                            X
             INDDEST=CSSL
*
CRDI   DFHDCT TYPE=INDIRECT,     RDO INSTALL LOGGING                 X
             DESTID=CRDI,                                            X
             INDDEST=CSSL
*
CSDL   DFHDCT TYPE=INDIRECT,     CEDA COMMAND LOGGING                X
             DESTID=CSDL,                                            X
             INDDEST=CSSL
*
CSFL   DFHDCT TYPE=INDIRECT,     FILE ALLOCATION MESSAGES            X
             DESTID=CSFL,                                            X
             INDDEST=CSSL
```

Figure 46 (Part 1 of 2). DCT entries for RDO command logs sent to CSSL

```
*
CSKL   DFHDCT TYPE=INDIRECT,     TRANSACTION+PROFILE RESOURCE LOGGING  X
             DESTID=CSKL,                                              X
             INDDEST=CSSL
*
CSPL   DFHDCT TYPE=INDIRECT,     PROGRAM RESOURCE LOGGING              X
             DESTID=CSPL,                                              X
             INDDEST=CSSL
*
CSRL   DFHDCT TYPE=INDIRECT,     PARTNER RESOURCE LOGGING              X
             DESTID=CSRL,                                              X
             INDDEST=CSSL
```

Figure 46 (Part 2 of 2). DCT entries for RDO command logs sent to CSSL

You must specify the block size, record size, and record format for these destinations in the DFHDCT TYPE=SDSCI macro. For example:

```
MSGUSR   DFHDCT TYPE=SDSCI,       CICS MESSAGES AND RDO LOGS           X
               BLKSIZE=136,                                            X
               BUFNO=1,                                                X
               DSCNAME=MSGUSR,                                         X
               RECFORM=VARUNB,                                         X
               RECSIZE=132,                                            X
               DEVADDR=SYSLST,                                         X
               DEVICER=1403,                                           X
               TYPEFLE=OUTPUT
```

Figure 47. Example DFHDCT TYPE=SDSCI statement for the RDO command logs

Making the CSD available to CICS

For a CICS execution, you need a DLBL statement, such as:

```
// DLBL DFHCSD,'CICS410.applid.DFHCSD',,VSAM,CAT=CICSUCT.
```

Installing the RDO transactions

The RDO transactions, CEDA, CEDB, and CEDC are defined in the CICS-supplied group, DFHSPI. This group is also included in DFHLIST, the CICS group list. Ensure that a copy of DFHSPI is included in a group list that you use for your CICS startup. You specify the group lists on the GRPLIST system initialization parameter.

For information about the CEDA, CEDB, and CEDC transactions, see the *CICS Resource Definition Guide*.

Moving your CICS tables to the CSD

When you have created a CSD, and initialized it with CICS-supplied definitions, you are ready to move the contents of your CICS tables to the CSD data set. You do this by running the offline utility, DFHCSDUP, using the MIGRATE command to specify the table you want to migrate. For information about the DFHCSDUP utility program and the available commands, see the *CICS Operations and Utilities Guide*. For information about moving the contents of CICS tables to the CSD, see the *CICS Resource Definition Guide*.

Release compatibility of the CSD

If you are using a CSD with an earlier release of CICS, upgrade your CSD as part of the process of migration. For information about upgrading the CSD, and about the release compatibility of the CSD after upgrading, see the *CICS Migration Guide*.

Installing definitions for national language support (NLS)

If you intend to use CICS national language support (NLS), install the definitions for the appropriate languages by running the DFHCSDUP utility and specifying the following UPGRADE commands:

For Kanji	UPGRADE USING(DFHRDJPN)
For German	UPGRADE USING(DFHRDGER)
For Chinese	UPGRADE USING(DFHRDCHI)

For information about the DFHCSDUP utility program and the available commands, see the *CICS Operations and Utilities Guide*.

XRF considerations

If you are running CICS with XRF, both the active and alternate CICS systems must refer to the same CICS system definition data set (that is, the CSD must be passively shared). The active and alternate CICS systems can share the same CSD even if they are running on different VSE images. A CICS system running with XRF=YES may also share the CSD with other CICS systems within the same VSE image. (For a definition of passively and actively shared data sets in an XRF environment, see page 98.)

The alternate CICS system does not open the CSD during initialization, or before takeover occurs. The alternate CICS system does not even open the CSD during takeover, if the CSD was not changed at any time by the active CICS system. (For example, the CSD might have been used only to install a group list at CICS startup, and subsequently by read-only operations.) However, if you use the CEDA transaction in an active CICS system to alter resource definitions, the CSD might be opened at takeover, to perform any file backout that is necessary. To enable file backout to occur, you must define the CSD as a recoverable resource using the system initialization parameter CSDRECOV; see page 227 for more information.

For more information about using the CSD as a recoverable file, see “Planning for backup and recovery” on page 140.

VSE/ESA-supplied resource definitions

VSE/ESA also supplies CICS definitions for, for example, autoinstall terminals. Take care when you are migrating your CSD that none of these definitions, on which the operating system relies, are lost. You have two options:

1. Add all the CICS-supplied and VSE/ESA-supplied definitions to your own migrated CSD by using the CSD created during the base installation of VSE/ESA and CICS Transaction Server for VSE/ESA Release 1 as the foundation for your CICS Transaction Server for VSE/ESA Release 1 CSD.

2. Add your own definitions to the VSE/ESA-supplied CSD. The DFHCSDUP EXTRACT command is particularly useful for this purpose. The DFH0CBDC EXTRACT sample program shows you how to retrieve your own definitions from your own CSD, and to create a batch job to insert them into a new CSD. See the *CICS Customization Guide* for further details on the DFH0CBDC sample program.

3. Migrating all unsupported table definitions to the CSD using the DFHCSDUP MIGRATE command.

4. Use the DFHCSDUP APPEND and ADD commands to extend the VSE/ESA supplied LIST to include your own definitions. For example:

   ```
   APPEND LIST  (VSELIST) TO (MYLIST)
   ADD    GROUP (PAYROLL) TO (MYLIST)
   ADD    GROUP (SALES)   TO (MYLIST)
   etc.
   ```

Chapter 15. Defining and initializing the restart data set

This chapter describes the restart data set and shows you how to define and initialize it.

The restart data set is a VSAM key-sequenced data set (KSDS). It is used during emergency restarts only, to temporarily hold the backout information read from the CICS system log. However, the restart data set must be available whenever CICS starts up; that is, you should always include a data definition statement for the restart data set in your CICS startup job stream.

Job control statements to define and initialize the restart data set

Before its first use, you must define and initialize the restart data set as a VSAM key sequenced data set (KSDS). To do this, you can use the sample job in Figure 48 on page 148.

The name in the CLUSTER definition must be the same as the file identifier in the DLBL statement for the restart data set in the CICS startup job stream.

The values of the RECORDS and CATALOG parameters may vary according to your installation's requirements. The value of the NAME parameter in the CLUSTER definition is the data set name, which must agree with the file identifier in the DLBL statement for the CICS execution.

You must initialize the RSD with 1 record containing a 22-byte key:

```
'ACTL 0001'
```

which must be followed by 13 blanks.

```
// JOB DEFRSD CREATE RESTART DATA SET
// EXEC IDCAMS,SIZE=AUTO
 DEFINE CLUSTER                     -
         (NAME(CICS410.applid.DFHRSD  -
          INDEXED                   -  1
          KEYS(22 0)                -  1
          FREESPACE(20 20)          -
          SHAREOPTIONS(2)           -
          VOLUMES(volid))           -
       DATA                         -
         (NAME(CICS410.applid.DFHRSD.DATA)      -
          RECORDSIZE(400 1000)      -
          RECORDS(500 100)          -  2
          CONTROLINTERVALSIZE(2048))  -  3
       INDEX                        -
         (NAME(CICS410.applid.DFHRSD.INDEX))    -
       CATALOG(usercat)
/*
/&
// JOB INITRSD INITIALIZE RESTART DATASET
// DLBL CICSUCT,'usercat',,VSAM
// DLBL DFHRSD,'CICS410.applid.DFHRSD',,VSAM,CAT=CICSUCT
// EXEC IDCAMS,SIZE=AUTO
 REPRO  INFILE                      -
         (SYSIPT                    -
          ENVIRONMENT               -
           (RECORDFORMAT(FIXUNB)    -
            BLOCKSIZE(80)           -
            RECORDSIZE(80)))        -
        OUTFILE(DFHRSD)
ACTL 0001
/*
/&
```

Figure 48. Sample job to define and initialize an RSD

Notes for Figure 48:

1 These parameters are mandatory.

2 Whichever IDCAMS parameter you use for the RSD space allocation (CYLINDERS, TRACKS, or RECORDS for CKD device, or BLOCKS for an FBA device), make sure you specify a secondary extent. CICS abends if your RSD fills and VSAM cannot create a secondary extent.

3 You can vary the CONTROLINTERVALSIZE from the recommended value of 2048 bytes. However, although a larger value reduces the number of CI-splits and CA-splits, other factors increase CICS shutdown times if you specify a CI size larger than 2048 bytes.

The default BUFFERSPACE value assumed by VSAM for this IDCAMS run is 4608 bytes, but you can code an explicit value to use larger buffers. A larger buffer space may improve emergency restart times, but at the expense of an increase in the virtual storage required for the VSAM buffers.

CICS locates the RSD buffers above the 16MB line, if storage is available.

Job control statements for CICS execution

If you defined the restart data set using the sample job shown in Figure 48 on page 148, the data definition statement for the CICS execution is:

```
// DLBL  DFHRSD,'CICS410.applid.DFHRSD',,VSAM,CAT=CICSUCT.
```

Note: The restart data set is a passively shared data set when running CICS with XRF.

Chapter 16. Defining and using catalog data sets

This chapter describes how to define and use the CICS **global** catalog data set (GCD), and the CICS **local** catalog data set (LCD). CICS needs both these data sets to catalog CICS system information. For the rest of this chapter, these data sets are referred to as the global catalog and the local catalog. (The CICS catalog data sets are not connected with any VSE catalogs, and contain data that is unique to CICS.).

Attention! Defining and initializing the CICS catalogs

You must define and initialize new CICS catalogs for CICS Transaction Server for VSE/ESA Release 1.

If, for any reason, you need to reinitialize one of the catalogs while running CICS, reinitialize them both. If you reinitialize only one of the catalogs, CICS fails on any subsequent restart.

For more information about how CICS uses the catalogs for startup and restart, see “Classes of start and restart” on page 209.

The global catalog

The global catalog is a VSAM key-sequenced data set (KSDS). In an XRF environment there is only one global catalog. It is shared passively between the active and the alternate CICS systems. The global catalog is used:

- During a controlled shutdown to record warm keypoint information for a subsequent warm start.
- During the running of CICS to hold the resource definitions that are ***installed***, either during initialization when CICS installs the group list, or by means of a CEDA INSTALL or EXEC CICS CREATE command. These definitions can be for:
  - Programs
  - Transactions and transaction profiles
  - Transaction classes
  - BMS map sets and partition sets
  - Terminals, including any that are autoinstalled
  - Sessions and connections, for communication with other CICS regions
  - Files
- To hold disk journal status information.

For further guidance information about what is written to the global catalog, and about how CICS uses the global catalog for startup and restart, see “Classes of start and restart” on page 209.

Job control statements to define and initialize the global catalog

Before its first use, you must define and initialize the CICS global catalog as a KSDS. You can use the sample job in Figure 49 to do this.

In Figure 49, the job step INITGCD initializes the data set with one record containing a blank 28-byte key. You can also use the job step to reinitialize the global catalog without redefining it. You can use any value for the key, but generally it is best to use a key comprising all blanks (X'40').

You must initialize the RSD with 1 record containing a 22-byte key:

```
'ACTL 0002'
```

which must be followed by 13 blanks.

```
// JOB DEFGCD CREATE LOCAL CATALOG DATA SET
// DLBL CICSUCT,'usercat',,VSAM
// DLBL DFHGCD,'CICS410.applid.DFHGCD',,VSAM,CAT=CICSUCT
// EXEC IDCAMS,SIZE=AUTO
   DEFINE CLUSTER                      -
            (NAME(CICS410.applid.DFHGCD)      - 1
            VOLUME(volid)              -
            KEYS(28 0)                 -
            INDEXED                    -
            CYL(n1 n2)                 - 2
            REUSE                      - 3
            RECSZ(8185 8185)           - 4
            BUFSP(303104)              - 5
            FSPC(10 10)                -
            SHR(2) )                   -
          DATA                         -
            (NAME(CICS410.applid.DFHGCD.DATA)    -
            CISZ(8192))                - 6
          INDEX                        -
            (NAME(CICS410.applid.DFHGCD.INDEX))  -
          CATALOG(usercat)
/*
// IF $RC GT 4 THEN
// GOTO $EOJ
* Initialise the GCD
// EXEC IDCAMS,SIZE=AUTO
   REPRO INFILE       -
           (SYSIPT     -
           ENVIRONMENT -
             (RECORDFORMAT(FIXUNB) -
              BLOCKSIZE(80)    -
              RECORDSIZE(80))) -
         OUTFILE(DFHGCD)  7
ACTL 0002
/*
```

Figure 49. Sample job to define and initialize the global catalog

Notes for Figure 49:

1 The data set name in the CLUSTER definition must be the same as the file id in the DLBL statement for the global catalog in the CICS startup job stream.

2 The primary and secondary extent sizes are shown as n1 and n2 cylinders. Calculate the size required to meet your installation's needs, and substitute your values for n1 and n2.

Whichever IDCAMS parameter you use for the GCD space allocation (CYLINDERS, TRACKS, or RECORDS), make sure that you specify a secondary extent. CICS abends if your GCD fills and VSAM cannot create a secondary extent.

3 To enable the global catalog to be opened again and again as a reusable cluster, you must specify the REUSE option on the DEFINE CLUSTER command, and the REPRO command used to re-initialize the global catalog.

4 If your maximum record size is greater than 8185, you must change the RECORDSIZE (RECSZ) parameter in the sample job to specify your own value, and increase the CISZ value. For information about record sizes, see "Space calculations" on page 154.

5 Based on the control interval sizes and the CICS STRNO, 303 104 bytes is the **minimum** buffer space that VSAM will use on OPEN. See the calculation in bullet **6** for more information.

6 You can vary the CONTROLINTERVALSIZE (CISZ) from the values shown in the VSAM definition. However, although larger values reduce the number of control interval (CI) and control area (CA) splits, other factors increase CICS shutdown times, and slow down a cold start.

For performance reasons, CICS defines a STRNO (number of strings) value of 32. Based on the example job stream in Figure 49 on page 152, the absolute minimum value of BUFSP is calculated as follows:

```
BUFND = (STRNO + 1) = 33
BUFNI = STRNO = 32
BUFSP = 33 * 8192 (BUFND * CI size) + 32 * 1024
        (BUFNI * CI size) = 303104 bytes
```

Note: 303 104 the smallest figure that can be used for BUFSP. For improved performance, use BUFNI=STRNO plus the number of index levels, minus 1.

Another way to define buffer space for the GCD is by means of the BUFSP parameter on the DLBL statement for the GCD in the CICS startup job stream, which you can use to override the default or defined value. Note, however, that VSAM uses the maximum buffer space. If you define a BUFSP value that is smaller than the BUFSP value specified in the DEFINE statment, the DEFINE value takes precedence.

Alternatively, you can use the BUFNI and BUFND operands on the DLBL statement instead of the BUFSP operand. For example, a for a 2-level index, code BUFNI=33,BUFND=33.

For further details of BUFSP, BUFNI and BUFND DLBL parameters, see the *VSE/ESA System Control Statements* manual.

The principal factors affecting CICS startup and shutdown times are the number of resources defined in CICS tables, and the number of resources defined in the group list for those definitions managed by RDO.

7 You must supply a single blank data card to initialize the data set with a dummy record containing a blank key.

Reusing the global catalog to perform a cold start

If you need to clear the catalog to perform a completely cold start, you can avoid deleting and redefining the data set by specifying the REUSE option on the cluster definition. If you specify REUSE on the cluster definition, you can include in the CICS cold start-job stream a job step to reinitialize the global catalog, as shown in the INITGCD job step in Figure 49 on page 152. To reinitialize the global catalog without redefining it, you must specify the REUSE option on the DEFINE CLUSTER command, and on the REPRO command of the INITGCD job step, as shown.

Be aware, however, that a start initiated by START=COLD is not entirely without reference to the previous run of a CICS system using the same global catalog. For example, CICS checks the global catalog for any data-set-name-block entries for VSAM data sets for which backout failures have occurred. This information is normally preserved across a cold start, but if you re-initialize the global catalog it is lost. For more information about the use of the global catalog in a cold start of CICS, see “Classes of start and restart” on page 209.

Space calculations

Each global catalog keypoint record has a 28-byte key.

To estimate the amount of space needed in your global catalog to keypoint installed resource definitions, table entries, and control blocks, use the sizes specified in Table 23.

Each entry is one VSAM record, and the records for each type of table have different keys.

There is also a restart control record.

The space requirements for a VSAM KSDS such as DFHGCD can vary for different CICS cold starts. This can occur even if no changes have been made to the CICS definitions to be stored on the VSAM KSDS. This is because VSAM will utilize the space in the data set differently depending on whether the data set has just initialized, or has data from a previous run of CICS. CICS will call VSAM to perform sequential writes. VSAM honours the ‘freespace’ value specified on the data set’s definition if the keys of the records being added sequentially are higher than the highest existing key. However, if the data set contains existing records with a higher key than the ones being inserted, ‘freespace’ is only honoured once a CI split has occurred.

The size of the index portion of the data set may also vary depending on the number of CI and CA splits that have occurred. This affects the index sequence set.

Table 23 (Page 1 of 2). Sizes for entries in the global catalog

Installed definition, table entry, or control block	***Number of bytes per entry (inc key)***
Installed PARTNER definitions	118 bytes
Installed PROGRAM definitions	44 bytes

Table 23 (Page 2 of 2). Sizes for entries in the global catalog

Installed definition, table entry, or control block	***Number of bytes per entry (inc key)***
Installed TRANSACTION definitions (without TPNAME)	140 bytes
Installed TRANSACTION definitions (with TPNAME or XTPNAME)	204 bytes
Installed TRANCLASS definitions	36 bytes
Remote File (DAM or VSAM)	60 bytes
DAM file control table entry (FCT)	212 bytes
DAM DTF control block	256 bytes
VSAM file or data table (FCT)	322 bytes
VSAM LSR control blocks 4	1180 bytes
Data set names (JCL or dynamically allocated)	72 bytes
Data set name blocks	108 bytes
Intrapartition transient data queue (DCT)	164 bytes
Terminal control table entry (TCT)	1
Installed model TERMINAL definitions 2	360 bytes
Dump Table Entry	37 bytes
Installed TYPETERM definitions 2	362 bytes
Automatic initiator descriptor (AID)	176 bytes
Interval control element (ICE)	176 bytes
Unit of recovery descriptors (URD)	160 bytes
Deferred work element (DWE)	108 bytes

Notes for Table 23 on page 154:

1 The sizes to use for calculating the space for TCT entries are:

Table 24. Sizes to use when calculating space for TCT entries (inc key)

Type of definition or TCT entry	***Number of bytes per entry***
The CICS-generated TCT entries (2 only)	696 (total)
VTAM terminals	829
Non-3270 devices with pipeline logical units and TASKLIMIT (nn) specified	653 x TASKLIMIT
VSE consoles	445
LUTYPE6.2 connection	314
LUTYPE6.2 mode	201 + (1031 x maximum number of sessions)
LUTYPE6.1 connection	261 + (835 x number of sessions)
IRC	265 + (567 x number of sessions)
EXCI connection	265 + (612 x number of sessions)

2 The RDO TYPETERM and model TERMINAL definitions are present if you are using autoinstall. They are stored directly in the global catalog when the definitions are installed, either by a CEDA transaction, or as members of a group installed via a group list. For example, if you start up CICS with the startup parameter GRPLIST=DFHLIST, the CICS-supplied TYPETERM and model terminal definitions, defined in the groups DFHTYPE and DFHTERM, are recorded in the global catalog. Allow space in your calculations for all autoinstall resources installed in your CICS region.

3 The value given is for a DWE chained off an LU6.1 TCTTE session, or an APPC TCTTE session.

4 One for each LSR pool, that is 15.

Job control statement for CICS execution

If you define the global catalog using the sample job in Figure 49 on page 152, the data definition statement for the CICS execution is:

```
// DLBL  DFHGCD,'CICS410.applid.DFHGCD',,VSAM,CAT=CICSUCT
```

DLBL VSAM tuning options are described in the *VSE/ESA System Control Statements* manual.

An example is shown in the CICS startup job stream in Chapter 23, “CICS startup” on page 289.

The local catalog

CICS Transaction Server for VSE/ESA Release 1 is divided into functional areas (or components) known as **domains**. These domains communicate through a central component, the CICS kernel, and their initialization and termination is controlled by the domain manager. The kernel, the domain manager, and the other domains all require an individual domain parameter record, and these are stored in the local catalog.

The CICS domains use the local catalog to save some of their information between CICS runs, and to preserve this information across a cold start. For further guidance information about what is written to the local catalog, and about how CICS uses the local catalog for startup and restart, see “Classes of start and restart” on page 209.

The local catalog is a VSAM key-sequenced data set (KSDS). It is not shared by any other CICS region, such as an alternate CICS in an XRF environment. If you are running CICS with XRF, you must define a unique local catalog for the active CICS region, and another for the alternate CICS region.

Unlike the global catalog, which must be defined with enough space to cope with any increase in installed resource definitions, the size of the local catalog is relatively static. The following section describes the information held on the local catalog.

Information written to the local catalog

Before the local catalog can be used to bring up a CICS region, it must be initialized with the following data:

- The domain manager parameter records, each of which contains information relating to one of the CICS domains. These records are identified by their domain names, which are:

Domain name in catalog	**Description**
DFHAP	Application domain
DFHCC	CICS local catalog domain
DFHDD	Directory manager domain

DFHDM	Domain manager domain
DFHDS	Dispatcher domain
DFHDU	Dump domain
DFHGC	CICS global catalog domain
DFHKE	Kernel domain
DFHLD	Loader domain
DFHLM	Lock manager domain
DFHME	Message domain
DFHMN	Monitoring domain
DFHPA	System initialization parameter domain
DFHPG	Program manager domain
DFHSM	Storage manager domain
DFHST	Statistics domain
DFHTI	Timer domain
DFHTR	Trace domain
DFHUS	User domain
DFHXM	Transaction manager domain
DFHXS	Security domain.

- Three loader domain parameter records, which contain information relating to:
  - DFHDMP, the CSD file manager
  - DFHEITSP, the RDO language definition table
  - DFHPUP, the CSD parameter utility program.

To enable you to initialize the local catalog correctly, with all the records in the correct sequence, there is a CICS-supplied utility called DFHCCUTL that you run immediately after you have defined the VSAM data set.

In addition to the information written to the local catalog when you first initialize it, the loader domain writes a program definition record for each CICS nucleus module. The number of records varies depending on the level of function you have included in your CICS region (for example, coding ISC=YES as a system initialization parameter adds a few more records). Allow for at least 225 of these loader-domain records.

Some domains also write a domain status record to the local catalog, for use in a warm or emergency restart. For example, in the case of the dump domain, the status record indicates which transaction dump data set was in use during the previous run. The domains that write to the local catalog are:

- Storage manager domain
- Dispatcher domain
- Message domain
- Dump domain.

You can add records to the local catalog to enable the CICS self-tuning mechanism for storage manager domain subpools. See the *CICS Operations and Utilities Guide* for more information on the DFHSMUTL utility.

Finally, when you define the VSAM cluster for the local catalog, specify a secondary extent value as a contingency allowance. See the sample job in Figure 50 on page 158.

Job control statements to define and initialize the local catalog

Before its first use, you must define and initialize the CICS local catalog as a VSAM key sequenced data set (KSDS). To do this, you can use the sample job in Figure 50.

```
// JOB DEFLCD CREATE LOCAL CATALOG DATA SET
// DLBL CICSUCT,'usercat',,VSAM
// DLBL DFHLCD,'CICS410.applid.DFHLCD',,VSAM,CAT=CICSUCT
// EXEC IDCAMS,SIZE=AUTO
   DEFINE CLUSTER                   -
            (NAME(CICS410.applid.DFHLCD)         -
            KEYS(28 0)              -
            INDEXED                 -
            RECORDS(300 10)         -  1
            VOLUME(volid)           -
            RECSZ(40 72)            -  2
            REUSE                   -
            FSPC(10 10)             -
            SHR(2) )                -
          DATA                      -
            (NAME(CICS410.applid.DFHLCD.DATA) -
            CISZ(8192))             -
          INDEX                     -
            (NAME(CICS410.applid.DFHLCD.INDEX))  -
          CATALOG(usercat)
/*
// IF $RC GT 4 THEN
// GOTO $EOJ
* Initialise the LCD
// EXEC DFHCCUTL,SIZE=DFHCCUTL
/*
```

Figure 50. Sample job to define and initialize the local catalog

Notes for Figure 50:

1 Space for about 300 records should be adequate for the local catalog, but also specify space for secondary extents as a contingency allowance.

2 The local catalog records are small by comparison with the global catalog. Use the record sizes shown, which, in conjunction with the number of records specified, ensure enough space for the data set.

Job control statement for CICS execution

If you define the local catalog using the sample job in Figure 50, the data definition statement for the CICS execution is:

```
// DLBL  DFHLCD,'CICS410.applid.DFHLCD',,VSAM,CAT=CICSUCT.
```

Chapter 17. Defining and using auxiliary trace data sets

This chapter describes the auxiliary trace data sets controlled by CICS.

There are several types of tracing available in CICS to help you with problem determination, and these are described in the *CICS Problem Determination Guide*. Among the various types of trace, the CICS tracing handled by the CICS trace domain allows you to control the amount of tracing that is done, and also to choose from any of two destinations for the trace data. Any combination of these two destinations can be active at any time:

1. The internal trace table, in main storage above the 16MB line in the CICS address space.
2. The auxiliary trace data sets, defined as SAM data sets on disk or tape.

For information about using CICS tracing for problem determination, see the *CICS Problem Determination Guide*.

Defining auxiliary trace data sets

If you decide to use auxiliary trace, you must define one or two sequential data sets, on either disk or tape. If you specify automatic switching for your auxiliary trace data sets, define two data sets. If you specify autoswitch for auxiliary trace, and define only one data set, auxiliary trace is stopped and CICS takes a dump.

The data set names of the auxiliary trace data sets are defined by CICS as DFHAUXT and DFHBUXT. If you define a single data set only, its data set name must be DFHAUXT. You must define the auxiliary trace data sets before starting CICS.

If your auxiliary trace data set is on tape, you must specify device SYS009 in the ASSGN job control statement. If a labeled tape is used, the TLBL name for the data set should be DFHAUXT.

Starting and controlling auxiliary trace

You may need to use auxiliary trace data sets to avoid the loss of diagnostic information, because internal trace table entries wrap around. When the end of the internal trace table is reached, subsequent entries overwrite those at the start of the table. To get a trace of CICS activity in which trace entries are not overwritten, use the auxiliary trace data sets. This can be particularly useful if you are using CICS trace during startup, because of the high volume of trace entries written when CICS is initializing.

You can start CICS tracing at initialization by coding system initialization parameters; you can also specify which destination you want CICS to use. The following is a summary of the trace system initialization parameters that you can code:

Keyword	**Description**
AUXTR	Switches auxiliary trace on or off at CICS startup.

AUXTRSW	Specifies automatic switching for auxiliary trace data sets when full.
INTTR	Switches internal trace on or off at CICS startup.
TRTABSZ	Defines the size of the CICS internal trace table.
SPCTR	Specifies the level of special tracing.
SPCTRxx	Specifies the level of special tracing for the “xx” component.
STNTR	Specifies the level of CICS standard tracing.
STNTRxx	Specifies the level of standard tracing for the “xx” component.
SYSTR	Switches the system master trace flag on or off at CICS startup.
USERTR	Switches the user trace flag on or off at CICS startup.

For more information about these system initialization parameters, and how to code them, see Chapter 22, “CICS system initialization” on page 187.

You can also control CICS tracing by means of the CICS-supplied transactions CETR and CEMT. (Note that you cannot use CETR through a VSE console.) For guidance information about the CICS control options available with **CETR** and **CEMT**, see the *CICS-Supplied Transactions* manual.

Job control statements for CICS execution

To define your auxiliary trace data sets on disk, you can use the job control statements shown in Figure 51 in the CICS startup job.

```
// DLBL DFHAUXT,'CICS410.applid.DFHAUXT',0,SD      1
// EXTENT SYSnnn,volid,1,0,rtrk,ntrks      2
// DLBL DFHBUXT,'CICS410.applid.DFHBUXT',0,SD
// EXTENT SYSmmm,volid,1,0,rtrk,ntrks      2
// ASSGN SYSnnn,cuu                        3
// ASSGN SYSmmm,cuu
```

Figure 51. JCL to define your auxiliary trace data sets on disk

Notes:

1 Entering the date value of 0 prevents the generation of VSE/ESA message 4933D:

```
4933D - EQUAL FILE ID IN VTOC
```

during execution of the trace program. See the *VSE/ESA Messages and Codes* manual for information about this message.

2 In this sample JCL, `rtrk` is the relative track number for the start of the extent (or relative block number if you are using an FBA device). `ntrks` is the number of tracks allocated for the extent (or number of blocks allocated for the extent if you are using an FBA device).

3 If you place the auxiliary trace data set on ***tape***, you **must** use the following job control statement:

```
// ASSGN SYS009,cuu
```

Note: The ASSGN statement **must** specify SYS009. If you omit the ASSGN statement, a tape unit is dynamically acquired. However, you should be aware that some other products may use SYS009 for other functions (for example, ICCF uses SYS009 as the default for the system console) and so, if you intend placing the CICS auxiliary trace data set on tape, ensure that you change the assignments for other products so that only CICS uses SYS009 for its tape unit.

You should assign the tape unit and mount a tape reel before entering the master terminal command to turn on the auxiliary trace. If you intend using a labeled tape volume, you need to add a TLBL statement as follows:

```
// TLBL DFHAUXT,'CICS410.applid.DFHAUXT'
```

Space calculations

Trace entries are of variable length, but the physical record length (block size) of the data written to the auxiliary trace data sets is fixed at 4096 bytes. As a rough guide, each block contains an average of 40 entries, although the actual number of entries depends on the processing being performed.

A track contains 10 blocks for a 3380 device, and 12 blocks for a 3390 device.

Auxiliary trace data sets as VSAM-managed SAM files

CICS auxiliary trace data sets are also supported as SAM files managed in VSAM space, using the VSE/VSAM Space Management for SAM feature. However, in common with dump data sets, you can only use a primary VSAM allocation for auxiliary trace data sets defined using this feature. There is no support for secondary allocations when the primary data set is full.

Further information about this feature can be found in the VSAM documentation.

Job control statements for defining VSAM-managed SAM files

For examples of defining CICS data sets that are supported in VSAM-managed space, see the descriptions given for the dump data sets in “Dump data sets as VSAM-managed SAM files” on page 166. Note that the block size in the VSE file definition for the auxiliary trace data sets is 4096 bytes, and this is the recommended value for the RECORDSIZE parameter on the DLBL statement.

XRF considerations

The active and the alternate CICS regions must refer to different auxiliary trace data sets; that is, they must be unique data sets. This means that you can capture auxiliary trace data for the active CICS region, while the alternate CICS region is running but before takeover occurs.

To control auxiliary trace data sets for the active CICS region, see the descriptions of the CEMT and CETR transaction in the *CICS-Supplied Transactions* manual. For the alternate CICS region, see the description of the CEBT transaction in the *CICS-Supplied Transactions* manual.

Trace utility program (DFHTU410)

If you write trace entries to CICS auxiliary trace data sets you can use the trace utility program, DFHTU410, to extract all or selected trace entries, and format and print the data.

To process the separate trace data sets for active and alternate CICS regions, you need separate DFHTU410 utility jobs for each set of data sets. For more information about DFHTU410, see the *CICS Operations and Utilities Guide*.

Attention!

If you use CICS auxiliary tracing facilities and process the results using DFHTU410, close your auxiliary trace data set *before* you run DFHTU410. If you don't do this, you can get data that is unrelated to the CICS run you were tracing in the output from DFHTU410.

This can happen if CICS does not completely fill the auxiliary trace data set with trace entries. Data, previously in the space now defined as the auxiliary trace data set, that has not been overwritten with trace entries is included in DFHTU410 processing. DFHTU410 does not distinguish between CICS trace entries and other types of data but formats and prints **all** the data it finds in the auxiliary trace data set.

Chapter 18. Defining dump data sets

This chapter describes how to define the following two types of dump data sets that CICS uses for recording dumps as a consequence of a failure detected during CICS execution, or upon explicit request:

1. CICS transaction dump data sets, for recording transaction dumps
2. VSE system dump (SYSDUMP) libraries, for recording system dumps that CICS requests using the VSE SDUMPX macro.

CICS has a dump table facility that enables you to control dumps. The dump table lets you:

- Specify the type of dump, or dumps, you want CICS to record
- Suppress dumping entirely
- Specify the maximum number of dumps to be taken during a CICS run
- Control whether CICS is to terminate as a result of a failure that results in a dump.

Setting dump options

You can set the options you want in the dump table in two ways:

1. Using the CEMT SET SYDUMPCODE and TRDUMPCODE master terminal commands. See the *CICS-Supplied Transactions* manual for further details.
2. Using the EXEC CICS SET SYSDUMPCODE and TRANDUMPCODE system programming commands. See the *CICS System Programming Reference* for details of the Systems Programming Interface.

When you start CICS for the first time, CICS uses system default values for the dump table options, and continues to use the system default values until you modify them with a CEMT or EXEC CICS command. For information about the dump table options you can set, see the *CICS Problem Determination Guide*.

System dumps

CICS produces a system dump using the VSE SDUMPX macro.

VSE SDUMPX macro

A VSE SDUMP dump results from CICS issuing a VSE SDUMPX macro. It includes almost the entire CICS address space, that is, the VSE nucleus and other common areas, as well as the CICS private storage areas. The dump is written to a SYSDUMP sublibrary, which you can process using Info/Analysis. For information about Info/Analysis and the associated VSE SYSDUMP libraries, see the *VSE/ESA Diagnostic Tools* manual.

If you are running CICS with XRF, the surveillance signal of the active CICS system stops during a VSE SDUMP of the active CICS region's address space, which could lead to unnecessary takeovers being initiated, if the ADI (alternate delay interval) for the alternate is set too low.

Suppressing system dumps that precede ASRx abends

The VSE system dump data sets can become full with unwanted SDUMPs that precede ASRA, ASRB and ASRD abends (after either message DFHAP0001 or DFHSR0001).

If CICS storage protection is active, (STGPROT system initialization parameter), you can suppress the system dumps caused by errors in application programs (after message DFHSR0001), while retaining the dumps caused by errors in CICS code (after message DFHAP0001). To do this, use either a CEMT SET SYDUMPCODE command, or an EXEC CICS SET SYSDUMPCODE command to suppress system dumps for system dumpcode SR0001.

```
CEMT SET SYDUMPCODE(SR0001) ADD NOSYSDUMP
```

CICS uses dumpcode SR0001 if an application program was executing in user-key at the time of the program check or VSE abend. This is only possible if storage protection is active. If the program was executing in CICS-key, dumpcode AP0001 is used instead.

Where storage protection is not active, SDUMPs can be suppressed by suppressing dumpcode AP0001. However, note that this suppresses dumps for errors in both application *and* CICS programs.

For more information about the storage protection facilities available in CICS Transaction Server for VSE/ESA Release 1, see “Storage protection” on page 301.

If you want SDUMPs for one of these transaction abends but not the other, select the one you want by using either a CEMT TRDUMPCODE or an EXEC CICS TRANDUMPCODE command. This specifies, on an entry in the dump table, that SDUMPs are to be taken for either ASRA, ASRB, or ASRD abends. For example, specifying:

```
CEMT SET TRDUMPCODE(ASRB) ADD SYSDUMP
```

adds an entry to the dump table and ensures that SDUMPs are taken for ASRB abends. However, in this case the SDUMPs are taken at a later point than SDUMPs usually taken for system dump code AP0001 and SR0001.

For information about the DFHAP0001 and DFHSR0001 messages, see *VSE/ESA Messages and Codes Volume 3*.

Processing system dumps

You process a system dump using VSE Info/Analysis services. See the *VSE/ESA Diagnosis Tools* manual for further information about Info/Analysis.

The CICS transaction dump data sets

CICS records transaction dumps on a sequential data set, or a pair of sequential data sets, on disk or tape. The data sets must be defined in the CICS run with the DLBL names DFHDMPA and DFHDMPB, but if you define a single data set only, its DLBL name must be DFHDMPA. In this chapter, a reference to “the CICS dump data set” means either DFHDMPA or DFHDMPB. Note that CICS always attempts to open at least one transaction dump data set during initialization. If you do not include a DLBL statement for at least one transaction dump data set in your CICS job, initialization continues after the following message is sent to the console:

```
DFHDU0306  applid Unable to open Transaction Dump Data set
           DFHDMPx where "x" can have the value A or B
```

With two data sets, you can print transaction dumps from one data set while CICS is running. To do this, first use CEMT SET DUMP SWITCH to switch the data sets. CICS closes the current data set after any transaction dump being recorded has been completed, and opens the other data set. You can print the completed data set with the DFHDU410 dump utility program. For information about the DFHDU410 dump utility program, see the *CICS Operations and Utilities Guide*.

In addition to switching dump data sets explicitly, the operator can use CEMT SET DUMP AUTO to cause automatic switching when the current data set becomes full. (Note that this permits ***one*** switch only.) When a transaction dump data set is full, CICS closes the data set and issues console messages as follows:

```
DFHDU0303I applid Transaction Dump Data set DFHDMPx closed.
DFHDU0304I applid Transaction Dump Data set DFHDMPy opened.
DFHDU0305I applid Transaction Dump Data set switched to DFHDMPy
```

where “x” and “y” can have the value A or B.

Global user exits for dumps

There are four global user exits that you can use with transaction dump data sets:

1. XDUCLSE, after the dump domain has closed a transaction dump data set
2. XDUREQ, before the dump domain takes a transaction dump
3. XDUREQC, after the dump domain takes a transaction dump.
4. XDUOUT, before the dump domain writes a record to the transaction dump data set.

For programming information about the global user exits, see **XDUCLSE**, **XDUREQ**, **XDUREQC**, and **XDUOUT** in the *CICS Customization Guide*.

Selecting the transaction dump data set at startup

You can code the DUMPDS system initialization parameter to specify which transaction dump data set is to be opened during CICS initialization. If you specify DUMPDS=AUTO, CICS opens, on a warm or emergency start, the data set that was ***not*** in use when CICS was last terminated. This lets you restart CICS after an abnormal termination without waiting to print the dump data set that was in use at termination. For more information about the DUMPDS parameter, see Chapter 22, “CICS system initialization” on page 187.

Copying disk dump data sets to tape

If you intend copying dump data sets to tape or disk, copy them with the same attributes as the dump distribution file, that is, variable-block, with a logical record length of 4092, and a block size of 4096.

Space calculations

For the initial installation of CICS, a dump data set of between 5 and 10MB should be enough. When normal operation begins, you can adjust this to suit your own installation's requirements.

Job control statements for CICS execution

You can include the sample job control statements in Figure 52 in your CICS start up job stream:

```
// DLBL DFHDMPA,'CICS410.applid.DFHDMPA',0,SD      1
// EXTENT SYSnnn,volid,1,0,rtrk,ntrks
// DLBL DFHDMPB,'CICS410.applid.DFHDMPB',0,SD
// EXTENT SYSnnn,volid,1,0,rtrk,ntrks      2
// ASSGN SYSnnn,cuu
```

Figure 52. Sample JCL for CICS execution

Notes for Figure 52:

1 On the DLBL statement, specify the date operand as 0 to avoid the "EQUAL FILE ID" message occurring. This message can occur during execution of the dump utility program (DFHDUP), or your CICS startup, if you did not specify 0 and the expiry date has not been reached:

```
4933D EQUAL FILE ID IN VTOC DFHDMPB SYSnnn=cuu volid - user.dfhdmpb
```

2 Substitute your own values for `rtrk` and `ntrks` to define the relative start track and number of tracks of each file extent; and also for `nnn` for the logical unit number, and `cuu` to supply the unit address.

The CICS dump control program defines the dump data set BLKSIZE as 7161, and the CISIZE for FBA devices as 7168; you ***cannot*** vary these sizes by either of the BLKSIZE or CISIZE parameters on a job control DLBL statement.

Dump data sets as VSAM-managed SAM files

CICS dump data sets are also supported as SAM files managed in VSAM space, using the VSE/VSAM Space Management for SAM feature.

Job control statements for defining VSAM-managed SAM files

To avoid having to define specific extent information, you can define your CICS dump data sets in VSAM-defined and cataloged space. To use this feature you must first define an ESDS.SAM model in VSAM space. The job shown in Figure 53 on page 167 is an example of this.

```
// JOB DVSAMSAM
* JOB to DEFINE ESDS.SAM models in VSAM-managed space.
* -----------------------------------------------
*  Step 1:  Delete old cluster for ESDS.SAM Models
* -----------------------------------------------
//  EXEC IDCAMS,SIZE=AUTO
     DELETE (DEFAULT.MODEL.ESDS.SAM)  -
     CLUSTER                          -
     PURGE                            -
     CATALOG (usercat)
/*
* -----------------------------------------------
*  Step 2:  Define new cluster for ESDS.SAM Models
* -----------------------------------------------
//  EXEC IDCAMS,SIZE=AUTO
     DEFINE CLUSTER (NAME(DEFAULT.MODEL.ESDS.SAM) -
            VOLUMES(volid)            -
            RECORDS(1 1)              -
            RECORDSIZE(2000 2000)     -
            RECORDFORMAT(UNDEF)       -
            REUSE                     -
            SPEED                     -
            NOALLOCATION              -
            NONINDEXED)               -
     CATALOG (usercat)
/*
/&
```

Figure 53. Sample job to define an ESDS.SAM model in VSAM

When you have successfully defined the ESDS.SAM model, define your CICS dump data sets with disk label information in your CICS startup job stream. The dump data set DLBL statements provide the suballocation information that VSAM needs to complete the space allocation. This is illustrated in Figure 54.

```
// DLBL CICSUCT,'usercat',,VSAM
// DLBL DFHDMPA,'%CICS410.applid.DFHDMPA',0,VSAM,CAT=CICSUCT,                *
               DISP=(NEW,KEEP),RECORDS=200,RECSIZE=7161  1 2 3
// DLBL DFHDMPB,'%CICS410.applid.DFHDMPB',0,VSAM,CAT=CICSUCT,                *
               DISP=(NEW,KEEP),RECORDS=200,RECSIZE=7161  1 2 3
```

Figure 54. DLBL statements needed by VSAM to complete space allocation

Notes for Figure 54:

1 The % symbol specifies a unique partition identification. VSAM completes the suballocation in VSAM space using the information provided in the DEFAULT.MODEL.ESDS.SAM cluster definition. The actual cluster that is created for the dump data set from this DLBL statement has the partition ID added to the name. For example, in the statement illustrated in Figure 54, the clusters are cataloged as `CICS410.DFHDMPA.Fn` and `CICS410.DFHDMPB.Fn` where `n` is the partition number. If the VSAM catalog is shared across VSE systems, the file-id must have the prefix %%.

If you have to code a DLBL statement over two lines, put an asterisk (*) in column 72 and begin the second line in column 16.

2 The DLBL statement provides the RECORDS and RECSIZE information that VSAM needs to complete the space allocation. You are not concerned (as you are in the case of the normal VSE SAM file) with supplying specific location information in terms of relative track or block number.

3 The RECORDS parameters in the DLBL job control statements specify a value for a primary extent only: there is no support for secondary extents if a VSAM-managed dump data set is full.

General note:

If you allow the date operand for the retention period to default (7 days), you receive the following VSE warning message if the retention period has not expired when you restart CICS:

```
4233A  EQUAL FILE-ID IN CATALOG (filename| SYSnnn=cuu|volid)
```

Type "DELETE" in reply to this message: VSAM deletes and reallocates the dump data sets, and processing continues.

Chapter 19. Defining the CICS availability manager data sets

This chapter tells you how to define the CICS availability manager (CAVM) data sets. The CAVM is the mechanism that enables active and alternate CICS regions to coordinate their processing when XRF=YES is coded as a system initialization parameter. If you code XRF=NO, these data sets are not used. The CAVM requires two data sets: the XRF control data set and the XRF message data set.

This pair of data sets is logically a single entity that contains:

- State data whose main purpose is to ensure that, at any given time, only one job is allowed to fulfill the active role for a particular generic APPLID
- Primary and secondary surveillance signals of active and alternate CICS regions, so that each CICS region can tell whether its partner is working correctly
- Messages about the state of particular resources in use on the active CICS region, that are written by the active CICS region, and read and processed by the alternate CICS region.

Both the active and alternate CICS regions must refer to the same pair of data sets. You define these data sets, but must not try to initialize them, and you are recommended to place the data sets on separate volumes, and **not** on the volume that contains the VSE lock file. The first time they are used, CICS recognizes them as a new pair of data sets. If they are new, CICS initializes them in such a way that, from then on, they can be used only as a pair with the original generic APPLID and for their original purpose (that is, as either an XRF message data set or an XRF control data set). If you need to redefine either data set, for any reason, you must redefine both of them.

You must define a separate pair of data sets for each generic APPLID in use. If a CICS complex consists of, for example, five regions, five pairs of data sets must be defined.

You do not need to take backup copies of these data sets because when neither of the active or alternate CICS regions is running, you can always start with a fresh pair of data sets.

Having two data sets is beneficial because if the access paths to the two volumes are separate, the CAVM is less vulnerable to hardware failures.

The XRF control data set

The XRF control data set is used:

- To record the presence or absence, identities, and current states of the active and alternate CICS regions' jobs
- For the primary surveillance signals of the active and the alternate CICS regions.

CAVM rejects a request from a CICS job to sign on as the active CICS region if the XRF control data set shows that an active CICS region is already present, or that a takeover is in progress. This ensures that the integrity of files and databases cannot be lost as a result of uncontrolled concurrent updating by two or more active CICS regions. As soon as an active or alternate CICS regions signs on, it starts to write its own surveillance signals, and to look for its partner's surveillance signals.

JCL to define the XRF control data set

You must define the XRF control data set, but not initialize it. You can use the JCL statements in Figure 55 to define the XRF control data set.

```
// JOB  CICSCTL
// DLBL IJSYSUC,'usercat',,VSAM
// EXEC IDCAMS,SIZE=AUTO
       DEFINE CLUSTER -
                (NAME(CICS410.applid.DFHXCTL) -
                 RECORDSIZE(4089 4089)        -                 1
                 CONTROLINTERVALSIZE(4096)    -                 2
                 RECORDS(4)                   -
                 NONINDEXED                   -
                 SHAREOPTIONS(3,3)            -                 3
                 VOLUMES(volid1))             -
              DATA                            -                 4
                (NAME(CICS410.applid.DFHXCTL.DATA)) -
                CAT(usercat)
/*
/&
```

Figure 55. Sample job to define the XRF control data set

Notes for Figure 55:

1 The RECORDSIZE must be at least 4089.

2 The control interval sizes of the XRF control data set and the XRF message data set must be equal, and at least 4096 bytes.

3 The SHAREOPTIONS must be specified as 3,3.

4 The data set must be VSAM-ESDS.

Serializing access to the XRF control data set

Access to the XRF control data set must be serialized during the critical sections of CAVM signon, sign-off, and takeover processing. This serialization is provided by the VSE LOCK/UNLOCK logic. Therefore, it would be unwise to place an XRF control data set on the same volume as the VSE lock file.

Space calculations

Only four control intervals are needed.

Job control statements for CICS execution

The data set name required for CICS execution is DFHXCTL. The following is an example of the JCL statement required:

```
// DLBL  DFHXCTL,'CICS410.applid.DFHXCTL',,VSAM,CAT=CICSUCT
```

The XRF message data set

The XRF message data set is used:

- Mainly to pass messages about the current states of specific resources from the active to the alternate CICS region
- For the secondary surveillance signals of the active and alternate CICS regions, when the control data set is unavailable for this purpose, either because the last write has not completed yet or because of I/O errors.

JCL to define the XRF message data set

Like the XRF control data set, the XRF message data set must be defined but not initialized. You can use the sample job in Figure 56 to define the XRF message data set.

If you the sample JCL in Figure 56, read the accompanying notes; the other options shown are suggestions only.

You should define the XRF message data set on a volume that does not contain the VSE lock file, and should not locate it where a single failure can make both it and the XRF control data set inaccessible. This reduces the risk of the surveillance signal being stopped accidentally while CICS is still running normally.

The XRF message data set is reserved for a short time for formatting when CICS uses it for the first time.

```
// JOB CICSMSG
// DLBL IJSYSUC,'usercat',,VSAM
// EXEC IDCAMS,SIZE=AUTO
       DEFINE CLUSTER -
                (NAME(CICS410.applid.DFHXMSG) -
                 RECORDSIZE(4089 4089)        -                1
                 CONTROLINTERVALSIZE(4096)    -                2
                 RECORDS(1500)                -
                 NONINDEXED                   -
                 SHAREOPTIONS(3,3)            -                3
                 VOL(volid2)                  -
              DATA                            -                4
                (NAME(CICS410.applid.DFHXMSG.DATA)) -
                 CAT(usercat)
/*
/&
```

Figure 56. Sample job to define the XRF message data set

Notes for Figure 56:

1 The RECORDSIZE must be at least 4089.

2 The control interval sizes of the XRF message data set and the XRF control data set must be equal, and at least 4096 bytes.

If the CI size of the XRF message data set is greater than 4096, the CI buffers occupy more real storage and virtual storage above the 16MB line, although fewer I/O operations occur during the “catch-up” phase.

3 The SHAREOPTIONS must be specified as 3,3.

4 The data set must not be indexed.

Space calculations

It is difficult to give a simple answer to the question: “How big should my XRF message data set be?”. A simple answer is that the size required depends on the length and number of messages that have been sent by the active CICS region but not yet received by the alternate CICS region.

The XRF message data set is written and read cyclically. When the alternate CICS region has read a message, that space becomes available for another message on the next cycle. It is important to make the data set large enough to store the backlog of messages that accumulates if the alternate CICS region is held up for any reason. If the data set is too small, you run the risk of the alternate CICS region being unable to read the data set correctly, and thereby becoming incapable of taking over. However, the active CICS region does not write messages to the data set until it has been notified that an alternate CICS region is present (signed on to CAVM), and able to receive them.

The peak in message traffic usually occurs during the “catch-up” phase shortly after the active CICS region detects the presence of the alternate CICS region. You may be able to estimate the amount of space you need from the number of terminal resources you have. The active CICS region sends messages about these resources:

- Installed:
  - VTAM terminals
  - ISC connections
  - MRO connections
  - EXCI connections
  - VSE Consoles
- Autoinstalled terminals

Table 25 lists the sizes of the various messages sent to the data set.

Table 25 (Page 1 of 2). Sizes of messages sent to the XRF message data set

Type of TCT entry	***Bytes per install***
The CICS-generated TCT entries (2 only)	696
VTAM terminals	829
Non-3270 devices with pipeline logical units and TASKLIMIT (nn) keyword specified	653 x TASKLIMIT value
VSE consoles	445
LUTYPE6.2 connection	314

Table 25 (Page 2 of 2). Sizes of messages sent to the XRF message data set

Type of TCT entry	***Bytes per install***
LUTYPE6.2 mode	201 + (1031 x maximum number of sessions)
LUTYPE6.1 connection	261 + (835 x number of sessions)
IRC	265 + (567 x number of sessions)
EXCI connection	265 + (612 x number of sessions)

For VTAM terminals only, you should also make allowance for the following:

Table 26. Additional space requirements (VTAM terminals only)

Bytes per logon	***Bytes per logoff***	***Bytes per signon***	***Bytes per sign-off***
70	35	45	29

The alternate CICS region issues some messages that can help you with your sizing. The following messages issued by the alternate CICS region can give you an idea of the rate of message transfer:

```
DFHTC1041I applid TERMINAL CONTROL TRACKING STARTED
DFHTC1040I applid TERMINAL CONTROL TRACKING RECORDS RECEIVED
DFHTC1043I applid TERMINAL CONTROL TRACKING ENDED - nnn RECORDS RECEIVED
```

The following messages may indicate that the XRF message data set is not large enough:

```
DFHXG6447I NON CRUCIAL XRF MESSAGE(S) DISCARDED
DFHXA6541I XRF HAS FAILED. THE XRF MESSAGE READER IN THE ALTERNATE
           SYSTEM HAS FALLEN TOO FAR BEHIND
```

Crucial and non-crucial messages

The active CICS region classifies its messages as crucial or non-crucial. An example of a crucial message is an autoinstall message that the alternate CICS region must receive if it is to remain eligible to take over. An example of a non-crucial message is a logon message. The alternate CICS region can tolerate the loss of such a message, and the loss only results in some degradation at takeover; no standby session is established for that terminal and it must be logged on again. Installation messages that form part of the initial description are also treated as non-crucial, because the active CICS region can try to send them again later, and the alternate CICS region can construct its tables from the CICS catalog if it does not receive a complete initial description.

The active discards non-crucial messages if it decides that sending them may overwrite messages that the alternate CICS region has not yet read, thereby making it ineligible to take over. It issues message DFHXG6447I for the first such discard. The active CICS region always sends crucial messages. If this causes an unread message to be overwritten, the alternate CICS region detects it and terminates after issuing message DFHXA6541I.

Effect of a full XRF message data set on the active CICS region

The active CICS region is not affected by the state of the XRF message data set. It continues running even when the data set is full; only the alternate CICS region fails. Further, the XRF message data set is only “full” to the alternate CICS region that fails; you can start a new alternate CICS region, using the same XRF message data set, and the active CICS region resends all the messages for the new alternate CICS region to begin tracking. If the first failure was caused by some unusual condition, you may not need to increase the size of the XRF message data set.

However, if messages DFHXG6447I or DFHXA6541I occur too often, you must stop the active CICS region so that you can change to a larger data set.

Job control statement for CICS execution

The data set name required for CICS execution is DFHXMSG. The following JCL statement can be used:

```
// DLBL DFHXMSG,'CICS410.applid.DFHXMSG',,VSAM,CAT=CICSUCT
```

Security

To ensure that the integrity and security of your CICS regions and terminal network are not compromised, you must protect your XRF data sets using an external security manager (ESM). When you have done so, give each CICS region CONTROL access to its own pair of data sets. If you are running your XRF systems with an overseer program, make sure that it has READ access to all the CAVM data sets. All other users must be denied access to the data sets.

I/O error handling

While the active CICS region can write its **surveillance signals** successfully to either the XRF control data set or the XRF message data set, it keeps running in spite of I/O errors. However, if the active CICS region has an I/O error while writing a message to the XRF message data set, the alternate CICS region cannot function correctly, so the active CICS region disables it to prevent it from taking over. If the active CICS region is unable to write to either the XRF control data set or the XRF message data set, it can neither disable the alternate CICS region nor keep it properly synchronized, and so the active CICS region fails.

While an alternate CICS region can receive the active CICS region’s surveillance signals and tracking messages successfully, in addition to writing its own surveillance signals to either the XRF control data set or the XRF message data set, it keeps running in spite of some types of I/O error. However, an isolated I/O error that would have no effect during tracking, may cause failure of the alternate CICS region if it occurs during takeover.

Note: When the active and alternate CICS regions are running in different VSE images, they are not necessarily affected in the same way by the failure of a control unit or channel path that provides access to a CAVM XRF data set.

Chapter 20. Defining user files

This chapter tells you how to define user files, including VSAM data sets, DAM data sets and data tables.

CICS application programs process files, which, to CICS, are logical views of a physical data set. A file is identified to CICS by a **file name** of up to seven characters and there can be many files defined to CICS that refer to the same physical data set. A data set, identified by a data set name (DSNAME) of up to 44 characters, is a collection of data held on disk. CICS file control processes only VSAM or DAM data sets. These data sets must be created and cataloged, so that they are known to VSE before any CICS job refers to them. Also, the data sets are usually initialized by being preloaded with at least some data before being used by CICS transactions.

You can use CICS shared data tables to improve the performance and function of CICS installations using files that refer to VSAM data sets. CICS Shared data tables offer a method of constructing, maintaining, and gaining rapid access to data records contained in tables held in a data space. Each data table is associated with a VSAM KSDS, known as its **source data set**. For further information about data tables, see “CICS shared data tables” on page 181.

VSAM data sets

You create a VSAM data set by running the VSE/VSAM utility program IDCAMS in a batch job. The IDCAMS DEFINE command specifies to VSAM and VSE the VSAM attributes and characteristics of your data set. You can also use it to identify the catalog in which your data set is to be defined.

If required, you can load the data set with data, again using IDCAMS. You use the IDCAMS REPRO command to copy data from an existing data set into the newly created one.

You can also load an empty VSAM data set from a CICS transaction. You do this by defining the data set to CICS (by allocating the data set to a CICS file), and then writing data to the data set, regardless of its empty state. See “Loading empty VSAM data sets” on page 176.

When you create a data set, you may define a data set name of up to 44 characters. This name, known as the data set name (or DSNAME), uniquely identifies the data set to your VSE system.

VSAM bases and paths

You store data in data sets, and retrieve data from data sets, using application programs that reference the data at the record level.

Depending on the type of data set, you can identify a record for retrieval by its key (a unique value in a predefined field in the record), by its relative byte address, or by its relative record number.

Access to records through these methods of primary identification is known as access via the **base**.

Sometimes you may need to identify and access your records by a secondary or alternate key. With VSAM, you can build one or more alternate indexes over a single base data set, so that you do not need to keep multiple copies of the same information organized in different ways for different applications. Using this method, you create an **alternate index path** (or paths), that links the alternate index (or indexes) with the base. You can then use the alternate key to access the records by specifying the **path** as the data set to be accessed; that is, by allocating the path data set to a CICS file.

When you create a path you give it a name of up to 44 characters, in the same way as a base data set. A CICS application program does not need to know whether it is accessing your data via a path or a base; except that it may be necessary to allow for duplicate keys if the alternate index was specified to have non-unique keys.

Loading empty VSAM data sets

As suggested above, there are several ways you can load data into an empty VSAM data set. VSAM imposes specific restrictions during initial data set load processing, when the data set is said to be in **load mode**. These restrictions apply from the time that a file referencing the empty data set is opened. They continue while records are being loaded, and finish when the file is closed.

An empty data set may be loaded using either of the following methods:

- Running the VSE/VSAM utility program, IDCAMS
- Writing records to the data set using CICS transactions.

Using IDCAMS

If you have a large amount of data to load into a new data set, run the VSE/VSAM utility program IDCAMS as a batch job, using the REPRO command to copy data from an existing data set to the empty data set. When you have loaded the data set with IDCAMS, it can be used by CICS in the normal way.

Note: A data set in VSAM load mode cannot have alternate indexes in the upgrade set. If you want to create and load a data set with alternate indexes, you must use VSE/VSAM, or some other suitable batch program, to load the data set and invoke BLDINDEX to create the alternate indexes.

Using CICS applications

If the amount of data to be loaded is small, and there is no upgrade set, you may load an empty data set using standard CICS file WRITE requests.

When the first write, or series of writes (mass insert), to the file is completed, CICS closes the file and leaves it closed and enabled, so it will be reopened for normal processing when next referenced. If you attempt to read from a file in load mode, CICS returns a NOTFOUND condition.

Reusing data sets

If you define a VSAM data set with the REUSE attribute, it may also be emptied of data during a CICS run. This allows it to be used as a work file. There are two ways in which this can be done:

- When the status of a file referencing the data set is CLOSED and DISABLED (or UNENABLED), you can use the SET FILE EMPTYREQ command, either from an application program using the EXEC CICS command-level interface, or from a master terminal using the master terminal CEMT transaction. The SET FILE EMPTYREQ command sets an indicator in the file control table entry for the file so that, when the file is next opened, the VSAM high-used relative byte address (RBA) is set to zero, and the contents of the data set are effectively cleared.

- If you define a file with the DFHFCT TYPE=FILE macro, and specify SERVREQ=REUSE, the data set referenced by the file is emptied each time CICS opens the file. The file remains in VSAM load mode until explicitly closed. There is no RDO equivalent of SERVREQ=REUSE.

Fixed length records in VSAM data sets

If you define a data set to VSAM with the average and maximum record lengths equal, and define a file to CICS with fixed length records to reference that data set, the size of the records written to the data set ***must*** be of the defined size. For example, if a record in a data set has been read for update, you get errors when rewriting the record if, for example, you:

- Defined the record sizes to VSAM as 250 bytes, with the parameter RECORDSIZE(250 250)
- Defined the file to CICS with the parameter RECFORM=FIXED
- Loaded the data set with records that are only 200 bytes long.

DAM data sets

CICS supports access to keyed and nonkeyed DAM data sets (CKD devices only), but first you must format and load the data set using a batch program. You define the DAM data set in the batch job you run to create it, using DLBL and EXTENT statements, coding the type of file label as DA. For example,

```
// DLBL usrname,'cicsusr.dam.file'DA
// EXTENT SYS001,volid,1,0,rtrk,ntrks
```

For more information about writing data to DAM files, see the *VSE/ESA System Macro User's Guide*.

Defining data sets to CICS

Before CICS can open a file referencing a data set, there must be an installed file definition for that file. For VSAM data sets, the definition can be installed either from the CSD or from a file control table (FCT). DAM files can only be installed from a file control table.

A file is identified by its file name, which you specify when you define the file. CICS uses this name when an application program, or the master terminal operator using the CEMT command, refers to the associated data set.

Each file must also be associated with its data set name in one of the following ways:

- Using JCL in the startup job stream.
- Using dynamic allocation and the DSNAME parameter of the RDO FILE resource definitions (VSAM only).

- Using dynamic allocation and the DSNAME option of a CEMT SET FILE command (VSAM only).
- Using dynamic allocation and the DSNAME option of an EXEC CICS CREATE FILE command (VSAM only).
- Using dynamic allocation and an EXEC CICS SET FILE command in an application program to change the DSNAME (VSAM only).

Using JCL

You can define the data set in a DLBL statement in the JCL of the CICS startup job. The file name must be the same as the file name that refers to the data set. For example, the following DLBL statements would correspond to file definitions for the file names VSAM1A and DAMFILE:

```
// DLBL VSAM1A,'user.vsam.user.file',,VSAM,CAT=usercat
// DLBL DAMFILE,'cicsusr.dam.file,,DA
// EXTENT SYS001,volid,1,0,rtrk,ntrks
```

If you define a data set to CICS in this way, the data set name **cannot** be changed by any of the following dynamic allocation techniques.

Using the DSNAME file resource definition operand

You can define a VSAM data set to CICS by specifying the DSNAME operand when you define the file. You can specify this parameter either using RDO for files, or by using DFHFCT macros. If you want to use DSNAME, do ***not*** provide a DLBL statement for the data set in the startup job stream.

If you use DSNAME on the file resource definition, CICS builds a DLBL dynamically, before the file is opened. At this stage, CICS associates the file name with the data set name. When CICS applications subsequently refer to the data set, they do so by specifying the file name. When you define a data set in this way, the DLBL is automatically deleted by CICS when the file is closed.

For information about using the DSNAME operand, see **Defining a FILE** or **Migrating the FCT to the CSD** in the *CICS Resource Definition Guide*.

Dynamic allocation using CEMT

This method of defining the data set to CICS allows a file definition to be associated with different data sets at different times.

You can set the data set name dynamically in an installed file definition by using the master terminal CEMT command:

```
CEMT SET FILE(filename) DSNAME(datasetname)
```

When you use this command, CICS builds a DLBL for OPEN processing as described above. The DLBL is automatically deleted when the last file entry associated with the data set is closed. Before you can dynamically allocate a file using the CEMT command, the file status must be CLOSED, and also be DISABLED or UNENABLED.

For information about the CEMT SET command, see the *CICS-Supplied Transactions* manual.

Dynamic allocation in an application program

You can assign the data set name dynamically from an application program by using one of:

- EXEC CICS CREATE FILE (filename)
- ATTRIBUTES ('DSNAME(datasetname)...') ATTRLEN(nn)
- EXEC CICS SET FILE(filename) DSNAME(datasetname)

The data set is then dynamically allocated in the same way as if you used the CEMT master terminal command.

For programming information about the EXEC CICS SET command, see the *CICS System Programming Reference* manual.

Opening VSAM or DAM files

Before your application programs can access a file, CICS must first have opened the file using the installed file definition referenced by your program. Part of the process of opening a file is to ensure that the control blocks and buffers required for subsequent processing of the data set are available. If you defined the file to use VSAM local shared resources (LSR), these control blocks and buffers are allocated from the pool of resources. If the LSR pool does not exist at the time of the opening, CICS calculates the requirements and builds the pool before the file is opened. If you defined the file to use nonshared resources, the required control blocks and buffers are allocated by VSAM as part of OPEN processing.

You may need to access a single VSAM data set either through the base or through one or more paths for different access requests. In this case, CICS uses a separate file definition (that is, a separate file), for each of the access routes. Each file definition must be associated with the corresponding data set name (a path is also assigned a data set name). Each file must be open before CICS can access the file using the attributes in its installed file definition. This is because opening ***one*** file for a data set that is logically defined as two or more files with different attributes does not mean that the data set is then available for all access routes.

CICS permits more than one file definition to be associated with the same physical data set name. For example, you may want to define files with different processing attributes that refer to the same data set.

CICS allows or denies access to data in a file, depending on whether the state of the file is ENABLED. An ***enabled*** file that is closed is opened by CICS automatically when the first access request is made. The file remains open until an explicit CLOSE request or until the end of the CICS job.

You can also open a file explicitly by using either of the commands:

```
CEMT SET FILE(filename) OPEN
EXEC CICS SET FILE(filename) OPEN
```

When you use one of these commands, the file is opened irrespective of whether its state is enabled or disabled. You may choose this method to avoid the overhead associated with opening the file being borne by the first transaction to access the file.

You can also specify that you want CICS to open a file immediately after initialization by specifying the OPENTIME(STARTUP) attribute on the RDO FILE definition (or the FILSTAT=OPENED parameter in the DFHFCT macro). If you specify that you want CICS to open the file after startup, and if the file status is ENABLED or DISABLED, the CICS file utility transaction CSFU opens the file. (CSFU does not open files that are defined as UNENABLED: the status of these remains CLOSED, UNENABLED.) CSFU is initiated automatically, immediately before the completion of CICS initialization. CICS opens each file with a separate OPEN request. If a user transaction starts while CSFU is still running, it can reference and open a file that CSFU has not yet opened; it does not have to wait for CSFU to finish.

Closing VSAM or DAM files

You can close files with a CLOSE command, with or without the FORCE option.

Closing files normally

You can close a file explicitly with one of the following commands:

```
CEMT SET FILE(filename) CLOSED
EXEC CICS SET FILE(filename) CLOSED
```

The file is closed immediately if there are no transactions using the file at the time of the request. The file is also disabled as part of the close operation, this form of disablement showing as UNENABLED on a CEMT display. This prevents subsequent requests to access the file implicitly reopening it.

A transaction in the process of executing a VSAM or DAM request, or executing a series of connected requests, is said to be a user of the file. For example, a transaction is a user during the execution of the following EXEC CICS requests:

```
READ UPDATE ---- REWRITE
STARTBR--------- READNEXT ... ---- ENDBR
```

A transaction is also a user of a file if it completes a recoverable change to the file but has not yet reached a syncpoint or the end of the transaction.

If there are users at the time of the close request, the file is not closed immediately. CICS waits for all current users to complete their use of the file. The file is placed in an UNENABLING state to deny access to new users but allow existing users to complete their use of the file. When the last user has finished with the file, the file is closed and UNENABLED.

Closing files using the FORCE option

You can also close a file using one of the following commands:

```
CEMT SET FILE(filename) FORCECLOSE
EXEC CICS SET FILE(filename) CLOSED FORCE
```

Any transactions that are current users of the file are abended and allowed to back out any changes as necessary, and the file is then closed and UNENABLED. A file UNENABLED as a result of a CLOSE request can be reenabled subsequently if an explicit OPEN request is made.

The FORCE option and data integrity

Closing a file using the FORCE option causes tasks of any current users of the file to be terminated immediately by the CICS task FORCEPURGE mechanism. Data integrity is not guaranteed with this mechanism. In some extreme cases (for example, if an error occurs during backout processing), CICS might terminate abnormally. For this reason, closing files using the FORCE option should be restricted to exceptional circumstances.

XRF considerations

For both VSAM and DAM files:

- The active and alternate CICS region must refer to the same data sets. To ensure that they do, you can define the files using the DSNAME parameter, and allow CICS to allocate the data sets dynamically. Omitting DLBL statements in the job streams for the active and alternate CICS regions minimizes the risk of inconsistency in data set naming.
- The alternate CICS region does not open the data sets before takeover occurs.

CICS shared data tables

A data table is defined by means of the RDO FILE resource definition, or DFHFCT TYPE=CICSTABLE|USERTABLE macro. When a table is opened, CICS builds it by extracting data from the table’s corresponding source VSAM data set and loading it into a VSE data space owned by the CICS data tables 'server' region, and constructing an index in CICS virtual storage above the 16MB line. The commands used to access these tables are the file control commands of the CICS application programming interface (API).

For information about defining CICS shared data tables, see the *CICS Resource Definition Guide*. For programming information about the file control commands of the application programming interface, see the *CICS Application Programming Reference* manual.

CICS supports two types of data tables:

- **CICS-maintained data tables** that CICS keeps in synchronization with their source data sets.
- **User-maintained data tables** that are completely detached from their source data sets after being loaded.

For either type, a global user exit can be used to select which records from the source data set should be included in the data table.

For programming interface information about global user exits, see the *CICS Customization Guide*. For further information on CICS data tables, see the *CICS Shared Data Tables Guide*.

Opening data tables

A data table must be opened before its entries can be accessed by an application. You can open a data table explicitly with an OPEN request, implicitly on first reference, or by the CSFU task just after startup, if OPENTIME(STARTUP) was specified in the file definition. When a data table is opened, CICS reads the complete source data set, copying the records into a VSE data space and building an index.

A global user exit can be invoked for each record copied into the data table. This copying is subject to any selection criteria of the user-written exit.

The commands used to open data tables, and the rules and options concerning their implicit and immediate opening are the same as those described in “Opening VSAM or DAM files” on page 179.

Loading data tables

A data table is built automatically when it is opened. An index is constructed to provide rapid access to the records. See the *CICS Shared Data Tables Guide* for more details.

For a user-maintained data table, the ACB for the source data set is closed when loading has been completed. The data set is deallocated if it was originally dynamically allocated and there are no other ACBs open for it.

Closing data tables

You can close a data table with a CLOSE command, with or without the FORCE option. When a data table is closed, the data space storage that was used to hold the records and the address apace storage used for the associated index, is freed as part of the CLOSE operation.

The commands used to close data tables, and the rules concerning current users of a data table are the same as those described in “Closing VSAM or DAM files” on page 180.

XRF considerations

After an XRF takeover, a data table must be reloaded from its source data set when the data table is opened. For a CICS-maintained data table, the effect is to restore the data table to its final state in the previous active CICS region, because CICS keeps data tables and source data sets in step. For a user-maintained data table, the relationship of the current contents of the source data set to the contents of the data table when the previous active CICS region terminated is application-dependent.

Chapter 21. DL/I definitions

This chapter describes the definitions you need if your system accesses DL/I VSE databases. Information can be found under the following headings:

- “Definitions needed for DL/I support”
- “File control table (FCT)”
- “Application control table (ACT)” on page 184
- “XRF considerations” on page 184.

Definitions needed for DL/I support

If your CICS system accesses DL/I VSE databases, there are two sets of definitions that you must provide, and one that is optional. These are:

- A CICS file control table (FCT) containing DL/I entries
- An application control table (ACT)

In addition to these definitions, you must also specify the version of the dynamic backout program supplied for use with DL/I (it is DFHDBP2$). You do this by specifying DBP=2$ in the SIT, or as a startup override parameter, whenever you are running CICS with DL/I.

File control table (FCT)

If your CICS system is to access DL/I databases, you must code a DFHFCT TYPE=FILE macro for each DL/I database descriptor (DBD) that corresponds to a physical, logical, and index database. FILE, ACCMETH, FILSTAT and OPEN are the only operands of the DFHFCT TYPE=FILE macro that you code for a DL/I database.

Note

DL/I databases **cannot** be defined using RDO.

You must code the ACCMETH operand as DLI. The name you code on the DFHFCT TYPE=FILE macro must be the same as the NAME parameter in the DBD.

For more information about coding DFHFCT macros, see the *CICS Resource Definition Guide*.

You must provide a DLBL statement, either in the CICS startup job stream, or in a label information subarea, for each data set defined in a DBD that has a corresponding FCT entry. The file name you define in the DLBL statement should be the same name as the DD1 operand specified in the DBD. If the type of DBD is SHSAM, and there is a DD1 and DD2 parameter, code a DLBL statement for the DD1 name and the DD2 name. For example, the DLBL job control statement for a DL/I database might be:

```
// DLBL dd1name,'CICS410.applid.dlifile1',,VSAM,CAT=CICSUCT
```

Application control table (ACT)

The DL/I application control table (ACT) provides the logical connection between CICS application programs and the DL/I databases that they refer to. You generate an ACT, using DL/I DOS/VS, as a module named DLZNUC into a suitable sublibrary. You can specify a 2-character suffix for the ACT in the DLZACT TYPE=INITIAL macro, in which case your generated module is named DLZNUCxx, where xx is the suffix.

To initialize CICS with DL/I, you must specify DLI=YES, or DLI=xx, in the DFHSIT macro, or as an initialization override. If you specify DLI=YES, CICS loads module DLZNUC, the unsuffixed version of the ACT. If you code DLI=xx, CICS loads module DLZNUCxx, a suffixed version of the ACT (where xx is the suffix).

XRF considerations

In an XRF system, the active and alternate CICS systems must refer to the same DL/I database data sets.

Further information about DL/I is given in Chapter 6, “Accessing DL/I databases using CICS online applications” on page 67.

Part 3. CICS system initialization

This section of the book describes how to define CICS system initialization parameters, how they are processed by CICS, and gives a startup job stream you can use to start a CICS system.

Table 27. System initialization road map

If you want to...	**Refer to...**
Know about the system initialization process in general	Chapter 22, “CICS system initialization” on page 187
Know how to create a system initialization table	“Creating a system initialization table” on page 196
Know which parameters you cannot code in the DFHSIT macro	“Parameters that you cannot code in the DFHSIT macro” on page 198
Know how to override a system initialization table at startup	“Overriding SIT parameters at system startup” on page 198
Know about CICS resource table and module keywords	“System initialization control keywords” on page 200
Know how to select different versions of CICS programs and tables	“Selecting versions of CICS programs and tables” on page 207
Know about the different classes of start and restart	“Classes of start and restart” on page 209
Know about the individual CICS system initialization parameters.	“The system initialization parameter descriptions” on page 215

Chapter 22. CICS system initialization

This chapter describes the CICS system initialization process; describes the DFHSIT macro and its parameters, and tells you how to supply system initialization parameters to CICS.

Specifying system initialization parameters

The primary method of providing system initialization parameters is with a system initialization table (SIT). The parameters of the SIT, which you assemble as a load table, supply the system initialization program with most of the information necessary to initialize the system to suit your unique environment. You can generate more than one SIT, and (at the time of system initialization) select the one that is appropriate to your needs.

You can also specify other system initialization parameters, which cannot be coded in the SIT. You specify which SIT you want, and other system initialization parameters (with a few exceptions), using any of the following methods:

1. On the PARM parameter of the EXEC DFHSIP statement
2. In the SYSIPT data set defined in the startup job stream
3. Through the system operator’s console.

You can also use these methods to override most of the system initialization parameters assembled in the SIT.

The information defined by system initialization parameters can be grouped into three categories:

1. Information used to initialize and control CICS system functions (for example, information such as the dynamic storage area limits and the region exit time interval).
2. Module suffixes used to load the user-specified version of the CICS modules (for example, DFHDBPxx) and tables (for example, DFHJCTxx).
3. Special information used to control the initialization process.

The syntax of the various DFHSIT macro parameters is listed in Table 28 on page 188. Except for those parameters marked “**SIT macro only**,” all the system initialization parameters can be provided at run-time, although there are restrictions in some cases. These restrictions are explained at the end of the description of the system initialization parameter to which they apply.

There are some other CICS system initialization parameters (and options of the parameters in Table 28 on page 188) that you cannot define in the DFHSIT macro. (See “Parameters that you cannot code in the DFHSIT macro” on page 198.) The parameters that you cannot define in the DFHSIT macro are listed in “Parameters that you cannot code in the DFHSIT macro” on page 198.

System initialization keywords grouped by function

For a list of all the system initialization keywords grouped by their functional area, see Appendix A, "System initialization parameters grouped by functional area" on page 311. This ensures that you do not miss coding an important parameter relating to a particular CICS function. For details of how to code a parameter, you still have to refer to the parameter descriptions that are listed alphabetically in this chapter.

Migration considerations

If you have existing system initialization tables, you must modify them. Remove all obsolete parameters, and specify the required values for new or changed parameters if you want to run with other than the defaults. When you have made the necessary changes, reassemble the tables using the CICS Transaction Server for VSE/ESA Release 1 macro libraries.

If you have system initialization parameters defined in CICS start-up procedures, you must modify these also.

To avoid generating specific system initialization tables for each CICS system, a simple solution is to let CICS load the default, unsuffixed table (DFHSIT) at start-up, and supply the system initialization parameters for each CICS system in a SYSIPT data set. For more information about the source of the default system initialization table, see "DFHSIT, the default system initialization table" on page 192.

The DFHSIT macro parameters

Table 28 (Page 1 of 4). The DFHSIT macro parameters

```
        DFHSIT   [TYPE={CSECT|DSECT}]
                 [,ADI={30|number}]
                 [,AIEXIT={DFHZATDX|DFHZATDY|name}]
                 [,AILDELAY={0|hhmmss}]
                 [,AIQMAX={100|number}]
                 [,AIRDELAY={700|hhmmss}]
                 [,AKPFREQ={200|number}]
                 [,APPLID=({DBDCCICS|name1}[,name2])]
                 [,AUTCONN={0|hhmmss}]
                 [,AUXTR={OFF|ON}]
                 [,AUXTRSW={NO|ALL|NEXT}]
                 [,BMS=({MINIMUM|STANDARD|FULL}[,COLD]
                      [,{UNALIGN|ALIGN}][,{DDS|NODDS}])]
                 [,CLSDSTP={NOTIFY|NONOTIFY}]
                 [,CLT=xx]
                 [,CMDPROT={YES|NO}]
                 [CMDSEC={ASIS|ALWAYS}
                 [,CONFDATA={SHOW|HIDETC}]
                 [,CONFTXT={NO|YES}]
                 [,CSDACC={READWRITE|READONLY}]
                 [,CSDBUFND=number]
                 [,CSDBUFNI=number]
```

Table 28 (Page 2 of 4). The DFHSIT macro parameters

[,CSDFRLOG=number]
[,CSDJID={**NO**|number}]
[,CSDLSRNO={**1**|number|NONE|NO}]
[,CSDRECOV={**NONE**|ALL|BACKOUTONLY}]
[,CSDSTRNO={**2**|number}]
[,CWAKEY={**USER**|CICS}]
[,DATFORM={**MMDDYY**|DDMMYY|YYMMDD}]
,DBP={1$|2$|xx|YES} (Must specify in the SIT macro)
[,DBUFSZ={**500**|number}]
[,DCT=({**YES**|xx|NO}[,COLD])]
[,DFLTUSER={**CICSUSER**|userid}]
[,DIP={**NO**|YES}]
[,DISMACP={**YES**|NO}]
[,{DLI|DL1}=({**NO**|YES|REMOTE}[,COLD])]
[,DLIOER={**ABEND**|CONTINUE]]
[,DSALIM={**5M**|number}]
[,DSHIPIDL={**020000**|hhmmss}]
[,DSHIPINT={**120000**|hhmmss}]
[,DTRPGM={**DFHDYP**|program-name}]
[,DTRTRAN={**CRTX**|name|NO}]
[,DUMP={**YES**|NO}]
[,DUMPDS={**AUTO**|A|B}]
[,DUMPSW={**NO**|NEXT}]
[,EDSALIM={**20M**|number}]
[,EODI={**E0**|xx}]
[,ESMEXITS={**NOINSTLN**|INSTLN}] ***(SIT macro only)***
[,FCT={**YES**|xx|NO}]
[,FEPI={**NO**|YES}]
[,FLDSEP={**‘   ’**|‘xxxx’}]
[,FLDSTRT={**‘ ’**|‘x’}]
[,FSSTAFF={YES|**NO**}]
[,GMTEXT={**‘WELCOME TO CICS’**|‘text’}]
[,GMTRAN={**CSGM**|CESN|name}]
[,GNTRAN={**CESF**|transaction-id}.]
[,GRPLIST={**DFHLIST**|name|
 (name[,name2][,name3][,name4])}]
[,ICP=COLD]
[,ICV={**1000**|number}]
[,ICVR={**5000**|number}]
[,ICVTSD={**500**|number}]
[,INITPARM=(pgmname_1=‘parmstring_1’
      [, ... ,pgmname_n=‘parmstring_n’])]
[,INTTR={**ON**|OFF}]
[,IRCSTRT={**NO**|YES}]
[,ISC={**NO**|YES}]
[,JCT={**YES**|xx|NO}]
[,LGNMSG={**NO**|YES}]
[,MCT={**NO**|YES|xx}]
[,MN={**OFF**|ON}]
[,MNCONV={**NO**|YES}]
[,MNEXC={**OFF**|ON}]

Table 28 (Page 3 of 4). The DFHSIT macro parameters

		[,MNFREQ={**0**|hhmmss}]
		[,MNPER={**OFF**|ON}]
		[,MNSYNC={**NO**|YES}]
		[,MNTIME={**GMT**|LOCAL}]
		[,MROBTCH={**1**|number}]
		[,MROFSE={**NO**|YES}]
		[,MROLRM={**NO**|YES}]
		[,MSGCASE={**MIXED**|UPPER}]
		[,MSGLVL={**1**|0}]
		[,MXT={**5**|number}]
		[,NATLANG=(**E**,x,y,z,...)]
		[,OPERTIM={**120**|number}]
		[,PARMERR={**INTERACT**|IGNORE|ABEND}]
		[,PDI={**30**|number}]
		[,PGAICTLG={**MODIFY**|NONE|ALL}]
		[,PGAIEXIT={**DFHPGADX**|name}]
		[,PGAIPGM={**INACTIVE**|ACTIVE}]
		[,PGCHAIN=character(s)]
		[,PGCOPY=character(s)]
		[,PGPURGE=character(s)]
		[,PGRET=character(s)]
		[,PLTPI={**NO**|xx|YES}]
		[,PLTPISEC={**NONE**|CMDSEC|RESSEC|ALL}]
		[,PLTPIUSR=userid]
		[,PLTSD={**NO**|xx|YES}]
		[,PRGDLAY={**0**|hhmm}]
		[,PRINT={**NO**|YES|PA1|PA2|PA3}]
		[,PRTYAGE={**32768**|number}]
		[,PSDINT={**0**|hhmmss}]
		[,PVDELAY={**30**|number}]
		[,RAMAX={**256**|number}]
		[,RAPOOL={**50**|value1|(value1,value2)}]
		[,RENTPGM={**PROTECT**|NOPROTECT}]
		[,RESP={**FME**|RRN}]
		[,RESSEC={**ASIS**|ALWAYS}
		[,RMTRAN=({**CSGM**|name1}[,{**CSGM**|name2}])]
		[,RUWAPOOL={**NO**|YES}]
		[,SEC={**YES**|NO}]
		[,SECPRFX={**NO**|YES}]
		[,SKRxxxx='page-retrieval-command']
		[,SNSCOPE={**NONE**|CICS|VSEIMAGE}]
		[,SPCTR={(**1,2**|1[,2][,3])|ALL|OFF}]
		[,SPOOL={**NO**|YES}]
		[,SRT={**YES**|xx|NO}]
		[,START={**AUTO**|COLD|STANDBY}]
		[,STARTER={**NO**|YES}] ***(SIT macro only)***
		[,STATRCD={**OFF**|ON}]
		[,STGPROT={**NO**|YES}]
		[,STGRCVY={**NO**|YES}]
		[,STNTR={**1**|(1[,2][,3])|ALL|OFF}]
		[,SUFFIX=xx] ***(SIT macro only)***

Table 28 (Page 4 of 4). The DFHSIT macro parameters

		[,SYDUMAX={**999**|number}]
		[,SVA={**NO**|YES}]
		[,SYSIDNT={**CICS**|name}]
		[,SYSTR={**ON**|OFF}]
		[,TAKEOVR={**MANUAL**|AUTO|COMMAND}]
		[,TBEXITS=([nm1][,nm2][,nm3][,nm4][,nm5])]
		[,TCP={**YES**|NO}]
		[,TCSACTN={**NONE**|UNBIND|FORCE}]
		[,TCSWAIT={**4**|number|NO|NONE|0}]
		[,TCT={**YES**|xx|NO}]
		[,TCTUAKEY={**USER**|CICS}]
		[,TCTUALOC={**BELOW**|ANY}]
		[,TD=({**3**|number1}[,{**3**|number2}])]
		[,TRAP={**OFF**|ON}]
		[,TRDUMAX={**999**|number}]
		[,TRTABSZ={**16**|number}]
		[,TRTRANSZ={**40**|number}]
		[,TRTRANTY={**TRAN**|ALL}]
		[,TS=([COLD][,{0|**3**|value-1}][,{**3**|value-2}])]
		[,TSMGSET={**4**|number}]
		[,TST={**NO**|YES|xx}]
		[,USERTR={**ON**|OFF}]
		[,USRDELAY={**30**|number}]
		[,VTAM={**YES**|NO}]
		[,VTPREFIX={\|character}]
		[,WRKAREA={**512**|number}]
		[,XAPPC={**NO**|YES}]
		[,XCMD={**NO**|name|YES}]
		[,XDCT={**NO**|name|YES}]
		[,XFCT={**NO**|name|YES}]
		[,XJCT={**NO**|name|YES}]
		[,XLT={**NO**|xx|YES}]
		[,XPCT={**NO**|name|YES}]
		[,XPPT={**NO**|name|YES}]
		[,XPSB={**NO**|name|YES}]
		[,XRF={**NO**|YES}]
		[,XRFSOFF={**NOFORCE**|FORCE}]
		[,XRFSTME={**5**|number}]
		[,XRFTODI={**30**|number}]
		[,XSWITCH=([{**0**|1-254}][{,**????????**|progname}][{,**A**|B}])]
		[,XTRAN={**YES**|name|NO}]
		[,XTST={**NO**|name|YES}]
		[,XUSER={**NO**|YES}]
		You must terminate your macro parameters with the following END statement.
	END	DFHSITBA

DFHSIT, the default system initialization table

The macro source statements used to assemble the default system initialization table are given in Figure 57.

This default example SIT is named DFHSIT$$. The source may be found in the VSE/ESA sublibrary, PRD1.BASE.

```
* SIT parameters (in alphabetical order)                               *
*                                                                      *
SIT      TITLE 'DFHSIT - CICS DEFAULT SYSTEM INITIALIZATION TABLE'
         DFHSIT TYPE=CSECT,                                            *
               ADI=30,           XRF(B) - Alternate delay interval     *
               AIEXIT=DFHZATDX,  Autoinstall user program name         *
               AILDELAY=0,       Delete delay period for AI TCTTEs     *
               AIQMAX=100,       Maximum no. of terminals queued for AI*
               AIRDELAY=700,     Restart delay period for AI TCTTEs    *
               AKPFREQ=200,      Activity keypoint frequency           *
               APPLID=DBDCCICS,  VTAM APPL identifier                  *
               AUTCONN=0,        Autoconnect delay                     *
               AUXTR=OFF,        Auxiliary trace option                *
               AUXTRSW=NO,       Auxiliary trace autoswitch facility   *
               BMS=(FULL,,UNALIGN,DDS), Basic Mapping Support options  *
               CLSDSTP=NOTIFY,   Notification for ISSUE PASS command   *
               CLT=,             The command list table option/suffix  *
               CMDPROT=YES,      Exec storage command checking         *
               CMDSEC=ASIS,      API command security checking         *
               CONFDATA=SHOW,    Confidential CICS data                *
               CONFTXT=NO,       Confidential VTAM data                *
               CSDACC=READWRITE, CSD access                            *
               CSDBUFND=,        Number of data buffers for the CSD    *
               CSDBUFNI=,        Number of index buffers for the CSD   *
               CSDFRLOG=NO,      Journal id. for CSD forward recovery  *
               CSDJID=NO,        Journal id. for CSD auto. journaling  *
               CSDLSRNO=1,       The VSAM LSR pool number for the CSD  *
               CSDRECOV=NONE,    CSD recoverable file option           *
               CSDSTRNO=2,       CSD Number of strings                 *
               CWAKEY=USER,      CWA storage key                       *
               DATFORM=MMDDYY,   CSA date format                       *
               DBP=1$,           Required version of DBP with DLI=NO   *
               DBUFSZ=500,       Dynamic backout buffer size           *
               DCT=YES,          Dest. control table option/suffix     *
               DFLTUSER=CICSUSER, Default user                         *
               DIP=NO,           Batch data interchange program        *
```

Figure 57 (Part 1 of 5). DFHSIT, the pregenerated default system initialization table

```
         DISMACP=YES,      Disable macro programs                 *
         DLI=NO,           DL/1 option                            *
         DLIOER=ABEND,     Abend if DL/I error                    *
         DSALIM=5M,        Upper limit of DSA below 16Mb line     *
         DSHIPIDL=020000,  Delete shipped idle time               *
         DSHIPINT=120000,  Delete shipped interval                *
         DTRPGM=DFHDYP,    Dynamic transaction routing program    *
         DTRTRAN=CRTX,     Default dynamic tran routing transid   *
         DUMP=YES,         Dump option                            *
         DUMPDS=AUTO,      CICS dump data set opening option      *
         DUMPSW=NO,        Dump data set autoswitch option        *
         EDSALIM=20M,      Upper limit of DSA above 16MB line     *
         EODI=E0,          End-of-data indicator for seq. devices*
         ESMEXITS=NOINSTLN, External security manager exits       *
         FCT=YES,          File control table option/suffix       *
         FEPI=NO,          Front-End Programming Interface        *
         FLDSEP='    ',    End-of-field separator characters      *
         FLDSTRT=' ',      Field start character for builtin fn   *
         FSSTAFF=NO,       Function-shipped START affinity option*
         GMTEXT='WELCOME TO CICS', Good-morning message text      *
         GMTRAN=CSGM,      Initial transaction                    *
         GNTRAN=NO,        Signoff transaction                    *
         GRPLIST=DFHLIST,  List name of CSD groups for startup    *
         ICP=,             Interval control pgm. start option     *
         ICV=1000,         Region exit interval (milliseconds)    *
         ICVR=5000,        Runaway task interval (milliseconds)   *
         ICVTSD=500,       Terminal scan delay interval (  "  )   *
         INITPARM=,        Initialization parms for programs      *
         INTTR=ON,         CICS internal trace option             *
         IRCSTRT=NO,       Interregion communication start        *
         ISC=NO,           Intersystem communication option       *
         JCT=YES,          Journal control table option/suffix    *
         LGNMSG=NO,        Extract VTAM logon data                *
         MCT=NO,           Monitoring cntl.table option/suffix    *
         MN=OFF,           CICS monitoring option                 *
         MNCONV=NO,        Monitoring converse recording option   *
         MNEXC=OFF,        Monitoring exception class option      *
         MNFREQ=0,         Monitoring frequency period            *
         MNPER=OFF,        Monitoring performance class option    *
         MNSYNC=NO,        Monitoring syncpoint recording option  *
         MNTIME=GMT,       Monitoring timestamp (GMT/LOCAL)       *
         MROBTCH=1,        Number of MRO requests to batch        *
         MROFSE=NO,        Extend lifetime of Long-running mirror*
         MROLRM=NO,        Long-running mirror task option        *
         MSGCASE=MIXED,    CICS messages in mixed case            *
         MSGLVL=1,         System console MSG level option        *
         MXT=5,            Maximum number of tasks in CICS        *
         NATLANG=E,        List of national languages             *
         OPERTIM=120,      Write to operator timeout (seconds)    *
```

Figure 57 (Part 2 of 5). DFHSIT, the pregenerated default system initialization table

```
                PARMERR=INTERACT, System init. parameter errors option  *
                PDI=30,           Primary delay interval - XRF active   *
                PGAICTLG=MODIFY,  PG autoinstall catalog state          *
                PGAIEXIT=DFHPGADX, PG autoinstall exit program          *
                PGAIPGM=INACTIVE, PG autoinstall state                  *
                PGCHAIN=,         BMS CHAIN command                     *
                PGCOPY=,          BMS COPY command                      *
                PGPURGE=,         BMS PURGE command                     *
                PGRET=,           BMS RETURN command                    *
                PLTPI=NO,         Program list table PI option/suffix   *
                PLTPISEC=NONE,    No PLT security checks on PI programs *
                PLTPIUSR=,        PLT PI userid = CICS system userid    *
                PLTSD=NO,         Program list table SD option/suffix   *
                PRGDLAY=0,        BMS purge delay interval              *
                PRINT=NO,         Print key option                      *
                PRTYAGE=32768,    Dispatcher priority ageing value      *
                PSDINT=0,         Persistent Session Delay Interval     *
                PVDELAY=30,       Timeout value for LUIT Table          *
                RAMAX=256,        Max. I/O area for RECEIVE ANY         *
                RAPOOL=50,        Max. RECEIVE ANY Request Parm.Lists   *
                RENTPGM=PROTECT,  Reentrant program write protection    *
                RESP=FME,         Logical unit response type            *
                RESSEC=ASIS,      Resource security check               *
                RMTRAN=CSGM,      VTAM persistent session recovery tran *
                RUWAPOOL=NO,      LE RUWA storage pool option           *
                SEC=YES,          External security manager option      *
                SECPRFX=NO,       Security prefix                       *
                SKRPA1=,          SKR PA1 PAGE RETRIEVAL CMD            *
                SKRPA2=,          SKR PA2 PAGE RETRIEVAL CMD            *
                SKRPA3=,          SKR PA3 PAGE RETRIEVAL CMD            *
                SKRPF1=,          SKR PF1 PAGE RETRIEVAL CMD            *
                SKRPF2=,          SKR PF2 PAGE RETRIEVAL CMD            *
                SKRPF3=,          SKR PF3 PAGE RETRIEVAL CMD            *
                SKRPF4=,          SKR PF4 PAGE RETRIEVAL CMD            *
                SKRPF5=,          SKR PF5 PAGE RETRIEVAL CMD            *
                SKRPF6=,          SKR PF6 PAGE RETRIEVAL CMD            *
                SKRPF7=,          SKR PF7 PAGE RETRIEVAL CMD            *
                SKRPF8=,          SKR PF8 PAGE RETRIEVAL CMD            *
                SKRPF9=,          SKR PF9 PAGE RETRIEVAL CMD            *
                SKRPF10=,         SKR PF10 PAGE RETRIEVAL CMD           *
                SKRPF11=,         SKR PF11 PAGE RETRIEVAL CMD           *
                SKRPF12=,         SKR PF12 PAGE RETRIEVAL CMD           *
                SKRPF13=,         SKR PF13 PAGE RETRIEVAL CMD           *
                SKRPF14=,         SKR PF14 PAGE RETRIEVAL CMD           *
                SKRPF15=,         SKR PF15 PAGE RETRIEVAL CMD           *
                SKRPF16=,         SKR PF16 PAGE RETRIEVAL CMD           *
                SKRPF17=,         SKR PF17 PAGE RETRIEVAL CMD           *
                SKRPF18=,         SKR PF18 PAGE RETRIEVAL CMD           *
                SKRPF19=,         SKR PF19 PAGE RETRIEVAL CMD           *
```

Figure 57 (Part 3 of 5). DFHSIT, the pregenerated default system initialization table

```
         SKRPF20=,         SKR PF20 PAGE RETRIEVAL CMD           *
         SKRPF21=,         SKR PF21 PAGE RETRIEVAL CMD           *
         SKRPF22=,         SKR PF22 PAGE RETRIEVAL CMD           *
         SKRPF23=,         SKR PF23 PAGE RETRIEVAL CMD           *
         SKRPF24=,         SKR PF24 PAGE RETRIEVAL CMD           *
         SKRPF25=,         SKR PF25 PAGE RETRIEVAL CMD           *
         SKRPF26=,         SKR PF26 PAGE RETRIEVAL CMD           *
         SKRPF27=,         SKR PF27 PAGE RETRIEVAL CMD           *
         SKRPF28=,         SKR PF28 PAGE RETRIEVAL CMD           *
         SKRPF29=,         SKR PF29 PAGE RETRIEVAL CMD           *
         SKRPF30=,         SKR PF30 PAGE RETRIEVAL CMD           *
         SKRPF31=,         SKR PF31 PAGE RETRIEVAL CMD           *
         SKRPF32=,         SKR PF32 PAGE RETRIEVAL CMD           *
         SKRPF33=,         SKR PF33 PAGE RETRIEVAL CMD           *
         SKRPF34=,         SKR PF34 PAGE RETRIEVAL CMD           *
         SKRPF35=,         SKR PF35 PAGE RETRIEVAL CMD           *
         SKRPF36=,         SKR PF36 PAGE RETRIEVAL CMD           *
         SNSCOPE=NONE,     Multiple CICS sessions per userid     *
         SPCTR=(1,2),      Level(s) of special tracing required  *
         SPOOL=NO,         System spooling interface option      *
         SRT=YES,          System recovery table option/suffix   *
         START=AUTO,       CICS system initialization option     *
         STARTER=YES,      Starter ($ and #) suffixes option     *
         STATRCD=OFF,      statistics recording status           *
         STGPROT=NO,       Storage protection facility           *
         STGRCVY=NO,       Storage recovery option               *
         STNTR=1,          Level of standard tracing required    *
         SUFFIX=$$,        Suffix of this SIT                    *
         SVA=NO,           Use SVA option for CICS/user modules  *
         SYDUMAX=999,      No of SYSDUMPS to be taken            *
         SYSIDNT=CICS,     Local system identifier               *
         SYSTR=ON,         Master system trace flag              *
         TAKEOVR=MANUAL,   XRF alternate takeover option         *
         TBEXITS=,         Transaction backout exit programs     *
         TCP=YES,          Terminal control program option/suffix*
         TCSACTN=NONE,     TC Shutdown action                    *
         TCSWAIT=4,        TC Shutdown wait                      *
         TCT=YES,          Terminal control table option/suffix  *
         TCTUAKEY=USER,    TCT user area storage key             *
         TCTUALOC=BELOW,   TCT user area below 16MB              *
         TD=(3,3),         Transient data buffers and strings    *
         TRAP=OFF,         F.E. global trap exit option          *
         TRDUMAX=999,      No of TRANDUMPS to be taken           *
         TRTABSZ=16,       Internal trace table size in 1K bytes *
         TRTRANSZ=40,      Transaction dump trace table size     *
         TRTRANTY=TRAN,    Transaction dump trace option         *
         TS=(,3,3),        Temporary storage buffers and strings *
         TSMGSET=4,        # of entries for pointers to TS MSGset*
         TST=NO,           Temporary storage table option/suffix *
         USERTR=ON,        Master user trace flag                *
         USRDELAY=30,      Timeout value for User Dir. Entries   *
```

Figure 57 (Part 4 of 5). DFHSIT, the pregenerated default system initialization table

```
         VTAM=YES,          VTAM access method option           *
         VTPREFIX=\         Client virtual terminal prefix      *
         WRKAREA=512,       Common work area (CWA) size in bytes  *
         XAPPC=NO,          ESM class APPCLU required           *
         XCMD=NO,           SPI don't perform ESM check         *
         XDCT=NO,           DCT don't perform ESM check         *
         XFCT=NO,           FCT don't perform ESM check         *
         XJCT=NO,           JCT don't perform ESM check         *
         XLT=NO,            Transaction list table option/suffix  *
         XPCT=NO,           PCT use default name for ESM  check   *
         XPPT=NO,           PPT don't perform ESM check         *
         XPSB=NO,           PSB don't perform ESM check         *
         XRF=NO,            Extended recovery feature (XRF) option*
         XRFSOFF=NOFORCE,   XRF -  Re-sign on after takeover     *
         XRFSTME=5,         XRF -  sign off timeout value        *
         XRFTODI=30,        XRF takeover delay inteval          *
         XSWITCH=(0,????????,A), Prgmable terminal switching unit*
         XTRAN=YES,         Transid use default name, ESM check   *
         XTST=NO,           TST don't perform ESM check         *
         XUSER=NO           No surrogate user checking
   END   DFHSITBA
```

Figure 57 (Part 5 of 5). DFHSIT, the pregenerated default system initialization table

Creating a system initialization table

Decide which parameters you want to use. List them, together with the appropriate setting, in a file named DFHSIT*xx*. The suffix *xx* can be any two characters that you want (apart from the characters reserved solely for use by CICS), and identifies the file as your own SIT. If you do not specify a two-character suffix, CICS uses the default system initialization table, DFHSIT1$.

See Appendix A, “System initialization parameters grouped by functional area” on page 311 which summarizes the groups of system initialization parameters needed for different CICS functions.

If you specify more than one value for a system initialization parameter, separate the operand values by commas and enclose them all in parentheses. If you do not specify a parameter as a system initialization override, the system uses the value supplied in whichever SIT you specify.

End your system initialization table with an

```
END DFHSITBA
```

statement.

“The system initialization parameter descriptions” on page 215 describes all the system initialization parameters on an individual basis.

There are some parameters that cannot be coded in the DFHSIT macro. These are summarized in “Parameters that you cannot code in the DFHSIT macro” on page 198.

Coding more than one system initialization table

You can code several SITs, each reflecting different parameter combinations and settings, as long as you remember to give each DFHSIT*xx* a different two-character suffix. CICS uses only one of these tables during initialization. Choose the SIT that best suits your needs at the time, and code its suffix on the SIT system initialization parameter.

Assemble your chosen SIT and link-edit it into one of the sublibraries defined in the LIBDEF statement for your CICS startup job stream.

CICS-supplied system initialization tables

The VSE/ESA sublibrary, PRD1.BASE, contains the sample system initialization tables, DFHSIT6$ and DFHSIT$$. DFHSIT6$ contains all the SIT values needed for a standard CICS system. DFHSIT$$ contains all the default values for the system initialization parameters.

System initialization tables supplied with VSE/ESA

VSE/ESA supplies two sample system initialization tables. DFHSITSP is provided for a standard CICS system, and DFHSITC2 is provided for a second predefined CICS system. There are listings for both these tables in the *VSE/ESA Planning* manual.

Assembling the system initialization table

When you have defined the parameters you need, add your system initialization table to your working CICS system by assembling the DFHSIT macro. The macro assembly process produces the linkage-editor control statements needed to link-edit your table into your CICS (or private) sublibrary.

When you have assembled your SIT, make it known to CICS using one of the following methods:

- Including your sublibrary in the LIBDEF statement of the CICS startup job stream.
- Coding the suffix of your system initialization table on the SIT system initialization override parameter.
- Including the suffix of chosen system initialization table on the PARM parameter of the EXEC DFHSIP statement. Figure 58 on page 203 gives an example of using the PARM parameter to specify a SIT with the suffix 1$.

For information about assembling and link-editing CICS control tables, see Chapter 2, “Defining resources in CICS control tables” on page 7.

Examples of the syntax notation for describing CICS macros are given in the *CICS Resource Definition Guide*.

Assembler errors from duplicated or undefined keywords

If you duplicate a keyword when using VSE/ESA 2.4 and the High Level Assembler, the following error messages are produced:

```
ASMA018S *** ERROR *** DUPLICATE KEYWORD IN MACRO CALL;
                       LAST VALUE IS USED - DFHSI/keyword
```

If you use an undefined keyword, the following error message is produced:

```
ASMA017W ** WARNING ** UNDEFINED KEYWORD PARAMETER; DEFAULT TO POSITIONAL,
INCLUDING KEYWORD - DFHSI/keyword
```

However, be aware that because of work space limitations, there is a limit to the number of undefined keyword errors that the assembler can generate. This means that if your SIT contains more undefined keywords than the assembler can generate messages for, the excess errors will not be flagged until a second or even later assembly and, as you correct the currently flagged errors, other (previously unflagged) errors may appear during re-assembly.

Parameters that you cannot code in the DFHSIT macro

There are some CICS system initialization parameters that you cannot define in the DFHSIT macro. These are:

- CDSASZE={<u>0K</u>|number}
- CHKSTRM={CURRENT|NONE}
- CHKSTSK={ALL|CURRENT|NONE}
- ECDSASZE={<u>0K</u>|number}
- ERDSASZE={<u>0K</u>|number}
- ESDSASZE={<u>0K</u>|number}
- EUDSASZE={<u>0K</u>|number}
- JSTATUS=RESET
- NEWSIT={YES|<u>NO</u>}
- PRVMOD={name|(name,name...name)}
- RDSASZE={<u>0K</u>|number}
- SDSASZE={<u>0K</u>|number}
- SIT=xx
- SPCTRxx={(1[,2][,3])|ALL|OFF}
- START=LOGTERM
- START=(option,ALL)
- STNTRxx={(1[,2][,3])|ALL|OFF}
- UDSASZE={<u>0K</u>|number}

These parameters can only be supplied at CICS startup time using any of the following methods:

1. The PARM parameter of the EXEC DFHSIP statement
2. The SYSIPT data set defined in the startup job stream
3. Through the system operator's console.

For information about coding system initialization parameters in PARM, SYSIPT or at the console, see "System initialization control keywords" on page 200.

Overriding SIT parameters at system startup

There may be times when you need to change some of the parameters specified in your SIT, but don't want to have to modify your SIT and reassemble it; or you may need to supply parameters that cannot be coded in the DFHSIT macro (as listed in "Parameters that you cannot code in the DFHSIT macro" on page 198). To do this you supply system initialization parameters as **overrides** using the:

- PARM parameter on the EXEC DFHSIP statement

- SYSIPT data set
- Operator console.

You can use one, or a combination of the above methods. However, parameter manager domain processes these three sources in the following strict sequence:

1. The PARM parameter
2. The SYSIPT data set (but only if SYSIPT is coded in the PARM parameter
3. The console (but only if CONSOLE is coded in either the PARM parameter or in the SYSIPT data set.

Note: If you supply duplicate system initialization parameters, either through the same or a different medium, CICS takes the **last** one that it reads. For example, if you specify DCT=1$ in the PARM parameter, DCT=2$ in the SYSIPT data set, and finally enter DCT=3$ through the console, CICS loads DFHDCT3$.

You complete the override process with the $END statement.

During startup, CICS uses your chosen system initialization table and any system initialization parameter overrides you have specified to determine the CICS functions you require.

The CICS parameter manager domain

In addition to loading the system initialization table at the start of initialization, and reading any other parameters from PARM, SYSIPT, or the system console, the parameter manager domain is responsible for the management of the SIT. With the exception of the application domain (AP) which uses the SIT directly, the parameter manager domain passes system initialization parameters to the other CICS domains on request. The domain initialization process is as follows:

Query the type of startup
With the exception of the trace domain, each domain asks the parameter manager for the type of startup, cold or warm. (For this purpose, the parameter manager domain treats an emergency restart as a warm start.)

Startup is cold
If startup is cold, domains do not read their domain records from the catalogs. Instead, they request system initialization parameters from the parameter manager domain. Because it is a cold start, the parameter manager domain returns all system initialization parameters from the SIT, modified by any overrides.

Startup is warm
If startup is warm, domains try to perform a warm start by reading their domain records from the catalogs:

- If they succeed in reading their status records, domains perform a warm start. Where applicable, they also request system initialization parameters from the parameter manager domain. Because it is a warm start, the parameter manager domain returns only those system initialization parameters supplied as overrides via PARM, SYSIPT, or the system console.
- If they fail for any reason to read their status records, domains perform a cold start. They do this either by requesting ***all*** system initialization parameters from the parameter manager domain, or by using system

default values if the domain does not have any system initialization parameters.

NEWSIT or new suffix

Although a START=AUTO may resolve to a warm start, parameter manager enforces most system initialization parameters if:

- You specify NEWSIT=YES as a system initialization parameter in PARM, SYSIPT, or through the console.
- You specify a different SIT suffix from the previous run of CICS. Parameter manager saves the suffix from each run in the global catalog, and, if it detects a new suffix, it forces the NEWSIT=YES option.

For details of the parameters that are ignored when you specify NEWSIT=YES, see the NEWSIT parameter description on page 249.

Note: The trace domain is an exception to the above rules in that it always cold starts. Trace does not save its status at CICS shutdown like the other domains, and regardless of the type of startup, it requests ***all*** of its system initialization parameters from the parameter manager domain.

System initialization control keywords

There are three special control keywords that you can use at start up to control how CICS reads in system initialization parameters. These are:

- SYSIN (abbreviated to SI)
- CONSOLE (abbreviated to CN)
- .END

The SYSIN (SI) keyword

Code the SYSIN keyword to tell CICS to read system initialization parameters from the SYSIPT data set. You can code SYSIN (or SI) only on the PARM parameter of the EXEC DFHSIP statement. The keyword can appear once only and must be at the end of the PARM parameter. CICS does not read the SYSIPT data stream until it has finished scanning all of the PARM parameter, or until it reaches an $END keyword before the end of the PARM parameter. An example of coding the SYSIN keyword is:

```
// EXEC DFHSIP,SIZE=DFHSIP,PARM='SI',OS390
```

The CONSOLE (CN) keyword

Use the CONSOLE (CN) keyword to tell CICS to read initialization parameters from the console. CICS prompts you with message DFHPA1104 when it is ready to read parameters from the console.

You can code the CONSOLE (or CN) keyword in the PARM parameter of the EXEC DFHSIP statement, or in the SYSIPT data stream. The keyword can appear anywhere in the PARM parameter or SYSIPT data stream, but you must code it in one place only.

If you code CONSOLE on the PARM parameter and the PARM parameter also contains the SYSIN keyword CICS does not begin reading parameters from the console until it has finished reading and processing the SYSIPT data stream. Similarly, if you code CONSOLE in the SYSIPT data stream, CICS does not begin reading parameters from the console until it has finished reading and processing the SYSIPT data stream.

In general, after a given CICS system has become the production system, you should perform startup with as little operator intervention as possible (that is, use PARM or SYSIPT, and ***not*** the console). Better still, use the SIT only, without any overrides. An example of coding the CONSOLE keyword is as follows:

```
// EXEC DFHSIP,SIZE=DFHSIP,PARM='CN',OS390
```

The .END keyword

The meaning of the .END keyword varies, depending on its context.

Note that $END is also recognized for compatibility with earlier versions of CICS.

If you are using the PARM parameter, the use of the .END keyword is optional. If you omit it, CICS assumes it to be at the end of the PARM parameter. If you code .END in the PARM parameter it can have one of two meanings:

1. If you also code one, or both, of the other control keywords (CONSOLE and/or SYSIPT) .END denotes the end of the PARM parameter only. For example:

   ```
   // EXEC DFHSIP,SIZE=DFHSIP,PARM='SI,CN,SIT=6$,.END',OS390
   ```

2. If you code .END as the only control keyword in the PARM parameter, it denotes the end of all system initialization parameters, and CICS begins the initialization process. For example:

   ```
   // EXEC DFHSIP,SIZE=DFHSIP,PARM='SIT=6$,.END',OS390
   ```

If .END is not the last entry in the PARM parameter, CICS truncates the PARM parameter and the parameters following the .END keyword are ignored.

If you are using the SYSIPT data stream, the use of the .END keyword is optional. If you omit it, CICS assumes it to be at the end of SYSIPT. If you code .END in the SYSIPT data stream its meaning depends on your use of the CONSOLE keyword, as follows:

- If you code the CONSOLE control keyword in the PARM parameter or in the SYSIPT data stream, .END denotes the end of the SYSIPT data stream only.
- If you do not code the CONSOLE control keyword in the PARM parameter or in the SYSIPT data stream, .END denotes the end of all CICS system initialization parameters, and CICS begins the initialization process.

If you code .END, and it is not the last entry in the SYSIPT data stream, or not at the end of a SYSIPT record, CICS initialization parameters following the .END are ignored. To avoid accidental loss of initialization parameters, ensure that the .END keyword is on the last record in the SYSIPT data stream, and that it is the only entry on that line. (However, if you want to remove some system initialization parameters from a particular run of CICS, you could position them after the .END statement just for that run.)

The following example shows the use of .END in a SYSIPT data set:

```
// EXEC DFHSIP,SIZE=DFHSIP,PARM='SI',OS390
* CICS system initialization parameters
SIT=6$,START=COLD,
AUXTR=OFF,CSDLSRNO=15
  ⋮
.END
/*
```

If you are using the CONSOLE keyword, the meaning of .END through the console depends on whether you are entering new parameters or entering corrections. The two meanings are as follows:

1. If you are keying new parameters in response to message DFHPA1104, .END terminates parameter reading, and CICS starts initialization according to the SIT it has loaded, but modified by any system initialization parameters you have supplied. Until you enter the .END control keyword, CICS continues to prompt you for system initialization parameters.
2. If you have coded PARMERR=INTERACT, and CICS detects a parameter error, either in the keyword or in the value that you have assigned to it, CICS prompts you to correct the error with message DFHPA1912 or DFHPA1915. If you enter the correct keyword or value, initialization resumes with CICS continuing to process either the PARM parameter, the SYSIPT data set, or prompting for more system initialization parameters through the console, depending on where CICS detected the error. If you cannot correct the error, but want CICS to continue with the initialization process, you can enter .END to end the error correction phase.

Processing the PARM parameter

If you omit the PARM parameter from the EXEC DFHSIP statement CICS assumes that there are no SIT override or other initialization parameters, and attempts to load an unsuffixed version of DFHSIT. As a general rule, this is unlikely to be your intention, so the PARM parameter should at least specify the suffix of your system initialization table, using the SIT system initialization parameter.

Alternatively, you can code the special SYSIN keyword as the only PARM parameter, and supply the suffix of your SIT and the other system initialization parameters from the SYSIPT data set.

CICS scans the PARM string looking for a SIT= parameter, any of the special control keywords, or any system initialization parameters, and proceeds as follows:

- If CICS finds a SIT= parameter but no SYSIN keyword, CICS tries to load the SIT as soon as it has finished scanning the PARM parameter. Processing any CICS system initialization parameters that are also present in the PARM parameter takes place only after the SIT has been loaded.
- If CICS finds a SIT= parameter **and** a SYSIN keyword, CICS does not try to load the SIT until it has also finished scanning the SYSIPT data set. In this case, loading the SIT is deferred because there can be other SIT= parameters coded in the SYSIPT data set that override the one in the PARM parameter.

  Processing any system initialization parameters that are also present in the PARM parameter takes place only after the SIT has been loaded.

Rules for coding PARM parameters

The rules for coding the PARM parameter on an EXEC job control statement are described fully in the *VSE/ESA System Control Statements* manual.

Briefly, the maximum number of characters you can code on each PARM parameter is 100, excluding the opening and closing apostrophes. All CICS system initialization parameters must be separated by a comma, and the separating commas are included in the 100 character limit. You can code the PARM parameter up to three times on one EXEC statement. The syntax rules above apply to each PARM parameter separately. Because of this limiting factor, you might prefer to limit the use of the PARM parameter to specify the SYSIN control keyword only.

Coding the PARM parameter over two lines

If you need to continue your PARM parameter on a second line, indicate the continuation with a nonblank character in column 72, and continue in column 16 on the next line. Do not leave any blank columns between your SIT overrides and the continuation character in column 72; your overrides must be entered up to, and including, column 71.

Figure 58 gives an example of the PARM parameter coded over two lines.

```
0----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
 // EXEC DFHSIP,SIZE=DFHSIP,PARM='SIT=1$,DCT=1B,GRPLIST=DFHLIST,TCT=5$,T*
               S=(,0),SPOOL=YES,START=AUTO',OS390
```

Figure 58. Example of a PARM parameter coded over two lines

Processing the SYSIPT data set

CICS scans the SYSIPT data set looking for a SIT= parameter and any of the special keywords, as well as system initialization parameters.

If CICS finds a SIT= parameter in the SYSIPT data set, it tries to load that SIT, overriding any that was specified in the PARM parameter. If CICS does not find a SIT= parameter in the SYSIPT data set, it tries to load any SIT specified in the PARM parameter.

However, if after scanning the PARM parameter and the SYSIPT data set CICS has not found a SIT= parameter, CICS does one of the following:

1. If you specified CONSOLE in the PARM parameter or in the SYSIPT data set, CICS prompts you with the following message to enter the SIT suffix as the first parameter through the console:

   ```
   DFHPA1921  DBDCCICS  PLEASE SPECIFY THE REQUIRED SIT SUFFIX, OR
                        SPECIFY 'NONE'(UNSUFFIXED).
   ```

2. If you did not specify CONSOLE, CICS automatically tries to load an unsuffixed SIT module (DFHSIT). If this load fails, CICS issues message DFHPA1106, requesting a SIT suffix in reply to the message.

Note: CICS does not process any system initialization parameters that are coded in the PARM parameter and the SYSIPT data set until after the SIT has been loaded.

Rules for coding CICS system initialization parameters in the SYSIPT data set

There are a few simple rules to observe when coding CICS system initialization parameters in the SYSIPT data set. These are:

- Use a comma to separate parameters that are on the same line. The use of a comma at the end of a SYSIPT record is optional.
- You can use an asterisk in column 1 to code comments, or to remove temporarily an initialization parameter from a particular execution of CICS.
- You can also add comments after the parameters on a SYSIPT line, but they must be preceded by at least one blank character.
- The SYSIPT data set is an 80-byte file. Everything that appears in positions 1 through 80 is treated by CICS as input data.
- You can continue, on another line in SYSIPT, parameters that have multiple operands if you make the split immediately after a comma. CICS concatenates the operands, omitting the intervening blanks.
- As a general rule, you cannot split an individual operand between lines. However, in the case of the GMTEXT parameter, you can enter the operand over more than one line up to the maximum of 246 characters. The format of this parameter is:

  ```
  GMTEXT='User''s text'
  ```

  You can use apostrophes to punctuate message text, provided that you code *two* successive apostrophes to represent a single apostrophe (as shown in the example above). The apostrophes delimiting the text are mandatory.
- You must take care when coding parameters that use apostrophes, parentheses, or commas as delimiters, because failure to include the correct delimiters is likely to cause unpredictable results.

Processing the console entries

Generally, CICS does not begin to read from the console until it has loaded the SIT and processed any initialization parameters that are coded in the PARM parameter and the SYSIPT data set. CICS accepts system initialization parameters from the console until you terminate the input with .END.

You can specify a SIT= parameter only as the first parameter through the console when prompted by message DFHPA1921, at which point CICS tries to load the specified SIT. If you try to specify a SIT= parameter after CICS has loaded the SIT it is rejected as an error.

Rules for coding CICS system initialization parameters at the console

When it is ready to read parameters from the console, CICS displays the following message (where nn is the reply ID):

```
nn DFHPA1104 applid - SPECIFY ALTERNATIVE SIT PARAMETERS, IF ANY,
                      AND THEN TYPE '.END'.
```

You can enter as many initialization parameters as you can get on one line of the console, but you must use a comma to separate parameters. CICS continues to prompt for system initialization parameters with displays of message DFHPA1105 until you terminate console input by entering the .END control keyword.

Entering corrections to initialization parameters through the console

If you have coded PARMERR=INTERACT, and CICS detects a parameter error, either in the keyword or in the value you have assigned to it, CICS prompts you to correct the error with message DFHPA1912 or DFHPA1915:

```
DFHPA1912 applid  SIT OVERRIDE 'keyword' IS NOT RECOGNIZED.
                  SPECIFY CORRECT SIT OVERRIDE.

DFHPA1915 applid INVALID DATA HAS BEEN DETECTED FOR SIT OVERRIDE
                 'keyword'.  RESPECIFY THE OVERRIDE.
```

CICS prompts you to enter corrections to any errors it find in the PARM parameter or the SYSIPT data set ***after*** it has loaded the SIT, and as each error is detected. This means that if there is an APPLID parameter ***following*** the parameter that is in error, either in the PARM parameter or in the SYSIPT data set, it is the APPLID coded in the SIT that CICS displays in messages DFHPA1912 and DFHPA1915.

Notes on CICS resource table and module keywords

Table 29 shows the system initialization keywords for those CICS resources that:

- Have a suffix option
- Result in a dummy program or table if you code resource=NO
- You can COLD start individually.

Table 29. Summary of resources with a suffix, a dummy load module, or a COLD option

DFHSIT keyword	Default 1	Suffix 2	Dummy 3	COLD start 4
BMS	FULL	-	-	COLD
CLT	-	xx	-	-
DBP	-	xx	-	-
DCT	YES	xx	-	COLD
DIP	NO	-	program	-
DL1 or DLI	NO	-	-	COLD
FCT	YES	xx	-	-
ICP	-	-	-	COLD
JCT	YES	xx	program	-
MCT	NO	xx	5	-
PLTPI	NO	xx	-	-
PLTSD	NO	xx	-	-
SRT	YES	xx	-	-
TCT	YES	xx	table	-
TS	-	-	-	COLD
TST	NO	xx	-	-
XLT	NO	xx	-	-

Notes for Table 29:

1 The **Default** column indicates the default value for the keyword in the DFHSIT macro.

If you code YES in the SIT for those keywords with the suffix option, an unsuffixed version of the table or program is loaded. For example, DCT=YES results in a table called DFHDCT being loaded.

You can also select an unsuffixed module or table at CICS startup by specifying ***keyword=,*** or ***keyword=YES***. For example, if you code:

```
DBP=, or DBP=YES
FCT=, or FCT=YES
```

blanks are appended to DFHDBP and DFHFCT respectively, and these unsuffixed names are used during initialization.

The result of specifying ***keyword=,*** as a system initialization parameter through PARM, SYSIPT, or CONSOLE is not necessarily the same as in the DFHSIT macro. For example, TST=, (or omitted altogether) when coding the DFHSIT macro is taken to mean TST=NO, but TST=, through any of the other three methods is the same as TST=YES.

2 The **Suffix** column indicates whether you can code a suffix. (*xx* indicates that a suffix can be coded.)

The DBP keyword is mandatory; you must code either a suffix or YES for DBP. For more information about the DBP system initialization parameter, see page 229.

A suffix can be any 1 or 2 characters, but you must not use DY. You cannot use NO as a suffix.

If you code a suffix, a table or program with that suffix appended to the standard name is loaded. For example, DBP=2$ causes DFHDBP2$ dynamic backout program to be included in your CICS region.

When the suffix option is specified with other values, the two values must be enclosed within parentheses: for example, DCT=(xx,COLD).

3 The **Dummy** column indicates whether a dummy version of the program or table is loaded if you code NO. In some cases, coding NO for the operand associated with the **table** results in a dummy **program**. For more information about the effect of this option, see “Selecting versions of CICS programs and tables” on page 207.

4 The **COLD start** column indicates whether the resource can be forced to start COLD. (COLD indicates that the resource can be cold started individually).

If COLD is coded, it can be overridden only by coding START=(...,ALL) as a system initialization parameter. For more information about this option, see page 264.

For more information about CICS table and program selection, see “Selecting versions of CICS programs and tables” on page 207.

5 If you code MCT=NO, the CICS monitoring domain builds dynamically a default monitoring control table. This ensures that default monitoring control table entries are always available for use when monitoring is on and a monitoring class is active.

Selecting versions of CICS programs and tables

A CICS program is usually made up from a group of related CICS functional modules, one example of which is the terminal control program. For most CICS programs you can only have one version, which is supplied with CICS. However, for some CICS programs you can create more than one version; for example, with different service levels. To select a particular version of a program, you can include the load library containing that version in the CICS startup JCL. For the following programs, however, you can select from different versions, by specifying the version you require at system initialization:

1. The dynamic backout program (DBP).
2. The basic mapping support (BMS) program suite.

There are two ways of selecting the versions you need:

1. Using a suffix. Use this method for DBP.
2. Explicitly selecting the level of function needed. Use this method for BMS.

You can also specify that a program is ***not*** needed (see “Excluding unwanted programs” for details).

You can use these methods ***only*** for the programs referred to in this section and in “Excluding unwanted programs,” by coding system initialization parameters.

Using a suffix to select the dynamic backout program

Suffixes are used to distinguish different versions of the dynamic backout program (DBP), a CICS management program. Two versions of DBP are supplied with CICS, both in pregenerated format. These two suffixed versions are:

Suffix	**Description**
1$	CICS local DL/I is not supported.
2$	CICS local DL/I is supported.

Using an explicit level of function to select programs

You use an explicit level of function to select the BMS suite of programs. When you specify your BMS requirement on the BMS system initialization parameter, you can select one of three versions. The BMS level of function is selected by the parameter options MINIMUM, STANDARD, or FULL, from which the system initialization program loads the set of programs you require.

Excluding unwanted programs

The three ways of excluding programs that are not required are by specifying one of programname=NO, tablename=NO, or function=NO.

Specifying programname=NO

If you code programname=NO in your system initialization table (for example, DIP=NO), or as a SIT override parameter, you exclude the named management program at CICS system initialization.

The programs that you can exclude by coding programname=NO are:

- Batch data interchange program (DIP)
- Terminal control program (TCP).

Note: In the case of DIP, you get a dummy version of the management program, which is supplied on the distribution tape with a suffix of **DY**.

Specifying tablename=NO for the program's control table

Not all of the CICS programs have a programname parameter in the SIT. The alternative method of excluding them is to code NO against the table name for these programs. This has the same effect as coding NO against a program name parameter, and the associated CICS program is excluded at system initialization, either by loading a dummy program, or by some other technique.

The tables that can be used in this way, and their associated management programs, are shown in Table 30.

Table 30. SIT control tables with a NO option

Control tables	***Associated management modules***
Journal control table (JCT)	Journal control program (JCP)
System recovery table (SRT)	System recovery program (SRP)

A dummy version of the management program (suffixed **DY**) is supplied for journal control. The dummy program is DFHJCPDY. because you don't need DFHJCP if you specify no journaling.)

The dummy TCT, DFHTCTDY

There is a special case where you can also specify tablename=NO, but this does not load a dummy terminal control program. You specify TCT=NO when you are using resource definition online, and all of your terminal resource definitions are in the CSD.

When you specify TCT=NO, CICS loads a dummy TCT named DFHTCTDY. If you specify TCT=NO, a generated table of this name must be available in a sublibrary of the LIBDEF PHASE, SEARCH chain for the CICS job when you start CICS. A pregenerated dummy table, and its source, are provided in the VSE/ESA sublibrary, PRD1.BASE.

The dummy TCT provides ***only*** the CICS and VTAM control blocks that you need if you are using VTAM terminals and using the CSD for storing terminal definitions. You define your VTAM terminals using the RDO transaction, CEDA, a user application program using EXEC CICS CREATE TERMINAL commands, or the DEFINE command of the DFHCSDUP utility program.

Specifying function=NO

If you code function=NO as a system initialization parameter (for example, XRF=NO), you exclude the management program associated with the named function at CICS system initialization.

You can exclude CICS DL/I support, intersystem communication (ISC), the 3270 print-request facility, the system spooling interface, or the extended recovery facility (XRF), in this way.

Classes of start and restart

The type of initialization that CICS performs is not only determined by the START parameter. The CICS local and global catalogs also play a major role in the initialization process, together with any system initialization parameters that you provide, either in the SIT or at run time by one of the three methods described in this chapter.

The global catalog

CICS uses the global catalog to save all resource definitions that are installed at CICS shutdown. These are:

- Programs
- Transactions and transaction profiles
- Transaction classes
- Terminals, including any which are autoinstalled
- Typeterms
- Connections and sessions
- BMS maps sets and partition sets
- Files.

The resource definitions that CICS saves at its shutdown may have been installed during a cold start (from a list of groups specified by a group list system initialization parameter), or during CICS execution (by CEDA INSTALL or EXEC CICS CREATE commands).

If you run CICS with START=AUTO, and a warm or emergency restart results, CICS restores all the installed resource definitions as they were at normal CICS shutdown, or at the time of system failure. The general rule is that you cannot alter installed resource definitions during a restart except by coding START=COLD.

The CICS domains also use the global catalog to save their domain status between runs. In some cases this information can be overridden during a restart by supplying system initialization parameters. For example, CICS monitoring uses the cataloged status at a restart, but modified by any system initialization parameters you provide. In other cases the domain information saved in the catalog is always used in a restart.

For example, CICS statistics interval time is always restored from the catalog in a warm or emergency restart, because the statistics domain does not have this as a system initialization parameter. To change this you must use CEMT or EXEC CICS commands after control is given to CICS. Alternatively, you can enforce system defaults by performing a cold start.

Note: If you need to reinitialize the CICS global catalog for any reason, you must also reinitialize the local catalog.

The local catalog

The CICS domains use the local catalog to save some of their information between CICS runs. If you delete and redefine the local catalog, you must:

- Initialize the CICS local catalog with an initial set of domain records.
- Use the CICS-supplied utility program, DFHSMUTL, to re-add records to enable the CICS self-tuning mechanism for storage manager domain subpools. For details of how to do this, see the *CICS Operations and Utilities Guide*.
- Delete and reinitialize the CICS global catalog.

For more information about initializing the local catalog, see “The local catalog” on page 156.

Some of the information that is saved in the local catalog can be overridden at CICS system initialization by system initialization parameters, such as CICS transaction dump data set status.

Note: If you need to reinitialize the local catalog for any reason, you must also reinitialize the global catalog.

The START system initialization parameter

You can influence the type of startup that CICS performs, by specifying the START system initialization parameter, as follows:

START=AUTO

If you code AUTO as the START operand, CICS determines, by inspecting the control record in the global catalog, which of the following three types of start to perform.

1. **WARM**

   If the control record in the global catalog indicates that the previous run of CICS terminated normally with a successful warm keypoint, CICS performs a warm restart. The local catalog must also contain the information saved by the CICS domains during the previous execution for the warm restart to be successful. A warm start restores CICS to the state it was in at the previous shutdown.

   If you are using disk journaling, the status of the disk journals is also saved in the global catalog. This information is used by CICS at startup to determine which journal data set is to be opened. You can use the JSTATUS=RESET startup parameter to cause the status in the global catalog to be ignored. During CICS startup, the status of all journal data sets is set to “ready for use”. For more information about using JSTATUS=RESET, see “Resetting the journal status” on page 114 and the description of the JSTATUS parameter on page 244.

   You can modify a warm restart by coding the NEWSIT system initialization parameter. This has the effect of enforcing the system initialization parameters coded in the SIT, overriding any cataloged status from the previous CICS shutdown.

The exceptions to this are the system initialization parameters FCT, the CSDxxxxx group (for example CSDACC), and GRPLIST, which are always ignored in a warm restart, even if you specify NEWSIT=YES. Specifying NEWSIT=YES causes, in effect, a partial cold start.

2. **COLD**
   If there is no control record in the CICS global catalog, CICS assumes that the catalog has been newly initialized, and forces a cold start. If you have recreated the global catalog for some reason, you must also reinitialize the local catalog.

3. **EMERGENCY**
   If the control record in the global catalog indicates that the previous run of CICS terminated in an immediate or uncontrolled shutdown, CICS performs an emergency restart.

   The emergency restart procedure uses the system log at the time of the failure to return recoverable resources to their committed states.

START=AUTO should be the normal mode of operation, with the choice of start being made by CICS automatically.

START=COLD

If you code COLD as the START operand, CICS initializes using the resource definitions specified by the system initialization parameters, ignoring any previously installed resource definitions saved in a warm keypoint in the global catalog. This includes all the groups of resources specified by the GRPLIST= system initialization parameter, and those resources specified in CICS control tables.

Be aware, however, that a start initiated by START=COLD is not entirely without reference to the previous run of a CICS system using the same global catalog.

For example, CICS checks the ‘global catalog’ for any dataset name block entries for VSAM data sets for which backout failures have occurred previously.

You can perform a fully cold start of CICS, without reference to any previous execution, only by reinitializing both CICS catalogs. Generally, do this only when you are starting your CICS system for the first time.

There may be times when it is necessary to restart CICS with START=COLD, irrespective of the type of system termination that has been recorded in the global catalog.

START=STANDBY

The STANDBY option is for use only with XRF=YES. START=STANDBY initializes an alternate CICS region. If you code START=STANDBY with XRF=NO, initialization fails with message DFHXA6530, and CICS terminates abnormally with a dump.

CICS initializes as an alternate CICS region by beginning to perform an emergency restart, which is then suspended until it needs to perform a takeover. During the period when initialization is suspended, the alternate CICS region is in standby mode and monitors the active CICS region. When it takes over, the alternate CICS region completes the emergency restart and becomes the active CICS region.

If you have specified a COLD start for other CICS resources, for example, DCT=(xx,COLD), they are cold started when the alternate CICS region (with

START=STANDBY specified) takes over. This may cause CICS to lose data on an XRF takeover; for example, coding ICP=COLD results in outstanding STARTs being lost. You are recommended to code START=(STANDBY,ALL) to ensure a full emergency restart during takeover, unless you wish to specifically cold start individual resources.

For information about operating a CICS region with XRF, see the *CICS Operations and Utilities Guide*.

START=LOGTERM

The LOGTERM operand on the START parameter is a special restart option. It is for use only when the system log is defined on ***disk*** data sets, to cater for an abnormal termination of a CICS system that does not close the system log. The LOGTERM option causes a CICS restart to put only an end of file label on the log, and then terminate ***before*** doing any backout processing. You can use START=LOGTERM, for example, if you need to close the system log to perform offline file recovery, particularly in those cases when you know (or suspect) that emergency restart won't work.

LOGTERM is available only as a system initialization parameter at run time, and cannot be coded in the system initialization table. It is also intended for use only when you are running CICS with XRF=NO.

Table 31 (Page 1 of 2). Effect of the START= parameter in conjunction with the catalogs

START= parameter	**State of the CICS catalogs**	**Result at restart**
COLD	Local and global catalogs are both newly initialized.	CICS performs a fully cold start. All domains are initialized using system default values, modified by any system initialization parameters. All resources are installed as specified by system initialization parameters.
COLD	The global catalog contains a successful warm keypoint from previous run, and the local catalog contains information saved by the CICS domains.	CICS performs a cold restart, installing the resource definitions specified by system initialization parameters. The domains initialize according to system initialization parameters, or using system default values where there are no parameters (for example the statistics domain). CICS also checks the global catalog for any data set name block entries for VSAM data sets for which backout failures were recorded in the system log.
AUTO	Local and global are both newly initialized.	CICS enforces a fully cold start. All domains are initialized with system default values, modified by any system initialization parameters. All resources are installed as specified by system initialization parameters.
AUTO	The global catalog contains a successful warm keypoint from the previous run. The local catalog is newly initialized.	CICS begins to perform a warm restart, but fails during the initialization process because of absence of expected records in the local catalog. (***Do not reinitialize only one of the catalogs***.)

Table 31 (Page 2 of 2). Effect of the START= parameter in conjunction with the catalogs

START= parameter	**State of the CICS catalogs**	**Result at restart**
AUTO	The global catalog contains a successful warm keypoint from the previous run, and the local catalog contains information saved by the domains.	CICS performs a warm restart, restoring all of your CICS system except the trace domain to the same status it was in at CICS shutdown. (CICS trace domain does not save the status of the various trace options at CICS shutdown, and always uses the system initialization parameters.) Only those resources that have a COLD option on their system initialization parameter can be cold started (for example, the destination control table or auxiliary temporary storage).

Table 32 shows the effect of the various START options, combined with system initialization parameters where applicable, on the CICS trace, monitoring, statistics, and dump domains:

Table 32 (Page 1 of 2). Effect of the START= parameter on the CICS domains at initialization

Domain	**State of the CICS catalogs**	**Result at startup if START=AUTO**	**Result at startup if START=COLD**
Trace	Not relevant	Domain initializes according to the system initialization parameters.	Domain initializes according to the system initialization parameters.
Monitoring	The global catalog is newly initialized.	Domain initializes according to the system initialization parameters.	Domain initializes according to the system initialization parameters.
Monitoring	The global catalog contains status of monitoring at the previous CICS shutdown.	Domain uses monitoring status from the catalog, but modified by any system initialization override parameters.	Domain initializes according to the system initialization parameters.
Statistics	The global catalog is newly initialized.	Domain initializes according to CICS-defined system default values.	Domain initializes according to CICS-defined system default values.
Statistics	The global catalog contains status of statistics at CICS shutdown.	Domain uses statistics status from the catalog.	Domain initializes according to CICS-defined system default values.

Table 32 (Page 2 of 2). Effect of the START= parameter on the CICS domains at initialization

Domain	State of the CICS catalogs	Result at startup if START=AUTO	Result at startup if START=COLD
Dump	The global catalog is newly initialized.	Domain initializes the dump table according to CICS-defined system default values. Other dump attributes are set by system initialization parameters.	Domain initializes an empty dump table, and takes CICS-defined default action for all dump requests. Other dump attributes are set by system initialization parameters.
Dump	The global catalog contains dump status at CICS shutdown.	Domain reads the dump table and dump status from the catalog, but the dump status is modified by any system initialization parameters.	Domain initializes an empty dump table, and takes CICS-defined default action for all dump requests. Other dump attributes are set by system initialization parameters.

CICS startup and the VTAM session

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

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

If you receive messages DFHSI1589D and DFHSI1572, and if the CICS region is not initializing as an alternate CICS region, you can start the CICS-VTAM session manually when VTAM is eventually started, by means of the CEMT SET VTAM OPEN command from any VSE console defined to CICS.

If VTAM is active, but CICS still cannot open the VTAM ACB because VTAM does not recognize the CICS APPLID, you receive the following messages:

```
+DFHSI1592I 'applid' CICS applid not (yet) active to VTAM.
+DFHSI1572  'applid' Unable to OPEN VTAM ACB - RC=00000008, ACB CODE=5A.
```

This may be caused by an error in the value of APPLID operand, in which case you must correct the error and restart CICS. For information about other causes and actions, see the *VSE/ESA Messages and Codes Volume 3* manual.

Concurrent initialization of VTAM and XRF alternate CICS regions

An XRF alternate CICS region cannot initialize properly until it has successfully opened the VTAM ACB.

Because VTAM and the alternate CICS region may be initialized concurrently, it is possible that several tries may have to be made to open the VTAM ACB. If VTAM is not active, the following message is written to the system console every 15 seconds:

```
DFHSI1589D 'applid' VTAM is not currently active.
```

If VTAM is active, but CICS cannot open the VTAM ACB, the following messages are written to the system console:

```
+DFHSI1572 'applid' Unable to OPEN VTAM ACB - RC=xxxxxxxx, ACB CODE=yy.
DFHSI1590 'applid'  XRF alternate cannot proceed without VTAM.
```

CICS abends with a dump (abend code 1590).

End of CICS startup

Whichever type of startup is performed, when the message:

```
DFHSI1517 'applid' Control is being given to CICS.
```

is displayed on the operating system console, CICS is ready to process terminal requests (*applid* is the value of the specific APPLID system initialization parameter).

When the startup process is completed, users are able to enter transactions from any terminals that are connected to CICS. For information about the CICS-supplied transactions, see the *CICS-Supplied Transactions* manual.

The system initialization parameter descriptions

Unless otherwise stated, all of the system initialization parameters described here can be defined to CICS by any of these four ways:

1. In a DFHSIT macro
2. In a PARM parameter on the DFHSIP statement
3. In the SYSIPT data set of the CICS startup job stream
4. Through the system console.

Default notation

Default values are **underscored**; for example, TYPE=**CSECT**. This notation applies to the SIT macro parameters only.

TYPE={CSECT|DSECT}
Indicates the type of SIT to be generated.

CSECT
A regular control section that is normally used.

DSECT
A dummy control section.

ADI={30|number}
Specifies, when you are running CICS with XRF, the alternate delay interval in seconds. The minimum delay that you can specify is 5 seconds. This is the time that must elapse between the (apparent) loss of the surveillance signal in the active CICS system, and any reaction by the alternate CICS system. The corresponding parameter for the active is PDI. ADI and PDI need not have the same value.

AIEXIT={DFHZATDX|DFHZATDY|name}
Specifies the name of the autoinstall user-replaceable program that you want CICS to use when autoinstalling local VTAM terminals, APPC connections, and remote terminals. Autoinstall is the process of installing resource definitions

automatically, using VTAM logon or BIND data, model definitions, and an autoinstall program.

Important

You can specify only one user-replaceable program on the AIEXIT parameter. Which of the CICS-supplied programs (or customized versions thereof) that you choose depends on what combination of resources you need to autoinstall.

For background information about autoinstall, see the *CICS Resource Definition Guide*.

DFHZATDX
A CICS-supplied autoinstall user program. This is the default. It installs definitions for locally-attached VTAM terminals, remote shipped terminals, and remote shipped connections.

DFHZATDY
A CICS-supplied autoinstall user program. It installs definitions for both locally-attached VTAM terminals, local APPC connections, remote shipped terminals, and remote shipped connections.

name
The name of your own customized autoinstall program, which may be based on one of the supplied sample programs. For programming information about writing your own autoinstall program, see the *CICS Customization Guide*.

AILDELAY={<u>0</u>|hhmmss}
Specifies the delay period that elapses after a session between CICS and an autoinstalled terminal is ended, before the terminal entry is deleted. A session is ended when a terminal logs off or when a transaction disconnects a terminal from CICS.

hhmmss
Specify a 1-to 6-digit number. The default is 0, meaning the terminal entry is deleted as soon as the session is ended. If you leave out the leading zeros, they are supplied (for example, 123 becomes 000123, that is, 1 minute 23 seconds).

Note: The AILDELAY parameter does not apply to autoinstall of APPC connections, because they are not deleted.

AIQMAX={<u>100</u>|number}
Specifies the maximum number of VTAM terminals and APPC connections that can be queued concurrently for autoinstall.

number
A number in the range 0 through 999. The default is 100.

A zero value disables the autoinstall function.

You should specify a number that is large enough to allow for both APPC connnections and terminals.

Note: This value does not limit the total number of terminals that can be autoinstalled. If you have a large number of terminals autoinstalled, shutdown can fail due to the MXT system initialization parameter being reached or CICS becoming short on storage. For information about preventing this possible cause of shutdown failure, see the *CICS Performance Guide*.

AIRDELAY={700|hhmmss}
Specifies the delay period that elapses after an emergency restart before autoinstalled terminal entries that are not in session are deleted.

hhmmss
Specify a 1-to 6-digit number. If you leave out the leading zeros, they are supplied. The default is 700, meaning a delay of 7 minutes. A value of 0 means that autoinstalled terminal definitions are not written to the global catalog and therefore are not restored at an emergency restart. For guidance about the performance implications of setting different AIRDELAY values, see the *CICS Performance Guide*.

Note: The AIRDELAY parameter does not apply to autoinstall of APPC connections, because they are not cataloged.

XRF restriction

If you are running CICS with XRF, set the same value on the AIRDELAY parameter for both the active and the alternate CICS systems. It is particularly important, if you want autoinstall sessions to be reestablished after a takeover, that you avoid coding a zero on this parameter for either the active or the alternate CICS systems.

For background information, see the *CICS XRF Guide*.

AKPFREQ={200|number}
If AKPFREQ is a number other than zero, it specifies the number of consecutive blocks, written by DFHJCP to the system log data set, that triggers the activity keypoint function. The minimum number that should be coded is 200 (the default) and the maximum number is 65535. (The CICS region must support activity keypointing: that is, the CSKP transaction and DFHAKP program must be defined. For information about supporting activity keypointing, see the *CICS Recovery and Restart Guide* and the *CICS Performance Guide*.)

If AKPFREQ=0 is coded, no activity keypoints are taken and a subsequent emergency restart is not possible.

APPLID={DBDCCICS|applid}
The VTAM application identifier for this CICS system.

applid
This name, 1 through 8 characters, identifies the CICS system in the VTAM network. It must match the name field specified in the APPL statement of the VTAM VBUILD TYPE=APPL definition.

When you define this CICS system to another CICS system, in a CONNECTION definition, you specify the applid as the NETNAME.

If the CICS system uses XRF, the form of the APPLID parameter is:

APPLID=(generic_applid,specific_applid)
Specifies the generic and specific XRF applids for the CICS system. Both applids must be 1 through 8 characters.

generic_applid

This is the generic applid for both the active and the alternate CICS systems. Therefore, you must specify the same name for *generic_applid* on the APPLID system initialization parameter for both CICS systems. Because IRC uses *generic_applid* to identify the CICS systems, there can be no IRC connection for an alternate CICS system until takeover has occurred and the alternate CICS system becomes the active CICS system.

When you define this XRF pair to another CICS system, in a CONNECTION definition, you specify the generic applid as the NETNAME.

specific_applid

This identifies the CICS system in the VTAM network. It must match the label specified in the VTAM VBUILD TYPE=APPL definition. You must specify a different *specific_applid* on the APPLID system initialization parameter for the active and for the alternate CICS system. Also, *generic_applid* and *specific_applid* must be different.

The active and alternate CICS systems use the VTAM MODIFY USERVAR command to set a user application name variable, so end users do not need to know which CICS system is active at any instant. For background information about using this command, see the *CICS XRF Guide*.

AUTCONN={<u>0</u>|hhmmss}

Specify this to delay the reconnection of terminals after an XRF takeover, to allow time for manual switching. The delay is hh hours, mm minutes, ss seconds. The default value of zero means that there is no delay in the attempted reconnection.

The interval specified is the delay before the CXRE transaction runs. CXRE tries to reacquire terminals that were in session at the time of the takeover.

Note that the same delay interval applies to the connection of terminals with AUTOCONNECT(YES) specified in the RDO TYPETERM definition, at a warm or emergency restart, whether or not you have coded XRF=YES.

AUXTR={<u>OFF</u>|ON}

Indicates if the auxiliary trace destination is to be activated at system initialization. This parameter controls whether any of the three types of CICS trace entry are written to the auxiliary trace data set. The three types are: CICS system trace (see the SYSTR parameter), user trace (see the USERTR parameter), and exception trace entries (that are always made and are not controlled by a system initialization parameter).

<u>OFF</u>

Do not activate auxiliary trace.

ON

Activate auxiliary trace.

For details of internal tracing in main storage, see the INTTR parameter on page 243.

AUXTRSW={<u>NO</u>|ALL|NEXT}

Specifies whether you want the auxiliary trace autoswitch facility.

<u>NO</u>
Disables the autoswitch facility.

NEXT
Enables the autoswitch facility to switch to the next data set at end of file of the first data set used for auxiliary trace. Coding NEXT permits one switch only, and when the second data set is full, auxiliary trace is switched off.

ALL
Enable the autoswitch facility to switch to the inactive data set at every end of file. Coding ALL permits continuous switching between the two auxiliary trace data sets, DFHAUXT and DFHBUXT, and whenever a data set is full, it is closed and the other data set is opened.

BMS=({MINIMUM|STANDARD|<u>FULL</u>}[, COLD][,{<u>UNALIGN</u>|ALIGN}][, { <u>DDS</u>|NODDS}])
Specifies which version of basic mapping support you want to be included. The function included in each version of BMS is shown in Table 33 on page 220. The parameter BMS can be overridden during CICS initialization.

You need full or standard function BMS, if you are using XRF and have specified MESSAGE for RECOVNOTIFY on any of your RDO TYPETERM resource definitions.

MINIMUM
The minimum version of BMS is included.

STANDARD
The standard version of BMS is included.

<u>FULL</u>
The full version of BMS is included. This is the default in the SIT.

COLD
CICS deletes delayed messages from temporary storage, and destroys their interval control elements (ICEs).

<u>UNALIGN</u>
Specifies that all BMS maps assembled before CICS/DOS/VS Version 1 Release 6 are unaligned. Results are unpredictable if the stated alignment does not match the actual alignment.

ALIGN
Code this to indicate that all BMS maps assembled before CICS/DOS/VS Version 1 Release 6 are aligned.

<u>DDS</u>
BMS is to load suffixed versions of map sets and partition sets. BMS first tries to load a version that has the alternate suffix (if the transaction uses the alternate screen size). If the load fails, BMS tries to load a version that has the default map suffix. If this fails too, BMS tries to load the unsuffixed version. DDS, which stands for “device dependent suffixing”, is the default.

You need to use map suffixes only if the same transaction is to be run on terminals with different characteristics (in particular, different screen sizes). If you do not use suffixed versions of map sets and partition sets, CICS need not test for them.

NODDS

BMS is not to load suffixed versions of map sets and partition sets. Specifying NODDS avoids the search for suffixed versions, saving processor time.

Table 33. Versions of BMS

BMS version	**Devices supported**	**Function provided**
MINIMUM	All 3270 system display units and printers except SNA character string printers, which are defined as DEVICE(SCSPRINT) on the RDO TYPETERM definition or as TRMTYPE=SCSPRT in DFHTCT	SEND MAP command, RECEIVE MAP command, SEND CONTROL command. Default and alternate screens; extended attributes; map set suffixes; screen coordination with null maps; and block data
STANDARD	All devices are supported by BMS. These are listed in the *CICS Application Programming Guide*	All function of MINIMUM, ***plus*** outboard formats, partitions, controlling a magnetic slot reader, NLEOM mode for 3270 system printers, SEND TEXT command, and Subsystem LDC controls
FULL	All devices supported by BMS. These are listed in the *CICS Application Programming Guide*	Same as STANDARD, ***plus*** terminal operator paging, cumulative mapping, page overflow, cumulative text processing, routing, message switching returning BMS-generated data stream to program before output.

CDSASZE={0K|number}

Specifies the size of the CDSA. The default size is 0 indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

number

Specify number as an amount of storage in the range 0 through 16777215 bytes in multiples of 262144 bytes (256KB). If the size specified is not a multiple of 256KB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4194304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

Restrictions

You can code the CDSASZE parameter in PARM, SYSIPT, or CONSOLE only.

CHKSTRM={CURRENT|<u>NONE</u>}
Activate, or deactivate, terminal storage-violation checking. The operands have the following meanings:

CURRENT
Code CURRENT to check for TIOA storage violations.

<u>NONE</u>
Code NONE to deactivate TIOA storage-violation checking.

You can also use the CICS-supplied transaction, CSFE, to switch terminal storage-violation checking on and off.

For information about checking for storage violations, see the *CICS Problem Determination Guide*.

Restrictions

You can code the CHKSTRM parameter in PARM, SYSIPT, or CONSOLE only.

CHKSTSK={ALL|CURRENT|<u>NONE</u>}
Activate, or deactivate, task storage-violation checking at startup. The operands have the following meanings:

ALL
Code ALL to check all storage areas on the transaction storage chains for all tasks.

CURRENT
Code CURRENT to check all storage areas on the transaction storage chain for the current task only.

<u>NONE</u>
Code NONE to deactivate task storage-violation checking.

You can also use the CICS-supplied transaction, CSFE, to switch task storage-violation checking on and off.

For information about checking for storage violations, see the *CICS Problem Determination Guide*.

Restrictions

You can code the CHKSTSK parameter in PARM, SYSIPT, or CONSOLE only.

CLSDSTP={<u>NOTIFY</u>|NONOTIFY}
Specifies the notification required for an EXEC CICS ISSUE PASS command. This parameter is applicable to both autoinstalled and non-autoinstalled terminals. You can use the notification in a user-written node error program to reestablish the CICS session when a VTAM CLSDST PASS request resulting from an EXEC CICS ISSUE PASS command fails. For more information about the EXEC CICS ISSUE PASS command, see the *CICS Application Programming Reference* manual.

<u>NOTIFY</u>
CICS requests notification from VTAM when the EXEC CICS ISSUE PASS command is executed.

NONOTIFY
CICS does not request notification from VTAM.

CLT=xx
Specify the suffix for the command list table (CLT), if this SIT is used by an alternate XRF system. The name of the table is DFHCLTxx.

For information about coding the macros for this table, see the *CICS Resource Definition Guide*.

CMDPROT={<u>YES</u>|NO}
Specifies whether to allow, or inhibit, CICS validation of start addresses of storage referenced as output parameters on EXEC CICS commands.

<u>YES</u>
If you specify YES, CICS validates the initial byte at the start of any storage that is referenced as an output parameter on EXEC CICS commands to ensure that the application program has write access to the storage. This ensures that CICS does not overwrite storage on behalf of the application program when the program itself cannot do so. If CICS detects that an application program has asked CICS to write into an area to which the application does not have addressability, CICS abends the task with an AEYD abend.

The level of protection against bad addresses depends on the level of storage protection in the CICS environment. The various levels of protection provided when you specify CMDPROT=YES are shown in Table 34.

NO
If you specify NO, CICS does not perform any validation of addresses of the storage referenced by EXEC CICS commands. This means an application program could cause CICS to overwrite storage to which the application program itself does not have write access.

Table 34. Levels of protection provided by CICS validation of application-supplied addresses

Environment	Execution key of affected programs	Types of storage referenced by applications that cause AEYD abends
Read-only storage (RENTPGM=PROTECT)	CICS-key and user-key	CICS key 0 read-only storage (RDSA and ERDSA).
Subsystem storage protection (STGPROT=YES)	User-key	All CICS-key storage (CDSA and ECDSA)
Base CICS (all storage is CICS key storage) (RENTPGM=NOPROTECT; and STGPROT=NO	CICS-key and user-key	VSE storage only

CMDSEC={<u>ASIS</u>|ALWAYS}
Specifies whether or not you want CICS to honor the CMDSEC option specified on a transaction's resource definition.

<u>ASIS</u>
means that CICS honors the CMDSEC option defined in a transaction's resource definition. CICS calls its command security checking routine only

when CMDSEC(YES) is specified in a RDO TRANSACTION resource definition.

ALWAYS
means that CICS overrides the CMDSEC option, and always calls its command security checking routine to issue the appropriate call to the System Authorization Facility (SAF) interface.

Notes:

1. Specify ALWAYS when you want to control the use of the SPI in all your transactions. Be aware that this might incur additional overhead. The additional overhead is caused by CICS issuing the command security calls on every eligible EXEC CICS command, which are *all* the system programming interface (SPI) commands.
2. If you specify ALWAYS, command checking applies to CICS-supplied transactions such as CESN and CESF. You must authorize all users of CICS-supplied transactions to use the internal CICS resources for the transactions, otherwise you will get unexpected results in CICS-supplied transactions.

Restrictions

You can code the CMDSEC parameter in the SIT, PARM, or SYSIPT only.

CONFDATA={SHOW|HIDETC}
Specifies whether CICS is to suppress (hide) user data that might otherwise appear in CICS trace entries or in dumps that contain the VTAM receive any input area (RAIA.). This option applies to initial input data received on a VTAM RECEIVE ANY operation, the initial input data received on an MRO link, and FEPI screens and RPLAREAs.

SHOW
Data suppression is not in effect. User data is traced regardless of the CONFDATA option specified in transaction resource definitions. This option overrides the CONFDATA option in RDO TRANSACTION resource definitions.

HIDETC
Specifies that you want CICS to ‘hide’ user data from CICS trace entries. It also indicates that VTAM RAIAs are to be suppressed from CICS dumps. The action actually taken by CICS is subject to the individual CONFDATA attribute on the RDO TRANSACTION resource definition (see Table 35 on page 225).

If you specify CONFDATA=HIDETC, CICS processes VTAM, MRO, and FEPI user data as follows:

- **VTAM**

  CICS clears the VTAM RAIA containing initial input as soon as it has been processed, and before the target transaction has been identified.

  The normal trace entries (FC90 and FC91) are created on completion of the RECEIVE ANY operation with the text “`SUPPRESSED DUE TO CONFDATA=HIDETC IN SIT`” replacing all the user data except the first 4 bytes of normal data, or the first 8 bytes of function management headers (FMHs).

CICS then identifies the target transaction for the data. If the RDO TRANSACTION resource definition specifies CONFDATA(NO), CICS traces the user data that it suppressed from the FC90 trace in the trace entry AP FC92. This trace entry is not created if the transaction is defined with CONFDATA(YES) on the RDO TRANSACTION resource definition.

- **MRO**

  CICS does not trace the initial input received on an MRO link.

  The normal trace entries (DD16, DD23, and DD25) are created with the text "`SUPPRESSED DUE TO CONFDATA=HIDETC IN SIT`" replacing all the user data.

  CICS then identifies the target transaction for the data. If the transaction definition specifies CONFDATA(NO), CICS traces the user data that it suppressed from DD16 in the trace entry AP FC92. This special trace entry is not created if the transaction is defined with CONFDATA(YES).

- **FEPI**

  FEPI screens and RPL data areas (RPLAREAs) areas are suppressed from all FEPI trace points if CONFDATA(YES) is specified in the transaction resource definition. The user data in the FEPI trace points AP 1243, AP 1244, AP 145E, AP 145F, AP 1460, AP 1461, AP 1595, AP 1596, AP 1597, AP 1598, and AP 1599 is replaced with the message "`SUPPRESSED DUE TO CONFDATA=HIDETC IN SIT`." If the RDO TRANSACTION resource definition specifies CONFDATA(NO), the FEPI trace entries are created with the user data as normal.

Mirror transactions

The CICS-supplied mirror RDO TRANSACTION resource definitions are specified with CONFDATA(YES). This ensures that, when you specify CONFDATA=HIDETC as a system initialization parameter, CICS regions running mirror transactions suppress user data as described for VTAM and MRO data.

Modified data

By waiting until the transaction has been identified to determine the CONFDATA option, VTAM or MRO data may have been modified (for example, it may have been translated to upper case).

The interaction between the CONFDATA system initialization parameter and the CONFDATA attribute on the RDO TRANSACTION resource definition is shown in Table 35 on page 225.

Table 35. Effect of CONFDATA system initialization and transaction definition parameters

CONFDATA on RDO TRANSACTION resource definition	**CONFDATA system initialization parameter**	
	SHOW	**HIDETC**
NO	Data not suppressed	VTAM RAIAs are cleared. Initial input of VTAM and MRO data is suppressed from the normal FC90, FC91, DD16, DD23, and DD25 trace entries. For FC90 and DD16 traces only, suppressed user data is traced separately in an FC92 trace entry. FEPI screens and RPLAREAs are traced as normal.
YES	Data not suppressed	VTAM RAIAs are cleared. All VTAM, MRO, and FEPI user data is suppressed from trace entries.

You cannot modify the CONFDATA option while CICS is running. You must restart CICS to make such a change.

Restrictions

You can code the CONFDATA parameter in the SIT, PARM, and SYSIPT only.

CONFTXT={NO|YES}
Specifies whether CICS is to prevent VTAM from tracing user data.

NO
CICS does not prevent VTAM from tracing user data.

YES
CICS prevents VTAM from tracing user data.

Restrictions

You can code the CONFTXT parameter in the SIT, PARM, and SYSIPT only.

CSDACC={READWRITE|READONLY}
Specifies the type of access to the CSD to be permitted to this CICS region. Note that this parameter is effective only when you start CICS with a START=COLD parameter. If you code START=AUTO, and CICS performs a warm or emergency restart, the file resource definitions for the CSD are recovered from the CICS global catalog. However, you can redefine the type of access permitted to the CSD dynamically with a CEMT SET FILE, or an EXEC CICS SET FILE, command.

READWRITE
Read/write access is allowed, permitting the full range of CEDA, CEDB, and CEDC functions to be used.

READONLY
Read access only is allowed, limiting the CEDA and CEDB transactions to only those functions that do not require write access.

CSDBUFND=number
Specifies the number of buffers to be used for CSD data. The minimum you should specify is the number of strings coded on the CSDSTRNO parameter plus 1, up to a maximum of 32768. Note that this parameter is used only if you have also coded CSDLSRNO=NONE; if you have coded CSDLSRNO=number, CSDBUFND is ignored.

If you specify a value for CSDBUFND that is less than the required minimum (the CSDSTRNO value plus 1), VSAM automatically changes the number of buffers to the number of strings plus 1 when CICS issues the OPEN macro for the CSD.

This parameter is effective only on a CICS cold start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog.

CSDBUFNI=number
Specifies the number of buffers to be used for the CSD index. The minimum you should specify is the number of strings coded on the CSDSTRNO parameter, up to a maximum of 32768. This parameter is used only if you have also coded CSDLSRNO=NONE; if you have coded CSDLSRNO=number, CSDBUFNI is ignored.

If you specify a value for CSDBUFNI that is less than the required minimum (the CSDSTRNO value), VSAM automatically changes the number of buffers to the number of strings when CICS issues the OPEN macro for the CSD.

This parameter is effective only on a CICS cold start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the CICS global catalog.

CSDFRLOG=number
Specifies a journal identifier to indicate the journal that you want to use for forward recovery of the CSD. This parameter is used only if CSDRECOV=ALL is specified, otherwise it is ignored. If you omit CSDFRLOG, but specify CSDRECOV=ALL, CSDFRLOG defaults to 1, which indicates that the CICS system log is to be used for forward recovery of the CSD.

The CSDRECOV, and CSDFRLOG system initialization parameters interact according to how they are specified. For information about their effects when the SIT is assembled and during CICS override processing, see “Planning for backup and recovery” on page 140.

This parameter is effective only on a CICS cold start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog.

number
The journal identification of the journal that is to be used for forward recovery. The number must be in the range 1 through 99. The number 1 indicates the CICS system log, and any other number refers to a user journal.

CSDJID={<u>NO</u>|number}
Specifies the journal identifier of the journal that you want CICS to use for automatic journaling of file requests against the CSD.

This parameter is effective only on a CICS cold start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog.

<u>NO</u>
Code NO if you do ***not*** want automatic journaling for the CSD. This is the default.

number
A number in the range 1 through 99 to identify the journal that CICS is to use for automatic journaling for the CSD. The number 1 indicates the CICS system log, and any other number refers to another CICS journal.

The automatic journaling options enforced for the CSD when you code CSDJID=number are JNLADD=BEFORE and JNLUPDATE=YES. These options are sufficient to record enough information for a user-written forward recovery utility. No other automatic journaling options are available for the CSD. For information about the options JNLADD=BEFORE and JNLUPDATE=YES, see the *CICS Resource Definition Guide*.

CSDLSRNO={<u>1</u>|number|NONE|NO}
Specifies whether the CSD is to be associated with a local shared resource (LSR) pool.

This parameter is effective only on a CICS cold start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog. However, you can redefine the LSR pool attribute for the CSD dynamically with an EXEC CICS SET FILE command.

<u>1</u> The default LSR pool number is 1.

number
The number of the LSR pool the CSD is to be associated with. The number of the pool must be in the range 1 through 15.

NONE|NO
Code NONE (or NO) if the CSD is not to be associated with an LSR pool.

CSDRECOV={<u>NONE</u>|ALL|BACKOUTONLY}
Specifies whether the CSD is a recoverable file.

The CSDRECOV, and CSDFRLOG system initialization parameters interact according to how they are specified. For information about their effects when the SIT is assembled and during CICS override processing, see “Planning for backup and recovery” on page 140.

This parameter is effective only on a CICS cold start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog.

<u>NONE</u>
The default is that the CSD is not recoverable.

ALL
Code ALL to specify that you want both forward recovery and backout for the CSD. If you code ALL, also specify CSDFRLOG with the journal

identification of the journal to be used for forward recovery of the CSD. If you do not code the CSDFRLOG parameter, CICS uses the system log for forward recovery.

Note: For backout purposes, CICS always uses the system log. (See the BACKOUTONLY option.)

BACKOUTONLY
Code BACKOUTONLY to limit CSD recovery to file backout only. If you specify backout for the CSD, CICS uses the system log to record before images for backout purposes.

CSDSTRNO={2|number}
Specifies the number of concurrent requests that can be processed against the CSD. When the number of requests reaches the STRNO value, CICS automatically queues any additional requests until one of the active requests terminates.

CICS requires two strings per CSD user, and you can increase the CSDSTRNO value, in multiples of two, to allow more than one concurrent CSD user.

See "Multiple users of the CSD within a CICS region" on page 135 before you code this parameter.

This parameter is effective only on a CICS cold start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog. However, you can redefine the number of strings for the CSD dynamically with an EXEC CICS SET FILE command.

2 The minimum number of concurrent requests for the CSD is 2.

number
This number must be a multiple of 2, in the range 2 through 254.

CWAKEY={USER|CICS}
Specifies the storage key for the common work area (CWA) if you are operating CICS with storage protection (STGPROT=YES). (You specify how much storage you want for the CWA on the WRKAREA parameter.) The permitted values are USER (the default), or CICS:

USER If you specify USER, or allow this parameter to default, CICS obtains storage for the CWA in user key. This allows a user program executing in any key to modify the CWA.

CICS If you specify CICS, CICS obtains storage for the CWA in CICS key. This means that only programs executing in CICS key can modify the CWA, and user-key programs have read-only access.

If CICS is running without storage protection, the CWAKEY parameter is ignored, and the CWA is always allocated from CICS-key storage.

DATFORM={MMDDYY|DDMMYY|YYMMDD}
Specifies the external date display standard that you want to use for CICS date displays. An appropriate indicator setting is made in the CSA. It is examined by CICS supplied system service programs that display a Gregorian date. CICS maintains the date in the form 0CYYDDD in the CSA (where C=0 for years 19xx, 1 for years 20xx, and so on; YY=year of century; and DDD=day of year), and converts it to the standard you specify for display.

MMDDYY
The date is in the form of month-day-year.

DDMMYY
The date is in the form of day-month-year.

YYMMDD
The date is in the form of year-month-day.

DBP={1$|2$|xx|YES}
Specifies which version of the dynamic transaction backout program is to be part of the system. This DBP parameter is mandatory, and has no default. (See page 207 for more information about coding this parameter.)

If you are using local DL/I support, specify DBP=2$. If you have generated a user-defined DFHDBP program for use with local DL/I support, specify either DBP=xx, where xx is the suffix of your program, or DBP=YES if you have defined an unsuffixed version of DFHDBP.

If you are not using local DL/I support, specify DBP=1$.

For background information about dynamic transaction backout, see the *CICS Recovery and Restart Guide*.

The dynamic transaction backout program needs a program resource definition entry in the CSD. This entry must be included in your GRPLIST list at CICS initialization. The CICS-supplied programs DFHDBP1$ and DFHDBP2$ have resource definition entries in the CSD group DFHBACK, which is included in the default group list DFHLIST. If you use your own dynamic backout program, you must create a RDO PROGRAM resource definition entry for it in the CSD, and include the entry in one of your GRPLIST lists at CICS initialization. You are recommended to code RESIDENT(YES) on your RDO PROGRAM definition. For information about the required program resource definition entry, see the *CICS Resource Definition Guide*.

Restrictions

You must code a DBP parameter in the system initialization table, although you can override that DBP parameter in PARM, SYSIPT, or CONSOLE.

DBUFSZ={500|number}
Specifies the maximum size of the dynamic log buffer (needed for dynamic transaction backout) for each transaction. CICS initially allocates for each buffer half the maximum size that you specify, and subsequently uses the value specified to calculate the actual size of the dynamic buffer. The value can be in the range 6 through 32000; the default is 500.

If the resultant dynamic buffer is too small, CICS spills the dynamic log data to a further area of main storage. (CICS allocates all dynamic log storage, including spilled buffers, above the 16MB line.)

For information on the dynamic log, and how you can tune this parameter, see *Dynamic transaction backout statistics* and *Dynamic log buffer size (DBUFSZ)* in the *CICS Performance Guide*.

DCT=({YES|xx|NO}[,COLD])
Specifies the destination control table suffix, for more information see page 205. For information about coding the macros for this table, see the *CICS Resource Definition Guide*.

DFLTUSER={CICSUSER|userid}
Specifies the external security manager (ESM) userid with the security attributes to be used for all terminal users who have not explicitly signed on (by the CESN transaction, the EXEC CICS SIGNON command, or by the preset security options of the TERMINAL resource definition).

The specified userid must be defined to your ESM if you are using external security (that is, you have coded SEC=YES).

The specified userid is signed on during CICS initialization. If it cannot be signed on, CICS fails to initialize.

Restrictions

You can code the DFLTUSER parameter in the SIT, PARM, or SYSIPT only.

DIP={NO|YES}
Code YES to include the batch data interchange program, DFHDIP, which supports the batch controller functions of the IBM 3790 Communication System and the IBM 3770 Data Communication System, for more information see page 205. (Support is provided for the transmit, print, message, user, and dump data sets of the 3790 system.)

DISMACP={YES|NO}
DISMACP=YES allows CICS to disable any transaction that terminates abnormally with an ASRD abend (caused by a user program invoking a CICS macro, or referencing the CSA or the TCA).

Note: DISMACP=YES has no effect if the ASRD abend is handled by an active abend exit.

{DLI|DL1}=({NO|xx|YES}[,COLD])
Specifies whether DL/I databases are to be accessed during this run of CICS.

This parameter selects only DL/I support that runs within the CICS address space.

NO
DL/I is not to be used.

xx DL/I is used. xx is the 1- or 2-character suffix of the DL/I application control table (ACT) specified on the DLZACT TYPE=INITIAL macro, which results in the module DLZNUCxx.

YES
DL/I is to be used. You must also specify DBP=2$.

COLD
If you code DLI=YES, you can specify the COLD option if you want to cold start the DL/I resources The default is the start option coded on the START parameter. If you specify COLD, DL/I databases are not backed out in the event of an emergency restart.

Note: On initialization of an XRF alternate system, the DLI COLD option is overridden so that DL/I is emergency restarted, and so DL/I databases are backed out.

DLIOER={ABEND|CONTINUE}
Code the DLIOER parameter to tell CICS what to do in the event of DL/I detecting an I/O error on a database.

ABEND
CICS abends. In an XRF configuration, the alternate system is canceled. Global user exit, XDBDERR, is not invoked.

CONTINUE
CICS continues to run. DL/I flags the database as unusable and allows the use of DL/I utilities to perform database recovery and to restart the database.

DSALIM={5M|number}
Specifies the upper limit of the total amount of storage within which CICS can allocate the individual dynamic storage areas (DSAs) that reside below the 16MB boundary.

5M
The default DSA limit is 5MB (5 242 880).

number
Specify *number* as an amount of storage in the range 2MB to 16MB (2 097 152 bytes to 16 777 216 bytes) in multiples of 262 144 bytes (256KB). If the size specified is not a multiple of 256KB, CICS rounds the value up to the next multiple.

You can specify *number* in bytes (for example, 4 194 304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

From the storage size that you specify on the DSALIM parameter, CICS allocates the following dynamic storage areas:

The user DSA (UDSA)
The user-key storage area for all user-key task-lifetime storage below the 16MB boundary.

The read-only DSA (RDSA)
The key-0 storage area for all reentrant programs and tables below the 16MB boundary.

The shared DSA (SDSA)
The user-key storage area for any non-reentrant user-key RMODE(24) programs, and also for any storage obtained by programs issuing EXEC CICS GETMAIN commands for storage below the 16MB boundary with the SHARED option.

The CICS DSA (CDSA)
The CICS-key storage area for all non-reentrant CICS-key RMODE(24) programs, all CICS-key task-lifetime storage below the 16MB boundary, and for CICS control blocks that reside below the 16MB boundary.

Note: CICS allocates the UDSA and the other DSAs below 16MB in multiples of 256KB. The maximum you can specify depends on a number of factors, such as how you have configured your VSE storage (which governs how much private storage remains below the line) and how much private storage you must leave free to satisfy VSE GETVIS requests for storage outside the DSAs.

For information about calculating the amount of storage to specify on the DSALIM parameter, see the *CICS Performance Guide*.

DSHIPIDL={020000|hhmmss}

Specifies the minimum time, in hours, minutes, and seconds, that an *inactive* shipped terminal definition must remain installed in this region. When the timeout delete mechanism is invoked, only those shipped definitions that have been inactive for longer than the specified time are deleted.

You can use this parameter in a transaction routing environment, on the application-owning and intermediate regions, to prevent terminal definitions having to be reshipped because they have been deleted prematurely.

The default minimum idle time is 2 hours.

hhmmss Specify a 1 to 6 digit number in the range 0–995959. Numbers that have fewer than six digits are padded with leading zeros.

DSHIPINT={120000|0|hhmmss}

Specifies the interval between invocations of the timeout delete mechanism. The timeout delete mechanism removes any shipped terminal definitions that have not been used for longer than the time specified by the DSHIPIDL parameter.

You can use this parameter in a transaction routing environment, on the application-owning and intermediate regions, to control:

- How often the timeout delete mechanism is invoked.
- The approximate time of day at which a mass delete operation is to take place, relative to CICS startup.

  Note: For more flexible control over when mass delete operations take place, you can use a CEMT SET DELETSHIPPED or EXEC CICS SET DELETSHIPPED command to reset the interval. (The revised interval starts *from the time the command is issued*, **not** from the time the remote delete mechanism was last invoked, nor from CICS startup.)

0 The timeout delete mechanism is not invoked. You might set this value in a terminal-owning region, or if you are not using shipped definitions.

hhmmss Specify a 1 to 6 digit number in the range 1–995959. Numbers that have fewer than six digits are padded with leading zeros.

DTRTRAN={CRTX|name|NO}

This is the name of the transaction definition that you want CICS to use for dynamic transaction routing. This is intended primarily for use in a CICS terminal-owning region, although you can also use it in an application-owning region when you want to daisy-chain transaction routing requests. In a dynamic transaction routing environment, the transaction named on DTRTRAN must be installed in the CICS terminal-owning regions if you want to eliminate the need for resource definitions for individual transactions.

CRTX

This is the default dynamic transaction definition. It is the name of the CICS-supplied sample transaction resource definition provided in the CSD group DFHISC.

name
The name of your own dynamic transaction resource definition that you want CICS to use for dynamic transaction routing.

NO
The dynamic transaction routing program is not invoked when a transaction definition cannot be found.

For information about the CICS-supplied sample transaction resource definition, CRTX, and about defining you own dynamic transaction routing definition, see the *CICS Resource Definition Guide*.

DTRPGM={DFHDYP|program-name}
Specifies the name of the dynamic transaction routing program you want to use for routing transactions that are defined with the DYNAMIC attribute. DFHDYP, the default, is the name of the CICS-supplied version.

DUMP={YES|NO}
Specifies whether the CICS dump domain is to take SDUMPs.

YES
SDUMPs are produced, unless suppressed by the options specified in the CICS system dump table.

NO
SDUMPs are suppressed.

Note: This does not prevent the CICS kernel from taking SDUMPs.

For more information about SDUMPs, see "System dumps" on page 163.

DUMPDS={AUTO|A|B}
Specifies the transaction dump data set that is to be opened during CICS initialization.

AUTO
For all types of start, CICS opens the transaction dump data set that was ***not*** in use when the previous CICS run terminated. This information is obtained from the CICS local catalog (DFHLCD).

If you specify AUTO, or let it default, code DLBL statements for both of the transaction dump data sets, DFHDMPA and DFHDMPB, in your CICS startup job stream.

A CICS opens transaction dump data set DFHDMPA.

B CICS opens transaction dump data set DFHDMPB.

DUMPSW={NO|NEXT}
Specifies whether you want CICS to switch automatically to the next dump data set when the first is full.

NO
Disables the CICS autoswitch facility. If the transaction dump data set opened during initialization becomes full, CICS issues a console message to notify the operator. If you want to switch to the alternate data set, you must do so manually using the CEMT or EXEC CICS SET DUMPDS SWITCH command.

NEXT

Enables the autoswitch facility to switch to the next data set at end of file of the data set opened during initialization. Coding NEXT permits one switch only. If you want to switch to the alternate data set again, you must do so manually using CEMT or EXEC CICS SET DUMPDS SWITCH command. If you specify NEXT, code DLBL statements for both of the transaction dump data sets, DFHDMPA and DFHDMPB, in your CICS startup job stream.

For more information about transaction dump data sets, see page 164.

ECDSASZE={<u>0K</u>|number}

Specifies the size of the ECDSA. The default size is 0 indicating that the DSA size can be changed dynamically. A non-zero value indicates that the DSA size is fixed.

number

Specify number as an amount of storage in the range 0 to 1073741824 bytes in multiples of 1048576 bytes (1MB). If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4194304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

EDSALIM={<u>20M</u>|number}

Specifies the upper limit of the total amount of storage within which CICS can allocate the individual extended dynamic storage areas (EDSAs) that reside above the 16MB boundary.

<u>20M</u>

The default EDSA limit is 20MB (20 971 520 bytes).

number

Specify *number* as a value in the range 10MB to 2047MB, in multiples of 1MB. If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify *number* in bytes (for example, 33 554 432), or as a whole number of kilobytes (for example, 32 768K), or a whole number of megabytes (for example, 32M).

The maximum value allowed depends on a number of factors, such as:

- The size of the CICS partition
- How much storage you require for the CICS internal trace table.
- How much private storage you must leave free to satisfy VSE GETVIS requests for storage above the 16MB boundary outside the DSAs.

From the storage value that you specify on the EDSALIM parameter, CICS allocates the following extended dynamic storage areas:

The extended user DSA (EUDSA)

The user-key storage area for all user-key task-lifetime storage above the 16MB boundary.

The extended read-only DSA (ERDSA)
The key-0 storage area for all reentrant programs and tables above the 16MB boundary.

The extended shared DSA (ESDSA)
The user-key storage area for any non-reentrant user-key RMODE(ANY) programs, and also for any storage obtained by programs issuing CICS GETMAIN commands for storage above the 16MB boundary with the SHARED option.

The extended CICS DSA (ECDSA).
The CICS-key storage area for all non-reentrant CICS-key RMODE(ANY) programs, all CICS-key task-lifetime storage above the 16MB boundary, and CICS control blocks that reside above the 16MB boundary.

CICS allocates all the DSAs above the 16MB boundary in multiples of 1MB.

Note: For information about calculating the amount of storage to specify on the EDSALIM parameter, see the *CICS Performance Guide*. For example, the value that you specify must allow for all temporary storage MAIN requests, for which CICS uses the ECDSA.

EODI={E0|xx}
Specifies the end-of-data indicator for input from sequential devices. The characters “xx” represent two hexadecimal digits in the range 01 through FF. The default value is `X'E0'`, which represents the standard EBCDIC backslash symbol (\).

ERDSASZE={0K|number}
Specifies the size of the ERDSA. The default size is 0 indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

number
Specify number as an amount of storage in the range 0 to 1073741824 bytes in multiples of 1048576 bytes (1MB). If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4194304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4MB).

ESDSASZE={0K|number}
Specifies the size of the ESDSA. The default size is 0 indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

number
Specify number as an amount of storage in the range 0 to 1073741824 bytes in multiples of 1048576 bytes (1MB). If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4194304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

ESMEXITS={NOINSTLN|INSTLN}

Specifies whether installation data is to be passed via the RACROUTE interface to the external security manager (ESM) for use in exits written for the ESM.

NOINSTLN

Specifies that the INSTLN parameter is not used in RACROUTE macros.

INSTLN

Specifies that CICS-related and installation-supplied data is passed to the ESM using the INSTLN parameter of the RACROUTE macro. For programming information, including the format of the data passed, see the *CICS Customization Guide*. This data is intended for use in exits written for the ESM.

Restrictions

You can code the ESMEXITS parameter in the SIT only.

EUDSASZE={0K|number}

Specifies the size of the EUDSA. The default size is 0 indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

number

Specify number as an amount of storage in the range 0 to 1073741824 bytes in multiples of 1048576 bytes (1MB). If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4194304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

FCT={YES|xx|NO}

Specifies the suffix of the file control table (FCT) to be used.

This parameter is effective only on a CICS cold start. CICS does not load an FCT on a warm or emergency restart, and all file resource definitions are recovered from the CICS global catalog.

For information about coding the macros for this table, see the *CICS Resource Definition Guide*.

You can use a mixture of macro definitions and RDO definitions for files in your CICS region. However, your FCT should not contain definitions for files that are also defined in an RDO group specified in the group list for the CICS startup, and the converse. In a cold start, CICS first loads the file definitions from the FCT, and adds any files that are defined in RDO groups later (using the groups specified in the GRPLIST parameter). If CICS tries to install a file definition from a group list for a file already defined by an entry from the FCT, the install may fail because of conflicting file attributes. You can avoid this potential conflict by ensuring that you do not have duplicate file entries in the FCT and in your group list.

FEPI={NO|YES}

Specifies whether or not you want to use the Front End Programming Interface feature (FEPI).

NO

Means that FEPI support is not required. You should specify NO on this parameter (or allow it to default) if you do not have the feature installed, or if you do not require FEPI support.

YES

Means you require FEPI support, and causes CICS to start the CSZI transaction.

This book does not contain any information about the installation process for the Front End Programming Interface feature. Installation information can be found in the *CICS Front End Programming Interface User's Guide*.

FLDSEP={'____'|'xxxx'

Specifies 1 through 4 field-separator characters, each of which indicates end of field in the terminal input data. The default is four blanks.

The field separator allows you to use transaction identifications of less than four characters followed by one of the separator characters. When less than four characters are coded, the parameter is padded with blanks, so that the blank is then a field separator. None of the specified field separator characters should be part of a transaction identification; in particular, the use of alphabetic characters as field separators is not recommended.

The character specified in the FLDSEP parameter must not be the same as any character specified in the FLDSTRT parameter. This means that it is invalid to allow both parameters to take the default value.

Restrictions

If you specify FLDSEP in the SIT, the characters must be enclosed in single quotation marks.

If you specify FLDSEP as a PARM, SYSIPT, or CONSOLE parameter, do ***not*** enclose the characters in quotation marks, and the characters you choose must not include an embedded blank, or any of these characters:

( ) ' = ,

FLDSTRT={'_'|'x'}

Specifies a single character to be the field-name-start character for free-form input for built-in functions. The default is a blank.

The character specified should not be part of a transaction identification; in particular, the use of alphabetic characters is not recommended.

The character specified in the FLDSTRT parameter must not be the same as any character specified in the FLDSEP parameter. This means that it is invalid to allow both parameters to take the default value.

Restrictions

If you specify FLDSTRT in the SIT, the parameter must be enclosed in single quotation marks.

If you specify FLDSTRT as a PARM, SYSIPT, or CONSOLE parameter, do ***not*** enclose the character in quotation marks, and the character you choose must not be a blank or any of the following characters:

( ) ' = ,

FSSTAFF={YES|NO}

specify this parameter in an application-owning region (AOR) to prevent transactions initiated by function-shipped EXEC CICS START requests being started against incorrect terminals.

You may need to code the function-shipped START affinity (FSSTAFF) parameter in an AOR if all of the following are true:

1. The AOR is connected to two or more terminal-owning regions (TORs) that use the same, or a similar, set of terminal identifiers.
2. One or more of the TORs issues EXEC CICS START requests for transactions in the AOR.
3. The START requests are associated with terminals.
4. You are using shippable terminals, rather than statically defining remote terminals in the AOR.

Consider the following scenario:

Terminal-owning region TOR1 issues an EXEC CICS START request for transaction TRAR, which is owned by region AOR1. It is to be run against terminal T001. Meanwhile, terminal T001 on region TOR2 has been transaction routing to AOR1; a definition of T001 has been shipped to AOR1 from TOR2. When the START request arrives at AOR1, it is shipped to TOR2, rather than TOR1, for transaction routing from terminal T001.

To prevent this situation, code `YES` on the FSSTAFF parameter in the AOR.

YES
: When a START request is received from a terminal-owning region, and a shipped definition for the terminal named on the request is already installed in the AOR, the request is always shipped back to a TOR, for routing, *across the link it was received on*, unless the request supplies a TOR_NETNAME and a remote terminal with the correct TOR_NETNAME is located.

  If the TOR to which the START request is returned is **not** the one referenced in the installed remote terminal definition, a definition of the terminal is shipped to the AOR, and the autoinstall user program is called. Your autoinstall user program can then allocate an *alias* termid in the AOR, to avoid a conflict with the previously installed remote definition. For information about writing an autoinstall program to control the installation of shipped definitions, see the *CICS Customization Guide*.

NO
: When a START request is received from a terminal-owning region, and a shipped definition for the named terminal is already installed in the AOR, the request is shipped to the TOR referenced in the definition, for routing.

Notes:

1. FSSTAFF has no effect:
   - On statically-defined (hard-coded) remote terminal definitions in the AOR. If you use these, START requests are always shipped to the TORs referenced in the definitions.
   - On START requests issued in the local region. It affects only START requests shipped from other regions.

- When coded on intermediate regions in a transaction-routing path. It is effective only when coded on an application-owning region.

2. If the AOR contains no remote definition of a terminal named on a shipped START request, the “terminal not known” global user exits, XICTENF and XALTENF, are called. For details of these exits, see the *CICS Customization Guide*.

GMTEXT={‘WELCOME TO CICS’|‘text’}
Specifies whether the default logon message text (WELCOME TO CICS) or your own message text is to be displayed on the screen by the CSGM (good morning) transaction when a terminal is logged on to CICS through VTAM, by the CESN transaction if used to sign on to CICS, or by your own transactions using the EXEC CICS INQUIRE SYSTEM GMMTEXT command.

You can use apostrophes to punctuate your message, in addition to using them as message delimiters. However, you must code *two* successive apostrophes to represent a single apostrophe in your text. For example,

```
GMTEXT='User''s logon message text.'
```

The whole message must still be enclosed by a pair of single delimiting apostrophes.

Your message text can be from 1 through 246 characters (bytes), and can extend over two lines by extending the text to column 80 on the first line, and continuing in column 1 of the second line. For example, the following might be used in the SYSIPT data set:

```
GMTEXT='An Information Development CICS Terminal-Owning Region (TOR) - CICSIDC.
This message is to show the use of continuation lines when creating a GMTEXT par
ameter in the SYSIPT data set'
```

The CSGM transaction displays this as follows (with the time appended to the end of message):

```
An Information Development CICS Terminal-Owning Region (TOR) - CICSIDC. This me
ssage is to show the use of continuation lines when creating a GMTEXT parameter
in the SYSIPT data set 09:56:14
```

The CESN transaction displays this as follows:

```
               Signon to CICS                                APPLID CICSHTH1

An Information Development CICS Terminal-Owning Region (TOR) - CICSIDC.
This message is to show the use of continuation lines when creating a GMTEXT
parameter in the SYSIPT data set
```

For any transaction other than CESN that displays the text specified by this parameter, you must use a RDO TYPETERM with LOGONMSG(YES) for all terminals requiring the logon message. For information about using RDO TYPETERM, see the *CICS Resource Definition Guide*.

GMTRAN={CSGM|CESN|transaction-id}
Specifies the name of the transaction that is initiated when terminals are logged on to CICS by VTAM. Do not specify the name of a remote transaction. The transaction must be capable of being automatically initiated (ATI). The default is the transaction CSGM, that displays the text specified in the GMTEXT

parameter. Alternatively, you can specify the CICS signon transaction, CESN, which also displays the text specified in the GMTEXT parameter. The GMTRAN parameter can be used with the LGNMSG parameter to retrieve VTAM logon data.

GNTRAN={<u>CESF</u>|transaction_id}
Specifies the transaction that you want CICS to invoke when a user's terminal-timeout period expires.

<u>CESF</u>
The default, CESF, is the basic CICS signoff transaction without any options. This transaction attempts to sign off a terminal, subject to the SIGNOFF attribute of the RDO TYPETERM resource definition for that terminal.

transaction_id
The name of an alternative timeout transaction to signoff the user at the timed-out terminal. Specifying your own transaction allows you to specify functions in addition to, or instead of, signoff. For example, your own transaction could issue a prompt for the terminal user's password, and allow the session to continue if the correct password is entered.

The transaction to be used must have been specially written to handle the GNTRAN commarea that is passed to it. Of the CICS-supplied transactions, only CESF has been written to handle the GNTRAN commarea. For more information about writing your own transactions for GNTRAN, see the *CICS Customization Guide*.

Note: When either the default CESF transaction, or your own transaction, attempts to sign off a terminal, the result is subject to the SIGNOFF attribute of the RDO TYPETERM resource definition for the terminal, as follows:

SIGNOFF	Effect
YES	The terminal is signed off, but not logged off.
NO	The terminal remains signed on and logged on.
LOGOFF	The terminal is both signed off and logged off.

Note: If GNTRAN fails to attach, and SIGNOFF(LOGOFF) has been specified, the terminal which has reached timeout will be signed off and logged off. GNTRAN will not run and will have no effect.

GRPLIST={<u>DFHLIST</u>|name|(name[,name2][,name3][,name4])}
Specifies the names (each 1 through 8 characters) of up to four lists of resource definition groups on the CICS system definition (CSD) file. The resource definitions in all the groups in the specified lists are loaded during initialization when CICS performs a cold start. If a warm or emergency start is performed, the resource definitions are derived from the CICS global catalog, and the GRPLIST parameter is ignored.

Each name can be either a real group list name or a generic group list name that incorporates global filename characters (+ and *). If you specify more than one group list (either by specifically coding two or more group list names or by coding a group list name with global filename characters), the later group lists are concatenated onto the first group list. Any duplicate resource definitions in later group lists override those in earlier group lists.

Use the CEDA command LOCK to protect the lists of resource groups specified on the GRPLIST parameter.

The default is DFHLIST, the CICS-supplied list that specifies the set of resource definitions needed by CICS. If you create your own group list, either add to it the groups specified in DFHLIST (omitting only those for CICS functions that you know you do not need) or specify the DFHLIST name on the GRPLIST parameter. Do not code GRPLIST=NO unless you have a group list named NO.

Notes:

1. Group lists specified by a generic group list name are concatenated in alphabetic then numeric order. For example, the generic list name CICSHT*, would concatenate the group lists CICSHT#1, CICSHTAP, CICSHTSD, and CICSHT3V in that order. If the order of concatenation is important (for example, to ensure that a particular resource definition overrides another), you should consider coding real group list names.
2. If a group list contains resource definitions that are needed by another group list, the prequisite group list must be installed first. For example, if list A has TYPETERM definitions needed for TERMINAL definitions in list B, list A must be installed first. This may mean that you have to specifically name the prerequisite group on the GRPLIST parameter.
3. Take care when using generic group list names because, if a group list on your CSD satisfies the generic name, it will be installed. This means that a group list can be installed more than once; for example, if you specify the real group list name and a generic group list name that it satisfies, or if you specify two generic group list names that the group list name satisfies.
4. To override one or more of the group lists specified on the GRPLIST system initialization parameter, you must specify all list names (both real and generic) that you want to use, even if you are not changing the names.

For example, if you want to use the four group lists CICSHT#1, CICSHTAP, CICSHT3V, and CICSHTSD, you could specify either of the following system initialization parameters:

```
GRPLIST=(CICSHT#1,CICSHTAP,CICSHT3V,CICSHTSD)
GRPLIST=(CICSHT*)
```

In the first example GRPLIST, the group lists are loaded in the order specified, and resource definitions installed from the CICSHTSD group list will override any duplicate definitions installed by the other groups.

In the second example GRPLIST, the group lists are loaded in the order CICSHT#1, CICSHTAP, CICSHTSD, then CICSHT3V, and resource definitions installed from the CICSHT3V group list will override any duplicate definitions installed by the other groups.

If your SIT contains the parameter:

```
GRPLIST=(CICSHT#1,CICSAP*,CICSHT3V,CICSHTSD)
```

and you want to replace the list CICSHT3V with the list ANOLST05, you should specify the override:

```
GRPLIST=(CICSHT#1,CICSAP*,ANOLST05,CICSHTSD)
```

In general, any required resource definitions should appear in *one of* the group lists specified on the GRPLIST system initialization parameter.

For information about resource definitions, groups, lists, and the CSD, see the *CICS Resource Definition Guide*.

ICP=COLD
Code this parameter if you want to cold start the interval control program, for more information see page 205.

ICV={<u>1000</u>|number}
Code this parameter with the region exit time interval in milliseconds. The ICV system initialization parameter specifies the maximum time in milliseconds that CICS releases control to the operating system when there are no transactions ready to resume processing. This time interval can be any integer in the range 100 through 3600000 milliseconds (specifying an interval up to 60 minutes). A typical range of operation might be 100 through 2000 milliseconds.

A low value interval can enable much of the CICS nucleus to be retained in dynamic storage, and not be paged-out at times of low terminal activity. This reduces the amount of dynamic storage paging necessary for CICS to process terminal transactions (thus representing a potential reduction in response time), sometimes at the expense of concurrent batch region throughput. Large networks with high terminal activity are inclined to run CICS without a need for this value, except to handle the occasional, but unpredictable, period of inactivity. These networks can usually function with a large interval (10000 to 3600000 milliseconds). Once a task has been initiated, its requests for terminal services and the completion of the services are recognized by the system and this maximum delay interval is overridden.

Small systems, or those with low terminal activity, are subject to paging introduced by other jobs running in competition with CICS. By specifying a low value interval, key portions of the CICS nucleus are referenced more frequently, thus reducing the probability of these pages being paged-out. However, the execution of the logic without performing productive work might be considered wasteful. The need to increase the probability of residency by frequent but unproductive referencing must be weighed against the overhead and response time degradation incurred by allowing the paging to occur. By increasing the interval size, less unproductive work is performed at the expense of performance if paging occurs during the periods of CICS activity. For information about the effect of ICV on performance, see the *CICS Performance Guide*.

Note: The region exit time interval process contains a mechanism to ensure that CICS does not constantly set and cancel timers (thus degrading performance) while attempting to meet its objectives for a low region exit time interval. This mechanism can cause CICS to release control to the operating system for up to 0.5 seconds when the interval has been set at <u>less</u> than 250; and up to 0.25 seconds more than the region exit time interval when the interval has been set <u>greater</u> than 250.

ICVR={<u>5000</u>|number}
Specifies the default runaway task time interval in milliseconds as a decimal number. You can code zero, or a number in the range 500 through 2 700 000, in multiples of 500. CICS rounds down values that are not multiples of 500. This is the RUNAWAY interval used by transactions defined with RUNAWAY(SYSTEM) (see the *CICS Resource Definition Guide*). CICS may purge a task if it has not given up control after the RUNAWAY interval for the transaction (or ICVR if the transaction definition specified

RUNAWAY(SYSTEM)). If you code ICVR=0, runaway task control is inoperative for transactions specifying RUNAWAY(SYSTEM) in their RDO TRANSACTION definition (that is, tasks do not get purged if they appear to be looping). The ICVR value is independent of the ICV value, and can be less than the ICV value. Note that CICS runaway task detection is based upon task time, that is, the interval is decremented only when the task has control of the processor. For information about commands that reinitialize the ICVR value, see the *CICS Problem Determination Guide*.

ICVTSD={<u>500</u>|number}
The terminal scan delay facility determines how quickly CICS deals with some terminal I/O requests made by applications. The range is 0 through 5000 milliseconds, with a default of ICVTSD=500.

There is an overhead in dealing with such requests. By specifying a nonzero value, the overhead may be spread over several transactions. A value close to zero (for example 200) would be adequate.

INITPARM=(pgmname_1=‘parmstring_1’[, ... ,pgmname_n=‘parmstring_n’])
Code this parameter in order to pass parameters to application programs that use the EXEC CICS ASSIGN INITPARM command. For example, you can use INITPARM to pass parameters to PLTPI programs to be executed in the final stages of system initialization. The area giving access to the parameters is specified by the EXEC CICS ASSIGN INITPARM command. For programming information about the EXEC CICS ASSIGN INITPARM command, see the *CICS Application Programming Reference* manual.

pgmname
Code pgmname with the name of a program. This name must be 1 through 8 alphanumeric or national language characters.

parmstring
Code parmstring with the parameter string (up to 60 characters enclosed by single quotation marks) to be passed to the associated program. Any quotation marks imbedded in the string must be duplicated. For information on coding INITPARM in the SYSIPT data set, see “Rules for coding CICS system initialization parameters in the SYSIPT data set” on page 204.

You can specify up to 255 pgmname=‘parmstring’ sets.

INTTR={<u>ON</u>|OFF}
Code this parameter to specify whether the internal CICS trace destination is to be activated at system initialization.

This parameter controls whether any of the three types of CICS trace entry are written to the internal trace table. The three types are: CICS system trace (see the SYSTR parameter), user trace (see the USERTR parameter), and exception trace entries (which are always made and not controlled by a system initialization parameter).

<u>ON</u>	Activate main storage trace.
OFF	Do not activate main storage trace.

IRCSTRT={<u>NO</u>|YES}
Code this parameter to indicate whether IRC is started up at system initialization. If IRCSTRT=YES is not coded, IRC can be initialized by issuing a CEMT or EXEC CICS SET IRC OPEN command.

ISC={<u>NO</u>|YES}

Code YES to include the CICS programs required for interregion or intersystem communication.

JCT={<u>YES</u>|xx|NO}

Code this parameter with the suffix of the journal control table to be used, for more information see page 205. This indicates whether journaling and volume control are to be used. Note that you must have journaling if you code XRF=YES. If you code JCT=NO, a dummy journal control program (DFHJCPDY) is loaded.

For information about coding the macros for this table, see the *CICS Resource Definition Guide*.

JSTATUS=RESET

Code JSTATUS=RESET to reset the journal status of all journal data sets that are defined on disk (JTYPE=DISK1|DISK2), but are ***not*** specified with JOUROPT=AUTOARCH.

This parameter resets the journal status held in the global catalog to “ready for use”. Before using the JSTATUS option at startup, ensure that any disk journals that contain essential information are archived, either to tape or another disk data set.

Restrictions

You can code the JSTATUS parameter in PARM, SYSIPT, or CONSOLE only. It is not applicable if you are using the CICS automatic archiving facility.

LGNMSG={<u>NO</u>|YES}

Code this to indicate whether VTAM logon data is made available to an application program.

<u>NO</u>

VTAM logon data is not available to an application program.

YES

VTAM logon data is available to an application program. The data can be retrieved with an EXEC CICS EXTRACT LOGONMSG command. For programming information about this command, see the *CICS Application Programming Reference* manual.

You can use this parameter with the GMTRAN parameter to retrieve the VTAM logon data at the time a terminal is logged on to CICS by VTAM.

MCT={<u>NO</u>|YES|xx}

Code this parameter to specify the monitoring control table suffix, for more information see page 205.

If you specify MCT=NO, CICS monitoring builds dynamically a default MCT, ensuring that default monitoring control table entries are always available for use when monitoring is on and a monitoring class (or classes) is active.

For information about coding the macros for this table, see the *CICS Resource Definition Guide*.

MN={<u>OFF</u>|ON}

Code this parameter to indicate whether monitoring is to be switched on or off at initialization, and use the individual monitoring class parameters to control which monitoring classes are to be active. (See the MNEXC, and MNPER parameter descriptions.) The default status is that the CICS monitoring facility is ***off***. The monitoring status is recorded in the CICS global catalog for use during warm and emergency restarts.

<u>OFF</u>	Switch off monitoring.
ON	Switch on monitoring. However, unless at least one individual class is active, no monitoring records are written. For details of the effect of monitoring status being on or off, in conjunction with the status of the various monitoring classes, see the following notes:

Note: If the monitoring status is ON, CICS accumulates monitoring data continuously. For the performance and exception monitoring classes, CICS writes the monitoring data for each class that is active to a CICS Data Management Facility (DMF) data set.

If the monitoring status is OFF, CICS does not accumulate or write any monitoring data, even if any of the monitoring classes are active.

You can change the monitoring status and the monitoring class settings at any time, as follows:

- During a warm restart by coding an MN system initialization parameter in PARM, SYSIPT, or through the system console.
- While CICS is running, by either of:
  - The CEMT SET MONITOR command
  - The EXEC CICS SET MONITOR command.

When you change the status of monitoring, the change takes effect immediately. If you change the monitoring status from OFF to ON, monitoring starts to accumulate data and write monitoring records to DMF for all tasks that start after the status change is made ***for all active monitoring classes***.

If the status is changed from ON to OFF, monitoring stops writing records immediately and does not accumulate monitoring data for any tasks that start after the status change is made.

The monitoring status operand can be manipulated independently of the class settings. This means that, even if the monitoring status is OFF, you can change the monitoring class settings and the changes take effect for all tasks that are started after the monitoring status is next set to ON.

For programming information about controlling CICS monitoring, see the *CICS System Programming Reference* manual.

MNCONV={<u>NO</u>|YES}

Specifies whether or not conversational tasks are to have separate performance class records produced for each pair of terminal control I/O requests.

Any clock (including user-defined) that is active at the time such a performance class record is produced is stopped immediately before the record is written. After the record is written, such a clock is reset to zero and restarted. Thus a clock whose activity spans more than one recording interval within the conversational task appears in multiple records, each showing part of the time, and the parts adding up to the total time the clock is active. The high-water-mark fields (which record maximum levels of storage used) are reset to their current values. All other fields are set to X'00', except for the key fields (transid, termid). The monitoring converse status is recorded in the CICS global catalog for use during warm and emergency restarts.

MNEXC={<u>OFF</u>|ON}
Code this parameter to indicate whether the monitoring exception class is to be made active during initialization. The monitoring exception class status is recorded in the CICS global catalog for use during warm and emergency restarts.

<u>OFF</u>	Set the exception monitoring class to “not active.”
ON	Set the exception monitoring class to “active.”

For programming information about exception monitoring records, see the *CICS Customization Guide*.

MNFREQ={<u>0</u>|hhmmss}
Specifies the interval for which CICS automatically produces a transaction performance class record for any long-running transaction. The monitoring frequency value is recorded in the CICS global catalog for use during warm and emergency restarts.

0 means that no frequency monitoring is active.

hhmmss
is the interval for which monitoring produces automatically a transaction performance class record for any long-running transaction. Specify a 1 to 6 digit number in the range 001500–240000. Numbers that are fewer than six digits are padded with leading zeroes.

MNPER={<u>OFF</u>|ON}
Code this parameter to indicate whether the monitoring performance class is to be made active during CICS initialization. The monitoring performance class status is recorded in the CICS global catalog for use during warm and emergency restarts.

<u>OFF</u>	Set the performance monitoring class to “not active.”
ON	Set the performance monitoring class to “active.”

For programming information about performance monitoring records, see the *CICS Customization Guide*.

MNSYNC={<u>NO</u>|YES}
Specifies whether or not you want CICS to produce a transaction performance class record when a transaction takes an implicit or explicit syncpoint (unit-of-work). No action is taken for syncpoint rollbacks. The monitoring syncpoint status is recorded in the CICS global catalog for use during warm and emergency restarts.

MNTIME={<u>GMT</u>|LOCAL}
Specifies whether you want the time stamp fields in the performance class monitoring data to be returned to an application using the EXEC CICS COLLECT STATISTICS MONITOR(taskno) command in either GMT or local

time. The monitoring time value is recorded in the CICS global catalog for use during warm and emergency restarts.

For programming information on the EXEC CICS COLLECT STATISTICS command, see the *CICS System Programming Reference*.

MROBTCH={1|number}
Code this parameter to specify the number of events that must occur before CICS is posted for dispatch due to the batching mechanism. The number can be in the range 1 through 255, and the default is 1.

Use this batching mechanism to spread the overhead of dispatching CICS over several tasks. If the value is greater than 1 and CICS is in a system wait, CICS is not posted for dispatch until the specified number of events has occurred. Events include MRO requests from connected systems or DASD I/O. For these events, CICS is dispatched as soon as one of the following occurs:

- The current batch fills up (the number of events equals MROBTCH).
- An ICV interval expires.

Therefore, ensure that the time interval you specify in the ICV parameter is low enough to prevent undue delay to the system.

If CICS is dispatched for another reason, the current batch is dealt with in that dispatch of CICS.

Note: During periods of low utilization, a value greater than 1 specified on the MROBTCH parameter may result in increased transaction response times. Transactions issuing file I/O requests may be delayed due to increased FCIOWAIT. For further guidance information about the effect of MROBTCH on performance, see the *CICS Performance Guide*.

MROFSE={NO|YES}
specifies whether you want to extend the lifetime of the long-running mirror to keep it allocated until the end of the task rather than after a user syncpoint for Function Shipping applications.

NO The lifetime of the MRO long-running mirror is not extended.

YES
The mirror task remains available to the application until the end of the application's task. This extended long-running mirror saves the overhead of re-attaching the mirror task following a user syncpoint.

This parameter is ignored for DPL requests (that is a DPL causes the session to be freed at the next syncpoint even if is has been kept for a previous sequence of syncpoints).

It should be used with caution. For additional information, see the Long Running Mirror sections of the *CICS Intercommunication Guide* and the *CICS Performance Guide*.

MROLRM={NO|YES}
Code this parameter to specify whether you want to establish an MRO long-running mirror task.

NO
The MRO long-running mirror task is not required.

YES
The mirror transaction remains available to the application issuing the remote request. This long-running mirror saves the overhead of re-establishing communication with the mirror transaction if the application makes more function shipping requests in this unit of work.

For information about long-running mirror tasks, see the *CICS Intercommunication Guide*.

MSGCASE={<u>MIXED</u>|UPPER}
CICS messages handled by the CICS message domain are in mixed case. Code the MSGCASE parameter to indicate how you want the message domain to display these mixed case messages.

<u>MIXED</u> This is the default in the SIT; all messages displayed by the CICS message domain remain in mixed case,

UPPER The message domain displays all mixed case messages in uppercase only.

Note: Mixed case output is not displayed correctly on Katakana display terminals and printers. Uppercase English characters appear correctly as uppercase English characters, but lowercase appears as Katakana symbols. If you have any Katakana terminals connected to your CICS region, specify MSGCASE=UPPER.

MSGLVL={<u>1</u>|0}
Code this parameter with the message level that controls the generation of messages to the console.

<u>1</u> All messages are to be printed.

0 Only critical errors or interactive messages are to be printed.

MXT={<u>5</u>|number}
Specifies the maximum number, in the range 1 through 999, of ***user*** tasks CICS allows to exist at any time. CICS queues requests for tasks above this number but does not action (attach) them until the number of tasks attached drops below the MXT limit.

Note: The MXT value does **not** include CICS system tasks.

NATLANG=(<u>E</u>,x,y,z,...)
Specify on this parameter the single-character codes for the languages to be supported in this CICS run. For more information see “Installing definitions for national language support (NLS)” on page 145.

<u>E</u> Code E for English, which is the *system* default (that is, is provided even if you do not specifically code E).

x,y,z,... Code the appropriate letters for the other supported languages that you require.

For the codes that you specify on this parameter, you must ensure that a DFHMET1x module (where x is the language code) is in a sublibrary in the LIBDEF PHASE,SEARCH chain for the CICS job.

English language support is provided, even if you do not specifically code E for English.

The first language code specifies the default language for those elements of CICS enabled to receive NLS messages, such as some destinations used for CICS messages, and the terminals or users not signed-on with an NLS code. The other language codes are provided to specify the language to be used for messages sent to terminals that are defined with the appropriate language support code.

For example, coding NATLANG=(G,C,K) has the same effect as coding NATLANG=(G,C,E,K); that is, in both cases the default NLS language is German (G), and the languages English, Chinese (C), and Kanji (K) are supported.

CICS console messages are available in English and German only.

NEWSIT={YES|NO}
Specify NEWSIT=YES to cause CICS to load the specified SIT, and enforce the use of all system initialization parameters, modified by any system initialization parameters provided via PARM, SYSIPT, or the system console, even in a warm start. Enforcing the use of system initialization parameters in this way overrides any parameters that may have been stored in a warm keypoint at shutdown.

However, there are some exceptions; the following system initialization parameters are always ignored in a warm start, even if they are supplied via PARM, SYSIPT, or the console:

CSDACC
CSDBUFND
CSDBUFNI
CSDFRLOG
CSDJID
CSDLSRNO
CSDRECOV
CSDSTRNO
FCT
GRPLIST

In a warm restart CICS uses the ***installed*** resource definitions saved in the CICS global catalog at warm shutdown, and therefore the CSD, FCT, and GRPLIST parameters are ignored. (At CICS startup, you can only modify installed resource definitions, including file control table entries, or change to a new FCT, by performing a cold start of CICS with START=COLD.)

For more information about the use of the NEWSIT parameter, see “Classes of start and restart” on page 209.

Restrictions

You can code the NEWSIT parameter in PARM, SYSIPT, or CONSOLE only.

OPERTIM={120|number}
Code this parameter with the write-to-operator timeout value, in the range 0 through 86400 seconds (24 hours). This is the maximum time (in seconds) that CICS waits for a reply before returning control to this transaction. For information about using the write-to-operator timeout value, see the *CICS Application Programming Reference* manual.

PARMERR={<u>INTERACT</u>|IGNORE|ABEND}

Code this parameter to specify what action you want to follow if CICS detects incorrect system initialization parameter overrides during initialization.

Note: When specified as an override, this parameter affects only subsequent system initialization parameter overrides. Errors in earlier system initialization parameter overrides are dealt with according to the PARMERR system initialization parameter value in the SIT.

<u>INTERACT</u>

Enables the operator to communicate with CICS via the console and correct parameter errors.

Note: INTERACT is overridden with IGNORE in the following cases:

- If errors are found in PARM or SYSIPT for system initialization parameter overrides that are not allowed to be entered from the console
- In certain circumstances, in response to invalid data when you have been trying to correct a previous invalid system initialization parameter keyword or value.

IGNORE

CICS ignores errors, and tries to complete initialization.

ABEND

CICS abends.

PDI={<u>30</u>|decimal-value}

Use this parameter in a SIT for an active CICS region. It specifies the XRF primary delay interval, in seconds. The minimum delay that you can specify is 5 seconds. This is the time that must elapse between the (apparent) loss of the surveillance signal in the alternate CICS region, and any reaction by the active CICS region. The corresponding parameter for the alternate CICS region is ADI. PDI and ADI need not have the same value.

PGAICTLG={<u>MODIFY</u>|NONE|ALL}

Specifies whether autoinstalled program definitions should be cataloged. While CICS is running, you can set whether autoinstalled programs should be cataloged dynamically, by using either the EXEC CICS SET SYSTEM or CEMT SET SYSTEM command.

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 CICS global catalog (DFHGCD). Definitions are autoinstalled on first reference.

ALL

Autoinstalled program definitions are written to the global catalog at the time of the autoinstall, and following any subsequent modification.

PGAIEXIT={DFHPGADX|name}
Specifies the name of the program autoinstall exit program. While CICS is running, you can set the name of the program autoinstall exit program dynamically, by using either the EXEC CICS SET SYSTEM or CEMT SET SYSTEM command.

PGAIPGM={INACTIVE|ACTIVE}
Specifies the state of the program autoinstall function at initialization. While CICS is running, you can set the status of program autoinstall dynamically, by using either the EXEC CICS SET SYSTEM or CEMT SET SYSTEM command.

INACTIVE
The program autoinstall function is disabled.

ACTIVE
The program autoinstall function is enabled.

PGCHAIN=character(s)
Code this parameter with the character string that is identified by terminal control as a BMS terminal page-chaining command. It can be 1 through 7 characters. For more information about the character string, see the notes under the PGRET parameter.

PGCOPY=character(s)
Code this parameter with the character string that is identified by terminal control as a BMS command to copy output from one terminal to another. It can be 1 through 7 characters. For more information about the character string, see the notes under the PGRET parameter.

PGPURGE=character(s)
Code this parameter with the character string that is identified by terminal control as a BMS terminal page-purge command. It can be 1 through 7 characters. For more information about the character string, see the notes under the PGRET parameter.

PGRET=character(s)
The character string that is recognized by terminal control as a BMS terminal page-retrieval command. It can be 1 through 7 characters.

Notes:

1. Each character string is unique with respect to the leading characters of every other transaction identification defined in the CSD. A command requested by a single character precludes the use of all other transaction identifications starting with this character.
2. In pseudoconversational mode, each character string is unique with respect to the leading characters of any terminal input message.
3. A field-separator or other suitable delimiter may be specified in each character string to separate this command code from the remainder of the paging command when entered by an operator. For example:

```
PGCHAIN    =  X/
PGCOPY     =  C/
PGPURGE    =  T/
PGRET      =  P/
```

   This reduces the risk of creating a nonunique command. (See Note 1.)

Restrictions

If you specify PGCHAIN, PGCOPY, PGPURGE, or PGRET in the SIT, the characters you choose must not include any of the following: ( ) '

If you specify PGCHAIN, PGCOPY, PGPURGE, or PGRET as a PARM, SYSIPT, or console parameter, do not enclose the characters in quotation marks. The characters you choose must not include an embedded blank or any of the following: ( ) ' =

4. PGCHAIN, PGCOPY, PGPURGE, and PGRET are required only if full function BMS is being used. For information about the BMS page retrieval transaction CSPG, see the *CICS-Supplied Transactions* manual.
5. CICS always processes a paging command entered by the operator before initiating a transaction in response to a macro request, that consists of either DFHBMS or DFHPC with the TRANSID operand coded.

PLTPI={<u>NO</u>|xx|YES}

Code this parameter with a program list table that contains a list of programs to be executed in the final stages of system initialization. For more information see page 205.

You can use the system initialization parameter INITPARM to pass parameters to those programs.

For information about coding the macros for this table, see the manual.

PLTPISEC={<u>NONE</u>|CMDSEC|RESSEC|ALL}

Specifies whether or not you want CICS to perform command security or resource security checking for PLT programs during CICS initialization. The PLT programs run under the authority of the userid specified on PLTPIUSR, which must be authorized to the appropriate resources defined by PLTPISEC.

<u>NONE</u>

Specifies that you do not want any security checking on on PLT initialization programs.

CMDSEC

Specifies that you want CICS to perform command security checking only.

RESSEC

Specifies that you want CICS to perform resource security checking only.

ALL

Specifies that you want CICS to perform both command and resource security checking.

Restrictions

You can code the PLTPISEC parameter in the SIT, PARM, or SYSIPT only.

PLTPIUSR=userid

Specifies the userid that CICS is to use for security checking for PLT programs that run during CICS initialization. All PLT programs run under the authority of the specified userid, which must be authorized to all the resources referenced by the programs, as defined by the PLTPISEC parameter.

PLT programs are run under the CICS internal transaction, CPLT. Before the CPLT transaction is attached, CICS performs a surrogate user check against the CICS region userid (the userid under which the CICS region is executing). This is to ensure that the CICS region is authorized as a surrogate for the userid specified on the PLTPIUSR parameter. This ensures that you cannot arbitrarily specify any PLT userid in any CICS region—each PLT userid must first be authorized to the appropriate CICS region.

If you do not specify the PLTPIUSR parameter, CICS runs PLTPI programs under the authority of the CICS region userid, in which case CICS does not perform a surrogate user check. However, the CICS region userid must be authorized to all the resources referenced by the PLT programs.

Restrictions

You can code the PLTPIUSR parameter in the SIT, PARM, or SYSIPT only.

PLTSD={<u>NO</u>|xx|YES}
Code this parameter with a program list table that contains a list of programs to be executed during system termination. For more information see page 205.

PRGDLAY={<u>0</u>|hhmm}
Code this parameter with the BMS purge delay time interval that is added to the specified delivery time to determine when a message is to be considered undeliverable and therefore purged. This time interval is coded in the form "hhmm" (where "hh" represents hours from 00 to 99 and "mm" represents minutes from 00 to 59). If PRGDLAY is not coded, or is given a zero value, a message remains eligible for delivery either until it is purged or until temporary storage is reinitialized.

Note: If you specify PRGDLAY as a SIT override, you must still specify a 4-character value (for example 0000).

The PRGDLAY facility requires the use of full function BMS. Note also that you must code a PRGDLAY value if you want the ERRTERM|ERRTERM(name) parameter on EXEC CICS ROUTE commands to be operative. For programming information about notification of undelivered messages, see the *CICS Application Programming Reference* manual.

The PRGDLAY value determines the interval between terminal page clean-up operations. A very low value causes the CSPQ transaction to be initiated continuously, and can have a detrimental effect on task-related resources.

A zero value stops CSPQ initiating terminal page clean-up. However, this can cause messages to stay in the system forever, resulting in performance problems with long AID queues or lack of temporary storage. The actual purge delay time interval specified is dependent on individual system requirements.

PRINT={<u>NO</u>|YES|PA1|PA2|PA3}
Code this parameter with the method of requesting printout of the contents of a 3270 screen.

<u>NO</u>
Screen copying is not required.

YES
Screen copying can be requested by terminal control print requests only.

PA1, PA2, or PA3

Screen copying can be requested by terminal control print request, or by using the PA (program attention) key specified.

The PA key specified by this parameter must not be specified by the TASKREQ option of the RDO TRANSACTION definition or be used for 3270 single keystroke retrieval.

When YES, PA1, PA2, or PA3 is specified, transaction CSPP is initiated which invokes program DFHP3270. The transaction and programs are defined in the CSD group DFHHARDC. In the case of 3270 and LUTYPE2 logical units, the resources defined in CSD group DFHVTAMP are required.

The 3270 print-request facility allows either the application program or the terminal operator to request a printout of data currently displayed on the 3270 display.

If CSPP is invoked to print the screen contents at an associated VTAM printer, the screen size of the printer is chosen according to the screen size defined in the profile for the transaction CSPP. The CICS-supplied definitions use the default screen size. Therefore, if you want DFHP3270 to use the alternate screen size of the printer, you must alter the screen size defined in the profile for the transaction CSPP. For information about defining profiles for transactions, see the *CICS Resource Definition Guide*.

For a VTAM 3270 display without the printer-adapter feature, the PRINT request prints the contents of the display on the first available 3270 printer specified by PRINTER and ALTPRINTER options of the RDO TERMINAL definition. For a printer to be considered available, it must be in service and not currently attached to a task. It is not necessary for the printer to be on the same control unit.

In an MRO environment, the printer must be owned by the same system as the VTAM 3270 display.

For the 3275 with the printer-adapter feature, the PRINT request prints the data currently in the 3275 display buffer on the 3284 Model 3 printer attached to the 3275.

The format of the print operation depends on the size of the display buffer. For a 40-character wide display, the print format is a 40-byte line, and for an 80-character wide display the format is an 80-byte line.

For the 3270 compatibility mode logical unit of the 3790 (if the logical unit has the printer-adapter feature specified), the PRINT request prints the contents of the display on the first printer available to the 3790. The allocation of the printer to be used is under the control of the 3790.

For 3274, 3276, and LUTYPE2 logical units with the printer-adapter feature, the PRINT request prints the contents of the display on the first printer available to the 3270 control unit. The printer to be allocated depends on the printer authorization matrix. For information, refer to the *3270 Information Display System Component Description* manual.

For the 3270 compatibility mode logical unit without the printer-adapter feature, see the preceding paragraph on VTAM 3270 displays without the printer-adapter feature.

PRTYAGE={32768|value}

Code this parameter with the number of milliseconds to be used in the priority aging algorithm for incrementing the priority of a task. The value can be in the range 0 through 65535, and 32768 is the default.

The priority aging factor is used to increase the effective priority of a task according to the amount of time it is held on a ready queue. The value represents the number of milliseconds that must elapse before the priority of a waiting task can be adjusted upwards by 1. For example, if you code PRTYAGE=3000, a task has its priority raised by 1 for every 3000 milliseconds it is held on the ready queue. Thus a high value for PRTYAGE results in a task being promoted very slowly up the priority increment range, and a low value enables a task to have its priority incremented quickly.

If you specify a value of 0, the priority aging algorithm is not used (task priorities are not modified by age) and tasks on the ready queue are handled according to the user assigned priority.

PRVMOD={name|(name,name...name)}

Code the PRVMOD parameter with the name of those modules that are not to be used from the VSE shared virtual area (SVA).

The operand is a list of 1- to 8-character module names. This enables you to use a private version of a CICS nucleus module in the CICS address space, and not a version that might be in the SVA. For more information about preventing modules from being used from the SVA, see “Preventing CICS from using modules installed in the SVA” on page 64.

Restrictions

You can code the PRVMOD parameter in PARM, SYSIPT, or CONSOLE only.

PSDINT={0|hhmmss}

Specifies the persistent session delay interval. This delay interval specifies if, and for how long, VTAM is to hold sessions in a recovery-pending state if CICS fails. The value for hours can be in the range 0 through 23; the minutes and seconds in the range 00 through 59 inclusive.

This value can be overridden during CICS execution (and hence change the action taken by VTAM if CICS fails).

0 A zero value specifies that, if CICS fails, sessions are terminated. This is the default.

hhmmss

Specifies a persistent session delay interval from 1 second up to the maximum of 23 hours 59 minutes and 59 seconds. If CICS fails, VTAM holds sessions in recovery pending state for up to the interval specified on the PSDINT system initialization parameter.

Specify a 1-to-6 digit time in hours, minutes and seconds, up to the maximum time. If you specify less than six digits, CICS pads the value with leading zeros. Thus a value of 500 is taken as five minutes exactly.

The interval you specify must cover the time from when CICS fails to when the VTAM ACB is opened by CICS during the subsequent emergency restart.

VTAM holds all sessions in recovery pending state for up to the interval specified (unless they are unbound through path failure or VTAM operator action, or other-system action in the case of intelligent LUs). The PSDINT value used must take account of the types and numbers of sessions involved.

You must exercise care when specifying large PSDINT values because of the problems they may give in some environments, in particular:

- Dial-up sessions—real costs may be incurred
- LU6.2 sessions to other host systems—such systems may become stressed.

Notes:

1. When specifying a PSDINT value, you must consider the number and, more particularly, the nature of the sessions involved. If LU6.2 sessions to other host systems are retained in recovery pending state, the other host systems may experience excessive queuing delays. This point applies to LU6.1 sessions which are retained until restart (when they are unbound).
2. The PSDINT parameter is incompatible with the XRF=YES parameter. If XRF=YES is specified, the PSDINT parameter is ignored.

PVDELAY={<u>30</u>|number}
Code this with the persistent verification delay as a value in the range 0 through 10080 minutes (up to 7 days). PVDELAY defines how long entries can remain in the signed-on-from lists for those connections for which persistent verification is specified in a connection resource definition. If you specify PVDELAY=0, entries are deleted immediately after use.

For information about the use of PVDELAY, see the *CICS Performance Guide*.

RAMAX={<u>256</u>|value}
Code this parameter with the size in bytes of the I/O area allocated for each RECEIVE ANY issued by CICS, in the range 0 through 32767 bytes.

Notes:

1. If you are using APPC, do not code a value less than 256; otherwise, the results are unpredictable.
2. If you are using pipeline terminals, do not code a value that is less than the RUSIZE (from the CINIT), because pipelines cannot handle data longer than this.

For information about coding this parameter, see the *CICS Performance Guide*.

RAPOOL={<u>50</u>|decimal-value}
Code this parameter to determine the size of the CICS receive any pool. It is the number of fixed request parameter lists (RPLs), receive any control elements (RACEs), and receive any input areas (RAIAs) that are to be generated.

Decimal-value can be in the range 1 through 999, and has a default of 2.

If you omit the RAPOOL parameter altogether, RAPOOL=50 is assumed. CICS maintains n VTAM RECEIVE ANYs, where n is either the RAPOOL “number active” value, or the MXT value minus the number of active tasks, whichever is the smaller. For example:

If RAPOOL=2, MXT=50, active tasks = 45 then RECEIVE ANY = 2

If RAPOOL=10, MXT=50, active tasks = 45 then RECEIVE ANY = 5
If RAPOOL=10, MXT=50, active tasks = 35 then RECEIVE ANY = 10

The number of RECEIVE ANYs needed depends on the expected activity of the system, the average transaction lifetime, and the MAXTASK value specified. For information about coding this parameter, see the *CICS Performance Guide*.

RDSASZE={0K|number}
Code this parameter with the size of the RDSA. The default size is 0 indicating that the DSA size can be changed dynamically. A non-zero value indicates that the DSA size is fixed.

number
Specify number as an amount of storage in the range 0 to 16,777,215 bytes in multiples of 262,144 bytes (256KB). If the size specified is not a multiple of 256KB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4,194,304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

Restriction

You can code the RDSASZE parameter in PARM, SYSIPT or CONSOLE only.

RENTPGM={PROTECT|NOPROTECT}
Code this parameter to specify whether you want CICS to allocate the read-only DSAs, RDSA and ERDSA, from read-only key-0 protected storage. The permitted values are PROTECT (the default), or NOPROTECT:

PROTECT	CICS obtains the storage for the read-only DSAs from key-0 protected storage.
NOPROTECT	CICS obtains the storage from CICS-key storage, effectively creating two more CICS DSAs (CDSA and ECDSA). This allows programs eligible for the read-only DSAs to be modified by programs that execute in CICS key.

You are recommended to specify RENTPGM=NOPROTECT for development regions only, and to specify RENTPGM=PROTECT for production CICS regions.

RESP={FME|RRN}
Code this parameter to specify the type of request that CICS terminal control receives from logical units.

FME	Function management end is the default.
RRN	Reached recovery node.

RESSEC={ASIS|ALWAYS}
Specifies whether or not you want CICS to honor the RESSEC option specified on a transaction's resource definition.

ASIS
means that CICS honors the RESSEC option defined in a transaction’s resource definition. CICS calls its resource security checking routine only

when RESSEC(YES) is specified in a transaction resource definition. This is normally a sufficient level of control, because often you will need only to control the ability to execute a transaction.

ALWAYS
means that CICS overrides the RESSEC option, and always calls its resource security checking routine to issue the appropriate call to the SAF interface.

Use this option only if you need to control or audit all accesses to CICS resources. You should be aware that using this option can significantly degrade performance.

Restrictions

You can code the RESSEC parameter in the SIT, PARM, or SYSIPT only.

RMTRAN=({<u>CSGM</u>|name1}[,{<u>CSGM</u>|name2}])
Code this parameter with the name of the transaction you want to initiate when a terminal session is recovered at system restart for a CICS region running with VTAM persistent sessions support.

If you do not specify a name on the RMTRAN parameter, CICS uses the value specified on the GMTRAN system initialization parameter; the CSGM transaction is the default CICS good morning transaction.

name1 This is the transaction that CICS initiates at terminals that do ***not*** remain signed-on after system restart (that is, they are still connected to CICS, but are signed off).

name2 This is the transaction that CICS initiates at terminals that remain signed-on after system restart. If you specify only name1, CICS uses the CSGM transaction as the default for name2.

RUWAPOOL={<u>NO</u>|YES}
specifies the option for allocating a storage pool the first time an LE-conforming program runs in a task.

NO CICS disables the option and provides no RUWA storage pool. Every EXEC CICS LINK to an LE-conforming application results in a GETMAIN for RUWA storage.

YES CICS creates a pool of storage the first time an LE-conforming program runs in a task. This provides an available storage pool that reduces the need to GETMAIN and FREEMAIN run-unit work areas (RUWAs) for every EXEC CICS LINK request.

SDSASZE={<u>0K</u>|number}
Code this parameter with the size of the SDSA. The default size is 0 indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

number
Specify number as an amount of storage in the range 0 to 16,777,215 bytes in multiples of 262,144 bytes (256KB). If the size specified is not a multiple of 256KB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4,194,304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

Restriction

You can code the SDSASZE parameter in PARM, SYSIPT or CONSOLE only.

SEC={YES|NO}
Code this parameter to indicate what level of external security you want CICS to use.

YES
Code YES if you want to use full external security. CICS requires the appropriate level of authorization for the access intent: a minimum of READ permission for read intent, and a minimum of UPDATE permission for update intent.

Note: You must also ensure that the default userid (CICSUSER or another userid specified on the DFLTUSER system initialization parameter) has been defined to your external security manager (ESM).

If command security checking is defined for EXEC CICS SP-type commands, then specifying SEC=YES means that the appropriate level of authority is checked for; therefore:

- A check for READ authority is made for EXEC CICS INQUIRE and COLLECT commands
- A check for UPDATE authority is made for EXEC CICS SET, PERFORM, and DISCARD commands

  For the results of the interaction between the access intent of the user application, and the permission defined to your ESM, see Table 36 on page 260.
- A check for ALTER authority is made for EXEC CICS CREATE commands.

NO
Code NO if you do not want CICS to use an ESM. All users have access to all resources, whether determined by attempts to use them or by the QUERY SECURITY command. Users are not allowed to sign on or off.

Note: With MRO bind-time security, even if you specify SEC=NO, the CICS region userid is still sent to the secondary CICS region, and bind-time checking is still carried out in the secondary CICS region. For information about MRO bind-time security, see the *CICS Security Guide*.

Define whether to use your ESM for resource level checking by using the XDCT, XFCT, XJCT, XPCT, XPPT, XPSB, and XTST system initialization parameters. Define whether to use an ESM for transaction-attach security checking by using the XTRAN system initialization parameter. Define whether ESM session security can be used when establishing APPC sessions by using the XAPPC system initialization parameter.

For information on defining command security checking for CICS SP-type commands, and about CICS security in general, see the *CICS Security Guide*.

For programming information about the use of external security for CICS system commands, see the *CICS System Programming Reference* manual.

Table 36. Results of ESM authorization requests (with SEC=YES)

Access Permission defined to ESM for CICS user	**Access intent in application**		
	READ	**UPDATE**	**ALTER**
NONE	Refused	Refused	Refused
READ	Permitted	Refused	Refused
UPDATE	Permitted	Permitted	Refused
ALTER	Permitted	Permitted	Permitted

Restrictions

You can code the SEC parameter in the SIT, PARM, or SYSIPT only.

SECPRFX={NO|YES}

Specifies whether or not CICS prefixes the resource names in any authorization requests to the ESM with a prefix corresponding to the ESM userid for the CICS region. The prefix to be used (the userid for the CICS region) is obtained by the DFHIRP module.

NO

CICS does not prefix the resource names in any authorization requests to the ESM.

YES

The ESM userid is used as the prefix for CICS resources defined to the ESM. CICS prefixes the resource name in any authorization requests to the ESM with a prefix corresponding to the ESM userid of the CICS region, obtaining the userid as follows:

- If you start CICS as a job, the prefix corresponds to the USER operand coded on the //ID JOB statement of the CICS startup job stream.
- If you start a CICS job without an associated ESM userid, the prefix defaults to CICS.

For information about using the PREFIX option, see the *CICS Security Guide*.

Restrictions

You can code the SECPRFX parameter in the SIT, PARM, or SYSIPT only.

The SECPRFX parameter is effective only if you specify YES for the SEC system initialization parameter.

SIT=xx

Code this parameter to specify the suffix, if any, of the system initialization table that you want CICS to load at the start of initialization. If you omit this parameter, CICS loads the unsuffixed table, DFHSIT, which is pregenerated with all the default values. This default SIT (shown in “DFHSIT, the default system initialization table" on page 192) named DFHSIT$$, and its source, is in the VSE/ESA sublibrary, PRD1.BASE.

Restrictions

You can code the system initialization parameter anywhere in PARM or SYSIPT, or as the ***first*** parameter entry at the CONSOLE.

SKRxxxx='page-retrieval-command'

Code this parameter if a single-keystroke-retrieval operation is required. 'xxxx' specifies a key on the 3270 keyboard which, during a page retrieval session, is to be used to represent a page retrieval command. The valid keys are PA1 through PA3, and PF1 through PF36. Thus up to 39 keys can be specified in this way (each by a separate command).

The 'page-retrieval-command' value represents any valid page retrieval command, and must be enclosed in apostrophes. It is concatenated to the character string coded in the PGRET parameter. The combined length must not exceed 16 characters.

Note: If full function BMS is used, all PA keys and PF keys are interpreted for page retrieval commands, even if some of these keys are not defined.

SNSCOPE={NONE|CICS|VSEIMAGE|}

Specifies whether or not a userid can be signed on to CICS more than once, within the scope of:

- A single CICS region
- A single VSE image

NONE

Each userid can be used to sign on for any number of sessions on any CICS region. This is the compatibility option, providing the same signon scope as in releases of CICS before CICS Transaction Server for VSE/ESA Release 1.

CICS

Each userid can be signed on once only in the same CICS region. A signon request is rejected if the userid is already signed on to the same CICS region. However, the userid can be used to signon to another CICS region in the same, or another, VSE image.

VSEIMAGE

Each userid can be signed on once only, and to only one of the set of CICS regions in the same VSE image that also specify SNSCOPE=VSEIMAGE. A signon request is rejected if the user is already signed on to another CICS region in the same VSE image.

The signon scope (if specified) applies to all userids signing on by an explicit signon request (for example, by an EXEC CICS SIGNON command or the CESN transaction). SNSCOPE is restricted to users signing on at local terminals, or signing on after using the CRTE transaction to connect to another system.

Signon scope specified by SNSCOPE *does not* apply to:

- Non-terminal users.

- The CICS default userid, specified by the DFLTUSER system initialization parameter.
- Preset userids, specified in the USERID option of the RDO TERMINAL resource definition.
- Userids for remote users, received in attach headers.
- Userids for link security. For information about which userid is used for link security on a specific connection, see the *CICS Security Guide*.
- The userid specified on the PLTPIUSR system initialization parameter.
- The CICS region userid.

Restrictions

You can code the SNSCOPE parameter in the SIT, PARM, or SYSIPT only.

SPCTR={(1,2|1[,2][,3])|ALL|OFF}
Code this parameter to set the level of tracing for all CICS components used by a transaction, terminal, or both, selected for special tracing. If you want to set different tracing levels for an individual component of CICS, use the SPCTRxx system initialization parameter. You can select up to three levels of tracing, but some CICS components do not have trace points at all these levels. For a list of all the available trace points and their level numbers, see the *CICS User's Handbook*. For information about the differences between special and standard CICS tracing, see the *CICS Problem Determination Guide*.

number Code the level numbers for the level of special tracing you want for all CICS components. The options are: 1, (1,2), or (1,2,3). The default, (1,2), specifies special tracing for levels 1 and 2 for all CICS components.

ALL Enables the special tracing facility for all available levels.

OFF Disables the special tracing facility.

SPCTRxx={(1,2|1[,2][,3])|ALL|OFF}
Code this parameter to set the level of tracing for a particular CICS component used by a transaction, terminal, or both, selected for special tracing. You identify the component by coding a value for xx in the keyword. You code one SPCTRxx keyword for each component you want to define selectively. For a CICS component being specially traced that does not have its trace level set by SPCTRxx, the trace level is that set by SPCTR (which, in turn, defaults to (1,2)). You can select up to three levels of tracing, but some CICS components do not have trace points at all these levels. The CICS component codes that you can code for xx on the SPCTRxx keyword are shown in Table 37:

Table 37 (Page 1 of 2). CICS component names and abbreviations

Code	Component name	Code	Component name
AP	Application domain	BF	Built-in function
BM	Basic mapping support	CP	Common programming interface
DC	Dump compatibility layer	DD	Directory manager domain
DI	Batch data interchange	DM	Domain manager domain

Table 37 (Page 2 of 2). CICS component names and abbreviations

Code	**Component name**	**Code**	**Component name**
DS	Dispatcher domain	DU	Dump domain
EI	Exec interface	FC	File control
GC	Global catalog domain	IC	Interval control
IS	Inter-system communication	JC	Journal control
KC	Task control	KE	Kernel
LC	Local catalog domain	LD	Loader domain
LM	Lock domain	ME	Message domain
MN	Monitoring domain	PA	Parameter domain
PC	Program control	PG	Program manager domain
RC	Report Controller	SC	Storage control
SM	Storage domain	SP	Sync point
ST	Statistics domain	SZ	Front end programming interface
TC	Terminal control	TD	Transient data
TI	Timer domain	TR	Trace domain
TS	Temporary storage	UE	User exit interface
US	User domain	XM	Transaction manager domain
XS	Security manager domain		

Note: The component codes BF, BM, CP, DC, DI, EI, FC, IC, IS, JC, KC, PC, RC, SC, SP, SZ, TC, TD, TS, and UE are sub-components of the AP domain. As such, the corresponding trace entries will be produced with a point ID of AP nnnn.

For details of using trace, see the *CICS Problem Determination Guide*.

number Code the level numbers for the level of special tracing you want for the CICS component indicated by xx. The options are: 1, (1,2), or (1,2,3).

ALL Code ALL to indicate that you want all the available levels of special CICS tracing switched on for the specified component.

OFF Code OFF to switch off all levels of special CICS tracing for the CICS component indicated by xx.

Restrictions

You can code the SPCTRxx parameter in PARM, SYSIPT, or CONSOLE only.

SPOOL=({NO|YES}[,{A|identifier}][,{A|class}{YES|NO}])

Specifies whether the system spooling interface is required. The identifier and class values are for the report controller.

Use the final YES|NO option to specify whether you want a formfeed (blank page) to precede the first report printed on a non-SCS printer started to the report controller.

The specified identifier is a single character used to identify the printer-owning CICS system to the cross-partition communication (XPCC) component in the device-driving system name (ddsname).

The specified class is the default output class to be used for reports that have no class specified.

NO
Specifies that the system spooling interface is not required.

YES
Specifies that the system spooling interface is required.

A Specifies that the name SYSCICSA is used to identify this CICS system.

identifier
Specifies a single character identifier, which is appended to SYSCICS to create a name used to identify this CICS system. This identifier should be unique for each CICS system running with the system spooling interface. If you do not provide a unique identifier, you might receive message DFHRC5301I (indicating that the POWER interface has failed to initialize).

A Class A is the default for the output class for reports and printers.

class
Specifies a single alpha-numeric character identifier (A to Z, 0-9 only), used as the default output class for reports and printers.

YES
Specifies a formfeed (blank page) is to precede the first report printed on a non-SCS printer started to the report controller.

NO
Specifies a formfeed (blank page) is not to precede the first report printed on a non-SCS printer started to the report controller. If you code NO, be aware that a report can overtype the last line of the previous output when the printer is started to the report controller.

SRT={YES|NO|xx}
Specifies the system recovery table suffix, for more information see page 205.

For information about coding the macros for this table, see the *CICS Resource Definition Guide*.

If SRT=NO is coded, the system recovery program (DFHSRP) does not attempt to recover from a program check or from an operating system abend. However, CICS issues VSE ESTAEX macros to intercept program checks to perform clean-up operations before CICS terminates. Therefore, an SRT must be provided if recovery from either program checks or abnormal terminations, or both, is required.

START=({AUTO|COLD|STANDBY|LOGTERM|}[,ALL])
Specifies the type of start for the system initialization program. The value specified for START, or the default of AUTO, becomes the default value for each resource.

AUTO
CICS performs either a warm or an emergency restart, according to the status of the control record written to the global catalog by the previous

execution of CICS. If the global catalog is newly initialized, it does not contain a control record, and CICS forces a cold start for START=AUTO.

If you code START=AUTO, you must provide a restart data set for use in case CICS has to perform an emergency restart, and the system log from the previous execution of CICS must be available. You must also have coded an activity keypoint value (see the AKPFREQ parameter on page 217) on the previous execution of CICS for an emergency restart to be successful.

COLD
Specifies a cold start. The status of CICS resource definitions saved in the global catalog at the previous shutdown is ignored, and all resource definitions are reinstalled, either from the CSD or CICS control tables. However, some information saved in the global catalog and local catalog is preserved across cold starts. For information about how CICS uses information saved in the global catalog and local catalog, see the *CICS Recovery and Restart Guide*.

LOGTERM
LOGTERM is a special option for use as an alternative to a full emergency restart when the previous run terminated in an uncontrolled shutdown, and is available only if the CICS system log is defined on disk data sets. If you specify START=LOGTERM, CICS initializes up to the point where it writes an end of file on the system log. After it has written the end of file, CICS ends the emergency restart process and shuts down without doing any backout processing.

For background information about this restart process, see the *CICS Recovery and Restart Guide*.

Restriction

START=LOGTERM is applicable only if XRF=NO is coded. You can code START=LOGTERM in PARM, SYSIPT, or CONSOLE only.

STANDBY
Coding START=STANDBY, but only when you have also specified XRF=YES, defines this CICS as the alternate CICS region in an XRF pair. In other words, you ***must*** specify START=STANDBY for the system that starts off as the alternate. (To start an active CICS region, specify AUTO or COLD, as you would without XRF.)

If you have specified a COLD start for other CICS resources, for example, DCT=(xx,COLD), they are cold started when the alternate CICS region (with START=STANDBY specified) takes over. This may cause CICS to lose data on an XRF takeover; for example, coding ICP=COLD results in outstanding STARTs being lost. You are recommended to code START=(STANDBY,ALL) to ensure a full emergency restart during takeover, unless you wish to specifically cold start individual resources.

(option,ALL)
Specifies that the ALL operand is a special option you can use on the START parameter when you supply it as a system initialization parameter at CICS startup; you cannot code it in the SIT. If you specify START=(AUTO,ALL), CICS initializes all resources according to the type of startup that it selects (warm, emergency, or cold). The ALL option

overrides any individual settings in other system initialization parameters (for example, DCT=(xx,COLD)).

However, if you do not use the ALL option, you can individually cold start those resources that have a COLD operand. For details of resources that have a COLD option, see Table 29 on page 205.

Restrictions

You can code START=(option,ALL) in PARM, SYSIPT, or CONSOLE only.

For more information about the types of CICS startup, see "Classes of start and restart" on page 209.

STARTER={NO|YES}
Code YES to indicate that the generation of starter system modules (with $ and # suffixes) is permitted, and various MNOTES are to be suppressed. This parameter should only be used when service is being performed on starter system modules.

Restrictions

You can code the STARTER parameter in the SIT only.

STATRCD=OFF|ON
Specifies the interval statistics recording status at CICS initialization. This status is recorded in the CICS global catalog for use during warm and emergency restarts. Statistics collected are written to the DMF data set.

OFF Interval statistics are not collected (no action is taken at the end of an interval).

End-of-day, Unsolicited and requested statistics are written to DMF regardless of the STATRCD setting.

ON Interval statistics are collected.

On a cold start of a CICS region, interval statistics are recorded by default at three-hourly intervals. All intervals are timed using the end-of-day time (midnight is the default) as a base starting time (***not*** CICS startup time). This means that the default settings give collections at 00.00, 03.00, 06.00, 09.00, and so on, regardless of the time that you start CICS.

On a warm or emergency restart the statistics recording status is restored from the CICS global catalog.

You can change the statistics recording status at any time as follows:

- During a warm or emergency restart by coding the STATRCD system initialization parameter.
- While CICS is running by using the CEMT or EXEC CICS SET STATISTICS command.

Whatever the value of the STATRCD system initialization parameter, you can ask for requested statistics and requested reset statistics to be collected. You can get statistics "on demand" for all, or for specified, resource types by using the CEMT or EXEC CICS PERFORM STATISTICS command. The period covered for statistics requested in this way is from the last reset time (that is, from the beginning of the current interval or from when you last issued a CEMT or EXEC CICS statistics command specifying RESETNOW) up to the time that you issue the PERFORM STATISTICS command.

For information about using these CEMT commands, see *CICS-Supplied Transactions* manual. For programming information about the EXEC CICS PERFORM commands, see the *CICS System Programming Reference* manual.

For information about the statistics utility program DFHSTUP, see *CICS Operations and Utilities Guide*.

STGPROT={NO|YES}
Specifies whether you want storage protection in the CICS region. The permitted values are NO (the default), or YES:

NO If you specify NO, or allow this parameter to default, CICS does not operate any storage protection, and runs in a single storage key as in earlier releases. See Table 39 on page 304 for a summary of how STGPROT=NO affects the storage allocation for the dynamic storage areas.

YES If you specify YES, and if you have the required hardware and software, CICS operates with storage protection, and observes the storage keys and execution keys that you specify in various system and resource definitions. See Table 39 on page 304 for a summary of how STGPROT=YES affects the storage allocation for the dynamic storage areas.

If you do not have the required hardware and software support, CICS issues an information message during initialization, and operates without storage protection.

STGRCVY={NO|YES}
Specifies whether CICS should try to recover from a storage violation.

NO CICS does not try to repair any storage violation that it detects.

YES CICS tries to repair any storage violation that it detects.

In both cases, CICS continues unless you have specified in the dump table that CICS should terminate.

In normal operation, CICS sets up four task-lifetime storage subpools for each task. Each element in the subpool starts and ends with a ‘check zone’ that includes the subpool name. At each freemain, and at end-of-task, CICS checks the check zones and abends the task if either has been overwritten.

Terminal input-output areas (TIOAs) have similar check zones, which are set up with identical values. At each freemain of a TIOA, CICS checks the check zones and abends the task if they are not identical.

If you specify the STGRCVY(YES) system initialization parameter, CICS resets the check zones correctly and the task continues running.

STNTR={1|(1[,2][,3])|ALL|OFF}
Specifies the level of standard tracing required for CICS as a whole. You can select up to three levels of tracing, but some CICS components do not have trace points at all these levels.

Note: Before globally activating tracing levels 3 and ALL for the storage manager (SM) component, read the warning given in the description for the STNTRxx system initialization parameter.

number	Code the level number(s) for the level of standard tracing you want for all CICS components. The options are: 1, (1,2), or (1,2,3). The default, 1, specifies standard tracing for level 1 for all CICS components.
ALL	Enables standard tracing for all levels.
OFF	Disables standard tracing.

For information about the differences between special and standard CICS tracing, see the *CICS Problem Determination Guide*.

STNTRxx={(1,21[,2][,3])|ALL|OFF}

Specifies the level of standard tracing you require for a particular CICS component. You identify the component by coding a value for xx in the keyword. You code one STNTRxx keyword for each component you want to define selectively. For a CICS component being specially traced that does not have its trace level set by STNTRxx, the trace level is that set by STNTR (which, in turn, defaults to 1). You can select up to three levels of tracing, but some CICS components do not have trace points at all these levels.

The CICS component codes that you can code for xx on this STNTRxx keyword are shown in Table 37 on page 262.

number	Code the level number(s) for the level of standard tracing you want for the CICS component indicated by xx. The options are: 1, (1,2), or (1,2,3).
ALL	Code ALL to indicate that you want all the available levels of standard tracing switched on for the specified component.
OFF	Code OFF to switch off all levels of standard CICS tracing for the CICS component indicated by xx.

Attention!

If you select tracing levels 3 or ALL for the storage manager (SM) component, the performance of your CICS system will be degraded. This is because options 3 and ALL switch on levels of trace that are also used for field engineering purposes. See the *CICS Problem Determination Guide* for information about the effects of trace levels 3 and ALL.

Restrictions

You can code the STNTRxx parameter in PARM, SYSIPT, or CONSOLE only.

SUFFIX=xx

Specifies the last two characters of the name of this system initialization table.

The first 6 characters of the name of the SIT are fixed as DFHSIT. You can specify the last two characters of the name, using the SUFFIX parameter. Because the SIT does not have a TYPE=INITIAL macro statement like other CICS resource control tables, you specify its SUFFIX on the TYPE=CSECT macro statement.

The suffix allows you to have more than one version of the SIT. Any one or two characters (other than NO and DY) are valid. You select the version of the table to be loaded into the system during system initialization by coding SIT=xx, either in the PARM parameter or the SYSIPT data set. (You can, in some circumstances, specify the SIT using the system console, but this is not recommended.)

Restrictions

You can code the SUFFIX parameter in the SIT only.

SVA={<u>NO</u>|YES}
Specifies whether any CICS management modules can be used from the VSE shared virtual area (SVA).

<u>NO</u>
No CICS management modules are used from the SVA.

YES
CICS management modules installed in the SVA can be used from there, instead of being loaded into the CICS system.

A list of the CICS modules that are read-only, and hence eligible for residence in the SVA, is given in Appendix B, "CICS modules eligible for the VSE Shared Virtual Area (SVA)" on page 315.

For details of the CICS system initialization parameter PRVMOD that you can use to override SVA=YES for selected modules, see page 255.

SYDUMAX={<u>999</u>|number}
Specifies the limit on the number of system dumps that may be taken per dump table entry. If this number is exceeded, subsequent system dumps for that particular entry will be suppressed.

number
Specifies a number in the range 0 through 999. The default, 999, enables an unlimited number of dumps to be taken.

SYSIDNT={<u>CICS</u>|name}
Specifies a 1- to 4-character name that is known only to your CICS region. If your CICS region also communicates with other CICS regions, the name you choose for this parameter to identify your local CICS region must not be the same name as an installed RDO CONNECTION resource definition for a remote region.

The value for SYSIDNT, whether specified in the SIT or as an override, can only be updated on a cold start. After a warm start or emergency restart, the value of SYSIDNT is that specified in the last cold start.

For information about the SYSIDNT of a local CICS region, see the *CICS Intercommunication Guide*.

SYSTR={<u>ON</u>|OFF}
Specifies the master system trace flag.

Code ON to obtain trace entries of CICS system activity. Entries are written to all the trace destinations that are active.

TAKEOVR={AUTO|MANUAL|COMMAND}

Code the TAKEOVR parameter in the SIT for an alternate XRF system. It specifies the action to be taken by the alternate CICS, following the (apparent) loss of the surveillance signal in the active CICS system. In doing this, it also specifies the level of operator involvement.

AUTO

No operator approval or intervention is needed for a takeover.

MANUAL

Specifies that the operator is asked to approve a take over if the alternate system cannot detect the surveillance signal of the active system.

Note that the alternate system does not ask the operator for approval if the active system signs off abnormally, nor is there an operator or program command for take over. In these cases, there is no doubt that the alternate system should take over, and manual involvement by the operator would be an unnecessary overhead in the take over process.

For example, you could code MANUAL to ensure manual take over of a master or coordinator region in MRO.

COMMAND

Take over only occurs when a CEBT PERFORM TAKEOVER command is received by the alternate system. It ensures, for example, that a dependent alternate CICS (in MRO) is activated only if it receives the command from the operator, or from a master (coordinator) region.

TBEXITS=([name1][,name2][,name3][,name4][,name5])

Code the names of your transaction backout exit programs. These exits are used for resource backout during emergency restart processing. For programming information about transaction backout exit programs, see the *CICS Customization Guide*. For background information about transaction backout, see the *CICS Recovery and Restart Guide*.

The order in which you code the names is critical. If you do not want to use all five exits, code commas in place of the ones you miss out. For example:

```
TBEXITS=(,,EXITF,EXITV)
```

name1	The name of your initialization/termination exit program.
name2	The name of your input exit program.
name3	The name of your file error exit program.
name4	The name of your open error exit program.
name5	The name of your DL/I error exit program.

If no transaction backout exit programs are required, you can do one of the following:

- Omit the whole parameter from the SIT
- Code TBEXITS=(,,,,) as a SIT parameter
- Code TBEXITS=(,,,,) as a SIT override.

TCP={YES|NO}

Code TCP=YES to include the pregenerated non-VTAM terminal control program, DFHTCP.

You must code TCP=YES if you intend using card reader/line printer (sequential) devices.

TCSACTN={<u>NONE</u>|UNBIND|FORCE}

Specifies the required action that CICS terminal control should take if the terminal control shutdown wait threshold expires. For details of the wait threshold, see the TCSWAIT system initialization parameter. TCSACTN only takes effect when TCSWAIT is coded with a value in the range 1 through 99. This parameter only applies to VTAM terminals (including LU Type 6.2 single-session APPC terminals), not VTAM intersystem connections (LU Type 6.1 and LU Type 6.2 parallel connections). This is a global default action. On a terminal-by-terminal basis, you can code a DFHZNEP routine to override this action.

<u>NONE</u>

No action is taken. This can be overridden by DFHZNEP.

UNBIND

CICS terminal control attempts to force-close the session by issuing a VTAM CLSDST and sending an SNA UNBIND command to the hung terminal. This can be overridden by DFHZNEP.

FORCE

CICS terminal control attempts to forceclose the CICS VTAM ACB if there are any hung terminals. All CICS VTAM terminals and sessions are released and CICS normal shutdown continues.

TCSWAIT={<u>4</u>|number|NO|NONE|0}

Specifies the required CICS terminal control shutdown wait threshold. The wait threshold is the time, during shutdown, that CICS terminal control allows to pass before it considers terminal shutdown to be hung. If all VTAM sessions shutdown and close before the threshold expires then the CICS shutdown process moves on to its next stage, and the terminal control wait threshold then no longer applies. If, however, some of the VTAM session do not complete shutdown and close, then CICS takes special action with these sessions. For details of this special action see the description of the TCSACTN system initialization parameter. The wait threshold only applies to VTAM sessions; that is, VTAM terminals and VTAM intersystem connections. The wait time is specified as a number of minutes, in the range 1 through 99. As a special case, TCSWAIT=NO may be specified to indicate that terminal control shutdown is never to be considered hung, no matter how long the shutdown and close process takes. TCSWAIT=NONE and TCSWAIT=0 are alternative synonyms for TCSWAIT=NO, and all three have the same effect (internally they are held as the one value 0 (zero)).

TCT={<u>YES</u>|xx|NO}

Specifies which terminal control table, if any, is to be loaded, for more information see page 205.

For guidance about coding the macros for this table, see the *CICS Resource Definition Guide*

If you reassemble the TCT after starting CICS, any changes are applied when you next start CICS, even if it is a warm or emergency startup.

If you have VTAM-connected terminals only, you can code TCT=NO. If you do this, note that a dummy TCT, called DFHTCTDY, is loaded during system initialization. For more information about DFHTCTDY, see page 208. (If you code TCT=NO, you must specify a CSD group list in the GRPLIST parameter.)

TCTUAKEY={USER|CICS}

Specifies the storage key for the terminal control table user areas (TCTUAs) if you are operating CICS with storage protection (STGPROT=YES). The permitted values are USER (the default), or CICS:

USER	If you specify USER, or allow this parameter to default, CICS obtains the amount of storage for TCTUAs in user key. This allows a user program executing in any key to modify the TCTUA.
CICS	If you specify CICS, CICS obtains the amount of storage in CICS key. This means that only programs executing in CICS key can modify the TCTUA, and user-key programs have read-only access.

If CICS is running without storage protection, the TCTUAKEY parameter only designates which DSA (User or CICS) the storage comes from. The TCTUAs are accessed in CICS-key whether they are in the UDSA or CDSA.

See “The terminal control table user areas” on page 302 for more information about TCTUAs.

TCTUALOC={BELOW|ANY}

Specifies where terminal user areas (TCTUA) are to be stored.

BELOW

Specifies that the TCTUAs are stored below the 16MB line.

ANY

Specifies that the TCTUAs are stored anywhere in virtual storage. CICS stores TCTUAs above the 16MB line if possible.

For more information about TCTUAs, see “The terminal control table user areas” on page 302.

For details about defining terminals using RDO, see the *CICS Resource Definition Guide*.

TD=({3|decimal-value-1}[,{3|decimal-value-2}])

Specifies the number of VSAM buffers and strings to be used for intrapartition transient data (TD).

decimal-value-1

Specifies that the number of buffers to be allocated for the use of intrapartition transient data. The value must be in the range 1 through 32 767. The default value is 3.

CICS obtains, above the 16MB line, storage for the TD buffers in units of the page size (4KB). Because CICS optimizes the use of the storage obtained, TD may allocate more buffers than you specify, depending on the control interval (CI) size you have defined for the intrapartition data set.

For example, if the CI size is 1536, and you specify 3 buffers (the default number), CICS actually allocates 5 buffers. This is because 2 pages (8192 bytes) are required to obtain sufficient storage for three 1536-byte buffers, a total of only 4608 bytes, which would leave 3584 bytes of spare storage in the second page. In this case, CICS allocates another 2 buffers (3072 bytes) to minimize the amount of unused storage. In this way CICS makes use of storage that would otherwise be unavailable for any other purpose.

decimal-value-2

Specifies that the number of VSAM strings to be allocated for the use of intrapartition transient data. The value must be in the range 1 through 255, and must not exceed the value specified in decimal-value-1. The default value is 3.

For example, TD=(8,5) specifies 8 buffers and 5 strings.

The operands of the TD parameter are positional. You must code commas to indicate missing operands if others follow. For example, TD=(,2) specifies the number of strings and allows the number of buffers to default.

TRAP={<u>OFF</u>|ON}

Specifies whether the FE global trap exit is to be activated at system initialization. This exit is for diagnostic use under the guidance of service personnel. For background information about this exit, see the *CICS Problem Determination Guide*.

TRDUMAX={<u>999</u>|number}

Specifies the limit on the number of transaction dumps that may be taken per dump table entry. If this number is exceeded, subsequent transaction dumps for that particular entry will be suppressed.

number

Specifies a number in the range 0 through 999. The default, 999, enables an unlimited number of dumps to be taken.

TRTABSZ={<u>16</u>|number-of-kilobytes}

Specifies the size in kilobytes of the internal trace table. (1KB = 1024 bytes.) The CICS trace table is allocated in virtual storage above the 16MB line, and it is allocated ***before*** the extended CICS-key DSA (ECDSA) and the extended user-key DSA (EUDSA). Ensure that there is sufficient virtual storage for the trace table, the ECDSA, and the EUDSA by defining a large enough VSE partition for your CICS job.

<u>16</u> 16KB is the default size of the trace table, and also the minimum size.

number The number of kilobytes of storage to be allocated for the internal trace table, in the range 16KB through 1048576KB. Subpool 1 is used for the trace table storage, which exists for the duration of the CICS execution. The table is page aligned and occupies a whole number of pages. If the value specified is not a multiple of the page size (4KB), it is rounded up to the next multiple of 4KB.

Trace entries are of variable lengths, but the average length is approximately 100 bytes.

Note: To switch on internal tracing, use the INTTR parameter; for a description of INTTR, see page 243.

TRTRANSZ={<u>40</u>|number-of-kilobytes}

specifies the size in kilobytes of the transaction dump trace table. (1KB = 1024 bytes.)

When a transaction dump is taken, CICS performs a VSE GETMAIN for storage above the 16MB line for the transaction dump trace table.

40 40KB is the default size of the transaction dump trace table. The minimum size is 16KB.

number
The number of kilobytes of storage to be allocated for the transaction dump trace table, in the range 16KB through 1048576KB.

TRTRANTY={TRAN|ALL}
specifies which trace entries should be copied from the internal trace table to the transaction dump trace table.

TRAN
Only the trace entries associated with the transaction that is abending will be copied to the transaction dump trace table.

ALL All of the trace entries from the internal trace table will be copied to the transaction dump trace table. If the internal trace table size is larger than the transaction dump trace table size, the transaction dump trace table could wrap. This results in only the most recent trace entries being written to the transaction dump trace table.

TS=([COLD][,{0|3|decimal-value-1}][,{3|decimal-value-2}])
Specifies:

- Whether or not you want to cold start temporary storage
- The number of VSAM buffers to be used for auxiliary temporary storage
- The number of VSAM strings to be used for auxiliary temporary storage.

COLD
Specifies that the type of start for the temporary storage program. If you do not want a cold start, code a comma before the second operand.

Note: IF ICP is warm-started (that is no ICP=COLD) and TS is cold-started, any ICEs and AIDs with data attached are lost.

0 No buffers are required; that is, only MAIN temporary storage is required.

decimal-value-1
Specifies that the number of buffers to be allocated for the use of auxiliary temporary storage. The value must be in the range 3 through 32 767.

CICS obtains, above the 16MB line, storage for the auxiliary temporary storage buffers in units of the page size (4KB). Because CICS optimizes the use of the storage obtained, TS may allocate more buffers than you specify, depending on the control interval (CI) size you have defined for the auxiliary temporary storage data set.

For example, if the CI size is 2048, and you specify 3 buffers (the default number), CICS actually allocates 4 buffers. This is because 2 pages (8192 bytes) are required to obtain sufficient storage for three 2048-byte buffers, a total of 6144 bytes, which would leave 2048 bytes of spare storage in the second page. In this case, CICS allocates another 2048-byte buffer to use all of the 8192 bytes obtained. In this way CICS makes use of storage that would otherwise be unavailable for any other purpose.

decimal-value-2
Specifies that the number of VSAM strings to be allocated for the use of auxiliary temporary storage. The value must be in the range 1 through

255, and must not exceed the value specified in decimal-value-1. The default value is 3.

For example, TS=(COLD,8,5) specifies 8 buffers and 5 strings.

The operands of the TS parameter are positional. You must code commas to indicate missing operands if others follow. For example, TS=(,8) specifies the number of buffers and allows the other operands to default.

TSMGSET={4|number}
Specifies that the number of entries for which dynamic storage is allocated for storing pointers to records put to a temporary storage message set. When the entries are used, space is acquired for the same number of entries as many times as required to accommodate the total number of records in the queue. The range is 4 through 100.

TST={NO|YES|xx}
Specifies the temporary storage table suffix, for more information see page 205.

For information about coding the macros for this table, see the *CICS Resource Definition Guide*

UDSASZE={0K|number}
Specifies the size of the UDSA. The default size is 0 indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

number
Specify number as an amount of storage in the range 0 to 16,777,215 bytes in multiples of 262,144 bytes (256KB). If the size specified is not a multiple of 256KB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4,194,304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

Restrictions

You can code the UDSASZE parameter in PARM, SYSIPT and CONSOLE only.

USERTR={ON|OFF}
To set the master user trace flag on or off. If the user trace flag is off, the user trace facility is disabled, and EXEC CICS ENTER TRACENUM commands receive an INVREQ condition if EXCEPTION is not specified. If the program does not handle this condition the transaction will abend with an abend code of AEIP.

For programming information about the user trace facility using EXEC CICS ENTER TRACENUM commands, see the *CICS Application Programming Reference* manual.

USRDELAY={30|number}
Specify the maximum time, in the range 0 through 10080 minutes (up to 7 days), that an eligible userid and its associated attributes are to be retained in the user table if the userid is unused. An entry in the user table for a userid that is retained during the delay period can be reused.

The userids eligible for reuse within the USRDELAY period are any that are:

- Received from remote systems
- Specified on SECURITYNAME in RDO CONNECTION definitions
- Specified on USERID in RDO SESSIONS definitions
- Specified on USERID on DFHDCT TYPE=INTRA definitions
- Specified to be used for non-terminal STARTed transactions.

Within the USRDELAY period, a userid in any one of these categories can be reused in one of the other categories, provided the request for reuse is qualified with the same qualifiers. If a userid is qualified by different group id, APPLID, or terminal id, a retained entry is not reused.

If a userid is unused for more than the USRDELAY limit, it is removed from the system, and the message DFHUS0200 is issued. You can suppress this message in an XMEOUT global user exit program. If you specify USRDELAY=0, all eligible userids are deleted immediately after use, and the message DFHUS0200 is not issued. Do not code USRDELAY=0 if this CICS region communicates with other CICS regions and:

- ATTACHSEC=IDENTIFY is specified on the CONNECTION definitions for the connections used
- The connections used carry high volumes of transaction routing or function shipping activity.

You should specify a value that gives the optimum level of performance for your CICS environment.

Note: If a value, other than 0, is specified for USRDELAY, the ability to change the user's attributes or revoke the userid becomes more difficult because the userid and its attributes are retained in the region until the USRDELAY value has expired. For example, if you have specified USRDELAY=30 for a userid, but that userid continues to run transactions every 25 minutes, the USRDELAY value will never expire and any changes made to the userid will never come into effect.

When running a remote transaction, a userid remains signed-on to the remote CICS region (after the conversation associated with the first attach request is complete) until the delay specified by USRDELAY has elapsed since the last transaction associated with the attach request for the userid has completed. When this event occurs, the userid is removed from the remote CICS region.

VTAM={<u>YES</u>|NO}
Code VTAM=NO only if you do not use the VTAM access method. The default is VTAM=YES.

VTPREFIX={\|character}
Specifies that the first character to be used for the terminal identifiers (termids) of autoinstalled virtual terminals. Virtual terminals are used by the External Presentation Interface (EPI) and terminal emulator functions of the CICS Client products.

Termids generated by CICS for autoinstalled Client terminals consist of a 1-character prefix and a 3-character suffix. The default prefix is `'\'`. The suffix can have the values `'AAA'` through `'999'`. That is, each character in the suffix can have the value `'A'` through `'Z'` or `'0'` through `'9'`. The first suffix generated by CICS has the value `'AAA'`. This is followed by `'AAB'`, `'AAC'`, ... `'AAZ'`, `'AA0'`, `'AA1'`, and so on, up to `'999'`.

Each time a Client virtual terminal is autoinstalled, CICS generates a 3-character suffix that it has not recorded as being in use.

By specifying a prefix, you can ensure that the termids of Client terminals autoinstalled on this system are unique in your transaction routing network. This prevents the conflicts that could occur if two or more terminal-owning regions (TORs) ship definitions of Client virtual terminals to the same application-owning region (AOR).

If such a naming conflict does occur—that is, if a Client virtual terminal is shipped to an AOR on which a remote terminal of the same name is already installed—the autoinstall user program is invoked in the AOR. Your user program can resolve the conflict by allocating an alias terminal identifier to the shipped definition. (For details of writing an autoinstall user program to install shipped definitions, see the *CICS Customization Guide*). However, you can avoid potential naming conflicts by specifying a different prefix, reserved for virtual terminals, on each TOR on which Client virtual terminals are to be installed.

You must not use the characters + − * < > = { } or blank.

Notes:

1. The autoinstall user program is not called at install of Client terminals, so cannot be used to specify termids.
2. When specifying a prefix, ensure that termids generated by CICS for Client terminals do not conflict with those generated by your autoinstall user program for user terminals, or with the names of any other terminals or connections.
3. Client terminal definitions are not recovered after a restart. Immediately after a restart, no Client terminals are in use, so when CICS generates suffixes it begins again with `'AAA'`. This means that CICS does **not** always generate the same termid for any given Client terminal. This in turn means that server applications should not assume that a particular CICS-generated termid always equates to a particular Client terminal.
4. Clients can override CICS-generated termids.

For further information about Client virtual terminals, see the *CICS Family: Communicating from CICS on System/390* manual.

WRKAREA={512|number}
Specifies the number of bytes to be allocated to the common work area (CWA). This area, for use by your installation, is initially set to binary zeros, and is available to all programs. It is not used by CICS. The maximum size for the CWA is 3584 bytes.

XAPPC={NO|YES}
Specifies whether ESM security can be used when establishing APPC sessions.

NO
ESM session security cannot be used. Only the BINDPASSWORD (defined to CICS for an APPC connection) is checked.

YES
ESM session security can be used.

If you specify BINDSECURITY=YES for a particular APPC connection, a request to the ESM is issued to extract the security profile. If the profile exists, it is used to bind the session. If it does not exist, only the BINDPASSWORD (defined to CICS for the connection) is checked.

Note: If you specify XAPPC=YES, the external security manager that you use must support the APPCLU general resource class, otherwise CICS fails to initialize.

Restrictions

You can code the XAPPC parameter in the SIT, PARM, or SYSIPT only.

XCMD={<u>NO</u>|name|YES}
specifies whether you want CICS to perform command security checking, and, optionally, the ESM resource class name in which you have defined the command security profiles. If you specify YES, or an ESM resource class name, CICS calls the ESM to verify that the userid associated with a transaction is authorized to use a CICS command for the specified resource. Such checking is performed every time a transaction tries to use a COLLECT, CREATE, DISABLE, DISCARD, ENABLE, EXTRACT, INQUIRE, PERFORM, RESYNC, or SET command, or any of the FEPI commands, for a resource.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the CMDSEC(YES) option on the RDO TRANSACTION resource definition.

For information about preparing for and using security with CICS, see the *CICS Security Guide*.

<u>NO</u>
CICS does not perform any command security checks, allowing any user to use commands that would be subject to those checks.

name
CICS calls the ESM using the specified resource class name, prefixed by C or V, to verify that the userid associated with a transaction is authorized to use a CICS command for the specified resource. The resource class name is C*name* and the grouping class name is V*name*.

The resource class name specified must be 1 through 7 characters.

YES
CICS calls the ESM using the default class name of CICSCMD, prefixed by C or V, to check whether the userid associated with a transaction is authorized to use a CICS command for the specified resource. The resource class name is CCICSCMD and the grouping class name is VCICSCMD.

Restrictions

You can code the XCMD parameter in the SIT, PARM, or SYSIPT only.

XDCT={<u>NO</u>|name|YES}

Specifies whether you want CICS to perform transient data resource security checking. If you specify YES or an ESM resource class name, CICS calls the ESM to verify that the userid associated with a transaction is authorized to access the transient data destination. Such checking is performed every time a transaction tries to access a transient data destination.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the RDO TRANSACTION resource definition.

For information about preparing for and using security with CICS, see the *CICS Security Guide*.

<u>NO</u>

CICS does perform any transient data security checks, allowing any user to access any transient data destination.

name

CICS calls the ESM using the specified resource class name to check whether the userid associated with the transaction is authorized to access the specified destination. The resource class name is D*name* and the grouping class name is E*name*.

The resource class name specified must be 1 through 7 characters.

YES

CICS calls the ESM with the default CICS resource class name of CICSDCT, prefixed by D or E, to verify whether the userid associated with the transaction is authorized to access the specified destination.

The resource class name is DCICSDCT and the grouping class name is ECICSDCT.

Restrictions

You can code the XDCT parameter in the SIT, PARM, or SYSIPT only.

XFCT={<u>NO</u>|name|YES}

Specifies whether you want CICS to perform file resource security checking, and optionally specifies the ESM resource class name in which you have defined the file resource security profiles. If you specify YES, or an ESM resource class name, CICS calls the ESM to verify that the userid associated with a transaction is authorized to access file control-managed files. Such checking is performed every time a transaction tries to access a file managed by CICS file control.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the RDO TRANSACTION resource definitions.

For information about preparing for and using security with CICS, see the *CICS Security Guide*.

<u>NO</u>

CICS does not perform any file resource security checks, allowing any user to access any file.

name
CICS calls the ESM, using the specified resource class name, to verify that the userid associated with a transaction is authorized to access files referenced by the transaction. The resource class name is F*name* and the grouping class name is H*name*.

The resource class name specified must be 1 through 7 characters.

YES
CICS calls the ESM, using the default CICS resource class name of CICSFCT prefixed by F or H, to verify that the userid associated with a transaction is authorized to access files reference by the transaction. The resource class name is FCICSFCT and the grouping class name is HCICSFCT.

Restrictions

You can code the XFCT parameter in the SIT, PARM, or SYSIPT only.

XJCT={<u>NO</u>|name|YES}
Specifies whether you want CICS to perform journal resource security checking. If you specify YES, or an ESM resource class name, CICS calls the ESM to verify that the userid associated with a transaction is authorized to access the referenced journal. Such checking is performed every time a transaction tries to access a CICS journal.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the RDO TRANSACTION resource definitions.

For information about preparing for and using security with CICS, see the *CICS Security Guide*.

<u>NO</u>
CICS does not perform any journal resource security checks, allowing any user to access any CICS journal.

name
CICS calls the ESM, using the specified resource class name prefixed by J or K, to verify that the userid associated with a transaction is authorized to access CICS journals.

The resource class name specified must be 1 through 7 characters.

YES
CICS calls the ESM, using the default CICS resource class name of CICSJCT prefixed by a J or K, to check whether the userid associated with a transaction is authorized to access CICS journals referenced by the transaction. The resource class name is JCICSJCT and the grouping class name is KCICSJCT.

Restrictions

You can code the XJCT parameter in the SIT, PARM, or SYSIPT only.

XLT={<u>NO</u>|xx|YES}
Specifies a suffix for the transaction list table, for more information see page 205.

The XLT contains a list of transactions that can be attached during the first quiesce stage of system termination.

NO
Specifies that a transaction list table is not used.

xx Specifies that the transaction list table DFHXLTxx is used.

YES
Specifies that the default transaction list table, DFHXLT, is used.

For guidance information about coding the macros for this table, see the *CICS Resource Definition Guide*

XPCT={NO|name|YES}
Specifies whether you want CICS to perform started transaction resource security checking, and optionally specifies the name of the ESM resource class name in which you have defined the started task security profiles. If you specify YES, or an ESM resource class name, CICS calls the ESM to verify that the userid associated with a transaction is authorized to use started transactions and related EXEC CICS commands. Such checking is performed every time a transaction tries to use a started transaction or one of the EXEC CICS commands: COLLECT STATISTICS TRANSACTION, DISCARD TRANSACTION, INQUIRE TRANSACTION, or SET TRANSACTION.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the RDO TRANSACTION resource definitions.

For information about preparing for and using security with CICS, see the *CICS Security Guide*.

NO
CICS does not perform any started task resource security checks, allowing any user to use started transactions or related EXEC CICS commands.

name
CICS calls the ESM using the specified resource class name, to verify that the userid associated with a transaction is authorized to use started transactions or related EXEC CICS commands. The resource class name is ACICSPCT and the grouping class name is BCICSPCT.

The resource class name specified must be 1 through 7 characters.

YES
CICS calls the ESM, using the default CICS resource class name CICSPCT prefixed with A or B, to verify that the userid associated with a transaction is authorized to use started transactions or related EXEC CICS commands.

The resource class name is ACICSPCT and the grouping class name is BCICSPCT.

Restrictions

You can code the XPCT parameter in the SIT, PARM, or SYSIPT only.

XPPT={<u>NO</u>|name|YES}

Specifies that CICS is to perform application program resource security checks, and optionally specifies the ESM resource class name in which you have defined the program resource security profiles. Such checking is performed every time a transaction tries to invoke another program by using one of the CICS commands: LINK, LOAD, or XCTL.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the RDO TRANSACTION resource definitions.

For information about preparing for and using security with CICS, see the *CICS Security Guide*.

<u>NO</u>

CICS does not perform any application program authority checks, allowing any user to use LINK, LOAD, or XCTL commands to invoke other programs.

name

CICS calls the ESM, with the specified resource class name prefixed by M or N, to verify that the userid associated with a transaction is authorized to use LINK, LOAD, or XCTL commands to invoke other programs. The resource class name is M*name* and the grouping class name is N*name*.

The resource class name specified must be 1 through 7 characters.

YES

CICS calls the ESM, using the default resource class name prefixed by M or N, to verify that the userid associated with a transaction is authorized to use LINK, LOAD, or XCTL commands to invoke other programs. The resource class name is MCICSPPT and the grouping class name is NCICSPPT.

Restrictions

You can code the XPPT parameter in the SIT, PARM, or SYSIPT only.

XPSB={<u>NO</u>|name|YES}

Specifies whether you want CICS to perform program specification block (PSB) security checking, and optionally specifies the ESM resource class name in which you have defined the PSB security profiles. If you specify YES, or an ESM resource class name, CICS calls the ESM to check that the userid associated with a transaction is authorized to access PSBs (which describe databases and logical message destinations used by application programs). Such checking is performed every time a transaction tries to access a PSB.

The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the RDO TRANSACTION resource definitions.

For information about preparing for and using security with CICS, see the *CICS Security Guide*.

<u>NO</u>

CICS does not perform any PSB resource security checks, allowing any user to access any PSB.

name

CICS calls the ESM using the specified resource class name prefixed by P or Q, to verify that the userid associated with a transaction is authorized to access PSBs. The resource class name is P*name* and the grouping class name is Q*name*.

The resource class name specified must be 1 through 7 characters.

YES

CICS calls the ESM, using the default resource class name CICSPSB prefixed by P or Q, to verify that the userid associated with a transaction is authorized to access PSBs. The resource class name is PCICSPSB and the grouping class name is QCICSPSB.

Restrictions

You can code the XPSB parameter in the SIT, PARM, or SYSIPT only.

XRF={NO|YES}

You must specify YES if you want XRF support to be included in the CICS region. If the CICS region is started with the START=STANDBY system initialization parameter specified, the CICS region is the **alternate CICS region**. If the CICS region is started with the START=AUTO or START=COLD system initialization parameter specified, the CICS region is the **active CICS region**. The active CICS region signs on as such to the CICS availability manager. For background information about XRF, see the *CICS XRF Guide*.

Note: You must code JCT=xx|YES, if you are using XRF.

XRFSOFF={NOFORCE|FORCE}

Specifies whether all users signed-on to the active CICS region are to remain signed-on following a takeover. This parameter is only applicable if you also code XRF=YES as a system initialization parameter.

NOFORCE

Code NOFORCE to allow CICS to determine sign-off according to the option set in either of:

- The CICS segment of the ESM database
- The RDO TYPETERM resource definition for the user's terminal.

Note: For a terminal to remain signed-on after an XRF takeover, NOFORCE must be specified in the SIT, ESM database, and the terminal's RDO TYPETERM resource definition.

FORCE

Code FORCE if you want ***all*** terminal users to be signed off in the event of a takeover by an alternate CICS region, regardless of individual options set in the ESM database or in the terminals' RDO TYPETERM resource definition.

For information about the XRFSOFF option of the RDO TYPETERM resource definition, see the *CICS Resource Definition Guide*.

XRFSTME={5|decimal-value}

Specifies, in minutes, a time-out delay interval for users who are still signed-on when an XRF takeover occurs.

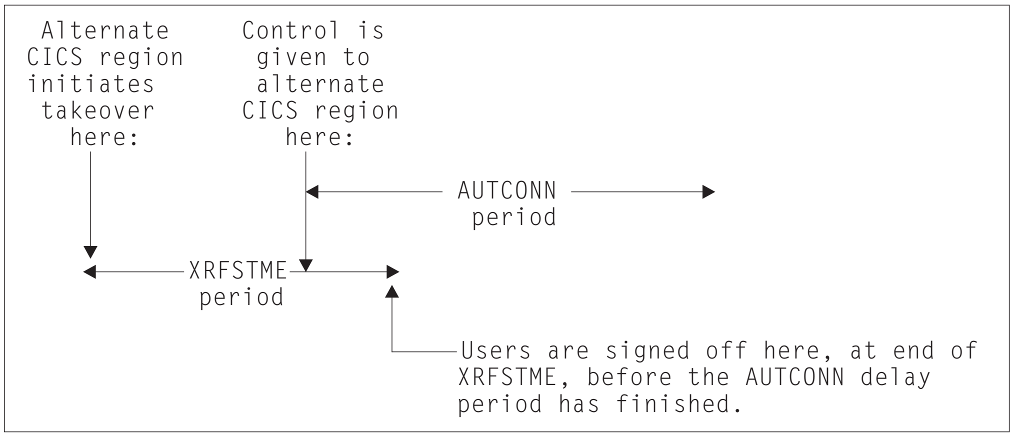
If you have specified NOFORCE in the ESM database, and in the terminals' RDO TYPETERM resource definition, and the takeover takes longer than the time specified in the XRFSTME, all users who are still signed-on after takeover are signed off.

5 The default value is five minutes.

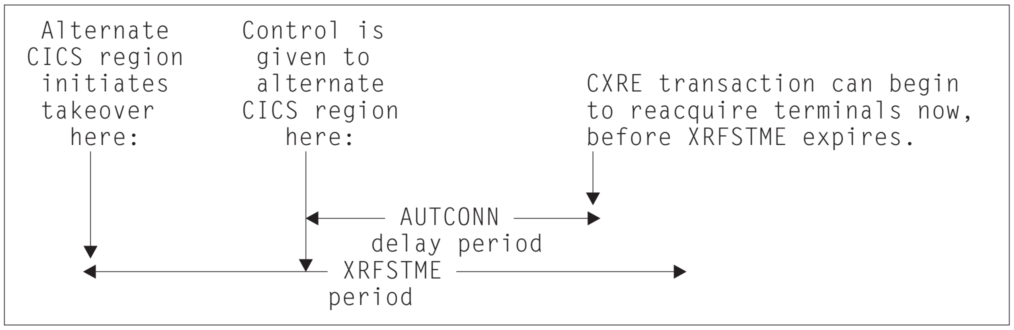
decimal-value
Code a value in the range 0 through 60 for the number of minutes CICS permits users to remain signed on during the takeover period. The takeover period is the time from when the takeover is initiated to the time at which CICS is ready to process user transactions. If the takeover takes longer than the specified period, all users signed-on at the time the takeover was initiated are signed-off.

A value of 0 specifies that there is no time-out delay, and terminals are signed off as soon as takeover commences, which means that XRFSTME=0 has the same effect as coding XRFSOFF=FORCE.

For non-XRF-capable terminals, take into account any AUTCONN delay period when setting the value for XRFSTME. (See the description of the AUTCONN parameter on page 218.) You may need to increase the XRFSTME value to allow for the delay to the start of the CXRE transaction imposed by the AUTCONN parameter; otherwise, terminals may be signed-off too early. For example:


Alternate CICS region initiates takeover here:
Control is given to alternate CICS region here:
AUTCONN period
XRFSTME period
Users are signed off here, at end of XRFSTME, before the AUTCONN delay period has finished.

You can avoid this situation by extending the XRFSTME period to exceed the AUTCONN period. For example:


Alternate CICS region initiates takeover here:
Control is given to alternate CICS region here:
CXRE transaction can begin to reacquire terminals now, before XRFSTME expires.
AUTCONN delay period
XRFSTME period

XRFTODI={30|number}
Use the XRFTODI parameter, in the SIT for an alternate XRF system, to specify, in seconds, the takeover delay interval. The minimum time value is 5

seconds. The alternate system has to ensure that the active system has been canceled before it can take over the resources owned by the active. Because the cancelation process is not fully automatic (for example, the primary CPC fails), the XRFTODI value specifies the interval before the operator becomes involved.

XSWITCH=([{<u>0</u>|1-254|NO}][,{<u>????????</u>|progname}][,{<u>A</u>|B}])
Specifies that the XSWITCH parameter defines a programmable terminal switching unit, which may be used with mid-range 2-CPC XRF systems instead of a communication controller. The program defined on the XSWITCH parameter instructs the unit to switch terminal lines to the active system's CPC at start up, and to the alternate system's CPC at take over.

<u>0</u>|1-254
Specifies that the logical unit to which the switch is assigned. 0 is the default.

<u>????????</u>|progname
Specifies that the user-written program that issues commands to the switching unit. The program is device-dependent, but has a standard interface that follows normal subroutine conventions.

<u>A</u>|B
Specifies that the CEC to which the terminal lines are to be directed. A is the active CPC and B is the alternate CPC,

XTRAN={<u>YES</u>|name|NO}
Specifies whether you want CICS to perform transaction-attach security checking, and optionally specifies the ESM resource class name in which you have defined the transaction security profiles. If you specify YES, or an ESM resource class name, CICS calls the ESM to verify that the userid associated with the transaction is permitted to run the transaction.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the RDO TRANSACTION resource definitions.

<u>YES</u>
CICS calls the ESM, using the default CICS resource class name of CICSTRN prefixed by T or G, to verify that the userid associated with the transaction is authorized to run the transaction. The resource class name is TCICSTRN and the grouping class name is GCICSTRN.

name
CICS calls the ESM, using the specified resource class name prefixed by T or G, to verify that the userid associated with the transaction is authorized to run the transaction. The resource class name is T*name* and the corresponding grouping class name is G*name*.

The name specified must be 1 through 7 characters.

NO
CICS does not perform any transaction-attach security checks, allowing any user to run any transaction.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter.

Restrictions

You can code the XTRAN parameter in the SIT, PARM, or SYSIPT only.

XTST={NO|name|YES}

specifies whether you want CICS to perform temporary storage security checking, and optionally specifies the ESM resource class name in which you have defined the temporary storage security profiles. If you specify YES or an ESM resource class name, CICS calls the ESM to verify that the userid associated with a temporary storage request is authorized to access the referenced temporary storage queue.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter, specified the RESSEC option on the RDO TRANSACTION resource definitions, and specified a DFHTYPE=SECURITY macro for the queue in the temporary storage table (TST).

NO

CICS does not perform any temporary storage security checks, allowing any user to access any temporary storage queue.

name

CICS calls the ESM, using the specified resource class name prefixed by S or U, to verify that the userid associated with a transaction is is authorized to access temporary storage queues.

The name specified must be 1 through 7 characters.

YES

CICS calls the ESM, using the default CICS resource class name of CICSTST prefixed by S or U, to verify that the userid associated with the transaction is authorized to access temporary storage queues referenced by the transaction. The resource class name is SCICSTST and the corresponding grouping class name is UCICSTST.

Restrictions

You can code the XTST parameter in the SIT, PARM, or SYSIPT only.

XUSER={NO|YES}

Specifies whether or not CICS is to perform surrogate user checks.

NO

Specifies that CICS is not to perform any surrogate user checking.

YES

specifies that CICS is to perform surrogate user checking in all those situations that permit such checks to be made (for example, on EXEC CICS START commands without an associated terminal). For information about the various circumstances in which CICS performs surrogate user checks, see the *CICS Security Guide*.

Restrictions

You can code the XUSER parameter in the SIT, PARM, or SYSIPT only.

Part 4. Running a CICS system

This section of the book describes how to operate a CICS system.

Table 38. CICS system startup road map

If you want to...	**Refer to...**
See a sample startup job stream	“Sample startup job stream” on page 290
The storage requirements for a CICS region	“Storage requirements for a CICS region” on page 300
The new sample statistics program, DFH0STAT	“The sample statistics program, DFH0STAT” on page 307

Chapter 23. CICS startup

This chapter describes how to start up CICS in the CICS region. Depending on your system environment, use a procedure cataloged in a VSE sublibrary, or submit the CICS startup job stream through the reader.

This chapter gives an example of each of these methods. For an example of a batch job that you can submit through the internal reader, see “A sample CICS startup job” on page 290.

When you run the startup job, you start a process called **CICS system initialization**. This process must finish before you run any transactions. Completion of CICS initialization is shown by the following message at the system console:

```
DFHSI1517 applid Control is being given to CICS.
```

CICS initialization involves many activities, some of which are:

- Obtaining the required storage for CICS execution from the private area in the CICS address space, above and below the 16MB line
- Setting up CICS system parameters for the run, as specified by the system initialization parameters
- Loading and initializing the CICS domains according to the start option specified by the START system initialization parameter
- Loading the CICS nucleus with the required CICS modules
- Installing CICS resource definitions by:
  - Loading, from the CSD, the groups of resources specified by the GRPLIST= system initialization parameter
  - Loading the control tables specified by system initialization parameters.
- If XRF=YES is specified, signing on to the CICS availability manager (CAVM) to check that it is possible to continue initialization and perform the role requested, that is, as an active or alternate CICS region
- Opening the data sets necessary for initialization, including any needed for backout if the previous run of your CICS region was not shut down normally (except for START=STANDBY, when most data sets are not opened until after takeover)
- Opening SAM sequential devices as required in the terminal control table (TCT).

Also, if you are operating CICS with CICS recovery options, backout procedures may be used to restore recoverable resources to a logically consistent state. Briefly, backout occurs if you start CICS in one of the following ways:

- With START=AUTO and CICS detects that the previous shutdown was immediate or uncontrolled
- With START=STANDBY and XRF=YES, and a takeover occurs.

For background information about backout, and recovery and restart, see the *CICS Recovery and Restart Guide*.

In the final stages of initialization, a set of programs can be executed as specified in a program list table (PLT). You specify the suffix of the PLT you want by means of the PLTPI parameter in the SIT. Initialization PLT programs run under control of a CICS task in CICS key. Their storage is in CICS-key storage, and protected from overwriting by other transactions. For more information about storage protection, see "Storage protection" on page 301. For programming information about writing PLT programs, see the *CICS Customization Guide*.

Sample startup job stream

You can use the sample job control statements shown in Figure 59 on the following pages to initialize a CICS region. The sample job stream shown in Figure 59 specifies START=AUTO and XRF=YES to start an active CICS region. The notes that follow Figure 59 explain which statements are unique to XRF, and can therefore be omitted if you want to run with XRF=NO.

The sample startup job stream is based on the system initialization parameters contained in the CICS-supplied sample table, DFHSIT6$. For details of all the system initialization options specified in the sample table, see the source of DFHSIT6$ which is contained in the VSE/ESA sublibrary, PRD1.BASE.

A sample CICS startup job

```
// JOB CICSACT
* Define the job options  1
*
// OPTION    LOG,SYSDUMP,SYSDUMPC
* Defining the user VSAM catalog
// DLBL CICSUCT,'usercat',,VSAM
// EXTENT SYS001,cicsvol
*
*       Auxiliary temporary storage data set
// DLBL DFHTEMP,'CICS410.CICSHTH1.DFHTEMP',,VSAM,CAT=CICSUCT  2
*
*       Intrapartition data set
// DLBL DFHNTRA,'CICS410.CICSHTH1.DFHNTRA',,VSAM,CAT=CICSUCT  3
*
*       The auxiliary trace data sets
// DLBL DFHAUXT,'CICS410.CICSHTH1.DFHAUXT',0,SD  4
// EXTENT SYS002,,1,0,rtrk,ntrks
// DLBL DFHBUXT,'CICS410.CICSHTH1.DFHBUXT',0,SD
// EXTENT SYS002,,1,0,rtrk,ntrks
```

Figure 59. CICS startup job stream 1 of 3

```
*
*       Extrapartition data sets
// DLBL LOGUSR,'CICS410.CICSHTH1.LOGUSR',0,SD       5
// EXTENT SYS002,,1,0,rtrk,ntrks
// DLBL MSGUSR,'CICS410.CICSHTH1.MSGUSR',0,SD
// EXTENT SYS002,,1,0,rtrk,ntrks
// DLBL CEEMSG,'CICS410.CICSHTH1.CEEMSG',0,SD
// EXTENT SYS002,,1,0,rtrk,ntrks
// DLBL CEEOUT,'CICS410.CICSHTH1.CEEOUT',0,SD
// EXTENT SYS002,,1,0,rtrk,ntrks
*
*       The CICS system log data sets
// DLBL DFHJ01A,'CICS410.CICSHTH1.DFHJ01A',0,SD 6
// EXTENT SYS019,,1,0,rtrk,ntrks
// DLBL DFHJ01B,'CICS410.CICSHTH1.DFHJ01B',0,SD
// EXTENT SYS019,,1,0,rtrk,ntrks
*
*       User journal data set
// DLBL DFHJ02A,'CICS410.CICSHTH1.DFHJ02A',0,SD 7
// EXTENT SYS018,,1,0,rtrk,ntrks
*
*       Automatic journal archiving control data set
// DLBL DFHJACD,'CICS410.CICSHTH1.DFHJACD',,VSAM,CAT=CICSUCT 8
*
*       The restart data set
// DLBL DFHRSD,'CICS410.CICSHTH1.DFHRSD',,VSAM,CAT=CICSUCT 9
*
*
*       The CICS local catalog data set
// DLBL DFHLCD,'CICS410.CICSHTH1.DFHLCD',,VSAM,CAT=CICSUCT 10
*
*       The CICS global catalog data set
// DLBL DFHGCD,'CICS410.CICSHTH1.DFHGCD',,VSAM,CAT=CICSUCT,    11
                BUFND=33,BUFNI=32,BUFSP=303104
*
*       The CAVM data sets - XRF
// DLBL DFHXMSG,'CICS410.CICSHTH1.DFHXMSG',,VSAM,CAT=CICSUCT 12
// DLBL DFHXCTL,'CICS410.CICSHTH1.DFHXCTL',,VSAM,CAT=CICSUCT
*
*       The transient data destination data set (CXRF)
// DLBL DFHCXRF,'CICS410.CICSHTH1.DFHCXRF',0,SD 13
// EXTENT SYS002,,1,0,rtrk,ntrks
*
*       The CICS transaction dump data sets
// DLBL DFHDMPA,'CICS410.CICSHTH1.DFHDMPA',0,SD
// EXTENT SYS002,,1,0,rtrk,ntrks                      14
// DLBL DFHDMPB,'CICS410.CICSHTH1.DFHDMPB',0,SD
// EXTENT SYS002,,1,0,rtrk,ntrks
*
*       VSE system dump data set (SYSDUMP)
// DLBL SYSDUMP,'VSE.DUMP.FILE',0,SD 15
// EXTENT SYS044,,1,0,rtrk,ntrks
// LIBDEF DUMP,CATALOG=SYSDUMP.cicsdump
```

Figure 60. CICS startup job stream 2 of 3

```
*
*      The CICS system definition (CSD) data set
// DLBL DFHCSD,'CICS410.CICSHTH1.DFHCSD',,VSAM,CAT=CICSUCT  16
*
*      FILEA and other permanently allocated data sets
*
// DLBL FILEA,'CICS410.CICSHTH1.FILEA',,VSAM,CAT=CICSUCT
*
// DLBL APPLICA,'CICS410.CICSHTH1.APPLICA',,VSAM,CAT=CICSUCT  17
// DLBL APPLICB,'CICS410.CICSHTH1.APPLICB',,VSAM,CAT=CICSUCT
*  Assign the disk logical units
// ASSGN SYS001,DISK,VOL=CICVOL,SHR
// ASSGN SYS002,DISK,VOL=CICVOL,SHR
// ASSGN SYS018,DISK,VOL=CICVOL,SHR
// ASSGN SYS019,DISK,VOL=CICVOL,SHR
// ASSGN SYS044,DISK,VOL=CICVOL,SHR
*      Your user-defined DL/I database
*
// DLBL DLIDB,'CICS410.CICSHTH1.DLIDB',,VSAM,CAT=CICSUCT
*
// LIBDEF PHASE,SEARCH=(your.program.library,                     *
               your.table.library,                                *
               PRD2.SCEECICS,PRD2.SCEEBASE) 18
// LIBDEF SOURCE,SEARCH=(archjcl.library)  8
*  The CICS EXEC statement
// EXEC DFHSIP,SIZE=DFHSIP,PARM='SIT=6$,DSALIM=6M,EDSALIM=20M,DBP=2$,REN*
               TPGM=PROTECT,STGPROT=YES,START=AUTO,SI',OS390  19  20
GRPLIST=(DFHLIST,userlist1,userlist2),
SVA=YES,
APPLID=CICSHTH1,
*
DFLTUSER=CICSUSER,   The default userid  21
MXT=40,              Maximum number of tasks is 40
ISC=YES,             Include intersystem communication program
IRCSTRT=YES,         Start interregion communication
.END
/*
  ⋮
                                                     \  22  23
    user transactions input from sequential terminal \
                                                     \
  ⋮

CESF GOODNIGHT\
/*
/&
```

Figure 61. CICS startup job stream 3 of 3

Notes:

1 The JOB statement

The JOB statement specifies the accounting information you want to use for this run of CICS. For example:

```
// JOB CICSACT <accounting information>
```

2 Auxiliary temporary storage (DFHTEMP) data sets

Define this data set if you want to save data to use later. The temporary storage queues used, identified by symbolic names, exist until explicitly deleted. Even after the originating task is deleted, temporary data can be accessed by other tasks, through references to the symbolic name under which it is stored.

For details of how to define these data sets, and for information about space calculations, see Chapter 10, “Defining the temporary storage data set” on page 101.

3 Intrapartition transient data (DFHNTRA) data sets

The transient data intrapartition data set is used for queuing messages and data within the CICS region.

For information about how to define these data sets, and about space calculations, see “Defining the intrapartition data set” on page 106.

4 Auxiliary trace (DFHAUXT and DFHBUXT) data sets

Define one or both of these sequential data sets, if you want to use auxiliary trace. If you define automatic switching for your auxiliary trace data sets, define both data sets. If you define only one data set, its DLBL name must be DFHAUXT.

For details of how to define these data sets, see Chapter 17, “Defining and using auxiliary trace data sets” on page 159.

The auxiliary trace data sets in this job stream are unique to the active CICS region, and as such are identified in our example by using the specific applid of the active CICS region (CICSHTH1) as a second-level qualifier. If you are using auxiliary trace with XRF=YES, the alternate CICS region also needs its own trace data sets. These could be identified by the specific applid of the alternate CICS region, for example, CICSHTH2.

5 Extrapartition transient data destinations

LOGUSR, and MSGUSR are examples of data sets for extrapartition transient data destinations. You can choose the DLBL names of these data sets, but they must agree with the DSCNAMEs on the corresponding DFHDCT TYPE=SDSCI definitions in the destination control table (DCT). In this example, based on the sample table DCT=2$:

- LOGUSR defines a user data set used by the CICS sample programs (DESTID=LOGA)
- MSGUSR defines the data set used by a number of CICS services (DESTID=CSSL)
- CEEMSG is used as an error queue only when you are running application programs under Language Environment for VSE/ESA.
- CEEOUT is used as an output queue only when you are running application programs under Language Environment for VSE/ESA.

For information about the destinations used by CICS in the sample table DFHDCT2$, see the copy member DFH$DCTR in the VSE/ESA sublibrary PRD1.BASE.

6 The CICS system log data sets

Journals on disk data sets

The sample job stream illustrates the DLBL statements you need when the CICS system log is defined on disk data sets.

Journals on unlabeled tapes

You can define the system log on unlabeled tapes, but this is not recommended; the preferred medium is disk. On unlabeled tapes the DLBL and EXTENT statements would be replaced with ASSGN job control statements of the form:

```
// ASSGN SYS004,CUU
// ASSGN SYS005,CUU
```

where SYS004 and SYS005 are specified as follows in the JCT

```
JTYPE=TAPE2,DEVADDR=(SYS004,SYS005)
```

Attention!

If your log is on unlabeled tapes, CICS acts as if any tape mounted is the correct one. You must be careful, therefore, when mounting unlabeled tapes on the DFHJ01A drive.

7 User journal data set

This is an example of how to specify a user journal data set.

8 The automatic journal archiving data sets

If you specify JOUROPT=(AUTOARCH,.,.) in the JCT, you must provide DLBL statements for DFHJACD. This data set provides control information for automatic journal archiving.

The sublibrary containing the journal archive skeletal JCL should be specified on a LIBDEF SOURCE,SEARCH statement.

9 The restart data set

The restart data set is used during emergency restarts only, to hold temporarily the backout information read from the CICS log.

For information about creating and initializing the RSD, see Chapter 15, “Defining and initializing the restart data set” on page 147.

10 The CICS local catalog data set

The CICS local catalog is used by the CICS domains to save some of their information between CICS runs, and to preserve this information across a cold start. The local catalog is not shared by any other CICS system. If you are running CICS with XRF, define a unique local catalog for the active CICS region, and another for the alternate CICS region. For information about creating and initializing a CICS local catalog, see Chapter 16, “Defining and using catalog data sets” on page 151.

11 The CICS global catalog data set

The CICS global catalog has a variety of uses, including:

- During the running of CICS, holding resource definitions that are installed, DL/I status information, and disk journal status information.
- During a controlled shutdown, recording warm keypoint information for a later warm start. There is only one global catalog, which is passively shared by the active and alternate CICS regions.

For information about creating and initializing a CICS global catalog, see Chapter 16, “Defining and using catalog data sets” on page 151.

This sample job illustrates the use of the VSE/VSAM parameters on the DLBL statement. Specifying these parameters can help to improve restart and shutdown time. This example is based on the recommended DEFINE CLUSTER statements shown in Figure 49 and the associated notes on page 153. The values given are the minimum values suitable for these parameters and should not be reduced.

12 The CAVM data sets

These data sets are required when you are running CICS with XRF. They are actively shared by the active and the alternate CICS regions. For details of how to create and initialize the CICS availability data sets, see Chapter 19, “Defining the CICS availability manager data sets” on page 169.

13 The DFHCXRF transient data set (CXRF)

This transient data destination is used by CICS as the target for messages sent to any transient data destination before CICS has completed intrapartition transient data initialization. It is particularly necessary in an XRF environment for use in an alternate CICS region before takeover has occurred, during the period when transient data initialization is suspended. For more information about the DFHCXRF data set, see “The DFHCXRF data set” on page 110.

14 CICS transaction dump data sets

CICS records transaction dumps on a sequential data set, or pair of sequential data sets, tape or disk. The data sets must be defined with the DLBL names DFHDMPA and DFHDMPB, but if you define only one data set, its DLBL name must be DFHDMPA. CICS always attempts to open at least one transaction dump data set during initialization.

For information about defining CICS transaction dump data sets and how they are used, see Chapter 18, “Defining dump data sets” on page 163.

The transaction dump data sets in this job stream are unique to the active CICS region, and as such are identified by using the specific applid of the active CICS region (CICSHTH1) as a second-level qualifier. The alternate CICS region needs its own transaction dump data sets, and these could be identified by using the specific applid of the alternate CICS region (CICSHTH2) as a second-level qualifier.

15 VSE system dump data sets

Include a SYSDUMP DLBL statement if you want SDUMP dumps sent to a VSE dump sublibrary. You must also specify a // OPTION SYSDUMP statement.

16 The CICS system definition (CSD) file

The system definition file (CSD) is required by CICS to hold some resource definitions.

You must provide job control DLBL statements for the CSD.

For information about creating and initializing the CSD, see Chapter 14, “Defining the CICS system definition data set” on page 129.

If you are running CICS with XRF, particularly in a multi-CPC environment, there are special considerations concerning CSD sharing; these considerations are discussed under “Sharing and availability of the CSD” on page 133.

17 Sample program file (FILEA) and other permanently allocated data sets

You may want to provide job control DLBL statements for those user files that are defined in the CSD (if you are using RDO) or in a file control table (if you are using the DFHFCT macro). If you do, the data sets are allocated at the time of CICS job step initiation, and ***remain allocated for the duration of the CICS job step***. FILEA, the distributed library containing the data for the sample application programs, is included in the startup job stream as an example of direct allocation by job control statement.

On the other hand, if you are using RDO, you may prefer to take advantage of the CICS dynamic allocation of files. For dynamic allocation, ***do not*** specify a DLBL statement for the file. CICS then uses the full data set name as specified in the DSNAME parameter of the RDO FILE resource definition (up to 44 characters), to allocate the file as part of OPEN processing. This form of dynamic allocation applies equally to files that are defined to be opened explicitly, and those that are to be opened on first reference by an application.

For more information about opening files, see Chapter 20, “Defining user files” on page 175. For information about the parameters that you can code on FILE resource definitions, see the *CICS Resource Definition Guide*.

18 CICS LIBDEF PHASE,SEARCH search chain

The LIBDEF PHASE,SEARCH chain must include the library containing your CICS application programs, shown in the example as *your.prog.library*, and your CICS control tables, shown in the example as *your.table.library*.

19 The EXEC DFHSIP statement

DFHSIP is the program that starts CICS initialization.

Attention!

CICS does not support more than one EXEC DFHSIP job step in the same VSE job.

You determine how much of the allocated private storage you want for the CICS dynamic storage areas, and how much CICS is to leave for demands on operating system storage, by setting values for the DSALIM and EDSALIM system initialization parameters. After obtaining the amount of space required for the DSAs from the total defined for the partition by the VSE ALLOC parameter, the remaining storage is available to meet demands for operating system storage.

In the sample job stream, these system initialization parameters are specified in the PARM parameter. (See **20** on page 297.)

For more details about the ALLOC parameter and CICS storage, see “Storage requirements for a CICS region” on page 300.

20 Options of the PARM parameter

You can use the PARM parameter of the EXEC statement to specify system initialization parameters as shown.

The information passed by the PARM parameter is limited to 100 characters. This limit includes all commas, but excludes the apostrophes delimiting the PARM parameter. (Internal parentheses enclosing system initialization operands ***are*** included.) You can code the PARM parameter up to three times on one EXEC statement. The syntax rules above apply to each PARM parameter separately.

If 300 characters are not sufficient for the system initialization parameters you want to provide at startup, indicate continuation by ending the PARM field with the “SYSIPT” or “CONSOLE” control keywords (or “SI” or “CN” for short).

If you specify SYSIPT, system initialization parameters are read from the SYSIPT data stream; if you specify CONSOLE, CICS prompts you to enter parameters through the console. However, if all of your run-time system initialization parameters are in the PARM parameter, you can end the PARM field simply without any control keywords, or by the .END control keyword.

In the example, DFHSIT6$ is the SIT selected, and CICS system initialization uses the values in that table, modified by the system initialization parameters supplied in the PARM field and the SYSIPT data stream. For this example, the following system initialization parameters are provided in the PARM parameter:

DBP Because CICS local DL/I support is being added to this run of CICS, the version of the dynamic backout program (DBP) that includes DL/I support (DFHDBP2$) is also specified.

RENTPGM Storage for the read-only DSAs, RDSA and ERDSA, is obtained from key-0, non-fetch protected storage by using the default PROTECT option on the RENTPGM system initialization parameter. You are recommended to specify RENTPGM=NOPROTECT for development CICS regions and RENTPGM=PROTECT for production CICS regions. For more information about the RENTPGM parameter, see page 257.

STGPROT Storage protection is obtained for this run of CICS by specifying YES on the STGPROT system initialization parameter. Before using this parameter, check in the *CICS Release Guide* that you have the required hardware and software. See page 267 for details about the STGPROT parameter itself.

START START=AUTO is normally the type of start you would select for a production CICS system, allowing CICS to determine the class of start to be performed (warm, emergency, or cold).

If you are running CICS with XRF, START=AUTO causes CICS to initialize as an active CICS region. To request initialization of CICS as an alternate CICS region, specify XRF=YES and START=STANDBY.

For information about the types of CICS startup and shutdown in an XRF environment, see the *CICS Operations and Utilities Guide*.

SI The PARM statement is terminated with SI (short for SYSIPT), to tell CICS to continue reading overrides from the SYSIPT data stream.

21 SYSIPT data stream

You include the SYSIPT data stream inline as part of the job stream. System initialization parameters entered in the SYSIPT data stream replace any, for the same keyword, that were entered in the PARM parameter. If you include the same parameter more than once, the ***last*** value read is the value used for initialization.

Unless you explicitly code the system initialization control keyword CONSOLE, CICS stops reading system initialization parameters when it reaches the end of SYSIPT or a .END control keyword.

In the sample job, CONSOLE is not coded in either PARM or SYSIPT. The .END control keyword is the last entry in SYSIPT, so CICS does not prompt through the console for further system initialization parameters. After reading the SYSIPT data set, CICS loads the specified SIT, applies any system initialization parameters supplied in the PARM field and the SYSIPT data stream, and begins the initialization process.

The SYSIPT data set in the example includes several system initialization parameters, as follows:

GRPLIST
The group list defined in DFHSIT6$ is DFHLIST, the IBM-defined list that is generated when you initialize the CSD using the DFHCSDUP INITIALIZE command. DFHLIST contains only the standard IBM-defined resource definitions required by CICS. One of the group lists specified on the GRPLIST system initialization parameter must contain those resource definitions required by CICS. You can do this either by including the resource definitions in one of your own group lists that you specify, or by specifying the DFHLIST explicitly, as shown. Your own group lists (userlist1 and userlist2 as shown) should contain all the resource definitions generated by your installation for your applications that are required for this CICS run. In addition, your group lists should contain any definitions required for IBM program products that you are using, such as LE/VSE or DB2®.

SVA
The SIT specifies that modules are not to be used from the VSE shared virtual area (SVA); SVA=YES in SYSIPT specifies that modules are to be used from the SVA in this run.

APPLID
The applid for this CICS region is CICSHTH1. If you want this CICS region to use VTAM for terminal access or ISC communication, this applid must be defined to VTAM.

If you want this CICS region to use XRF, you must define both a generic and specific applids; for example, by:

```
APPLID=(CICSID,CICSHTH1),
```

CICSID is the generic applid of this CICS region, and CICSHTH1 is the specific applid of the active CICS region. With XRF=YES, the active and alternate CICS regions share the same generic applid, but have different specific applids.

The specific applid can be useful for naming those data sets that are unique (for example, dump data sets). Where necessary, it can be used as the second-level qualifier to distinguish the data sets of the active and alternate CICS regions.

DFLTUSER
CICSUSER is the default userid specified to the security manager. During startup, CICS tries to sign on the default userid. If it cannot be signed on (for example, if not defined), CICS issues a message and terminates CICS initialization. After the valid default userid is signed on, its security attributes are used for all CICS terminal users who do not sign on with the CESN transaction. If the default userid is defined to the ESM with a CICS segment, the operator attributes in that segment are also used for users who do not sign on.

MXT
The maximum number of user tasks is limited to 40 for this run. For information about what tasks are included in the MXT parameter, see page 248.

INITPARM
Passes parameters to programs.

ISC
Include the intersystem communication program, DFHISP, in order to use interregion communication (IRC).

IRCSTRT
This CICS is running with MRO, and interregion communication is started during initialization.

22 Sequential (SAM) devices as simulated terminals

The card reader/line printer (CRLP) simulated terminals shown in our sample job stream are defined in the sample TCT (not used in this startup job). See the copy member DFH$TCTS, in the VSE/ESA sublibrary, PRD1.BASE, for the source statements you need for such devices. For information about defining these devices in a TCT, see the *CICS Resource Definition Guide*.

23 Quiescing sequential devices

For sequential devices, the last entry in the input stream can be CESF GOODNIGHT\ to provide a logical close, and quiesce the device. However, if you close a device in this way, the receive-only status is recorded in the warm keypoint at CICS shutdown. This means that the terminal is still in RECEIVE status in a subsequent warm start, and CICS does not then read the input file. For more information about how to restart a device that has been closed in a previous run of CICS by means of a CESF GOODNIGHT transaction, see page 76.

Note the end-of-data character (the “\” symbol) at the end of each line of the sample.

Storage requirements for a CICS region

This section describes considerations about CICS storage requirements. For information about CICS storage requirements, and the effect on CICS performance, see the *CICS Performance Guide*.

You should review the partition size specified on the ALLOC parameter for CICS partitions. The increase in CICS use of virtual storage above the 16MB boundary means that you will probably need to increase the value of the ALLOC parameter. CICS allocates storage above 16MB in multiples of 1MB and storage below the 16MB line in multiples of 256K. The CICS Transaction Server for VSE/ESA Release 1 requirement for real storage varies according to the transaction load at any one time. As a guideline, each task in the system requires 9KB of real storage, and this should be multiplied by the number of concurrent tasks that can be in the system at any one time (governed by the MXT system initialization parameter).

However, automatic DSA sizing removes the need for accurate storage estimates, with CICS dynamically changing the size of DSAs as demand requires.

When calculating the size of your CICS partition, allow for the following areas of private storage in the CICS region, above and below the 16MB line:

- Storage that CICS reserves at the start of CICS initialization for exclusive use by the CICS kernel.
- Storage that the CICS storage manager reserves for the dynamic storage areas, which you specify on the system initialization parameters DSALIM and EDSALIM.
- Storage remaining in the private area, after the kernel and DSA requirements have been allocated, to meet additional CICS storage demands from the operating system, for control blocks and buffers for the various access methods.
- If you are running CICS with DL/I support, the amount of storage left after deducting DSA requirements must be sufficient for DL/I VSE code; DL/I VSE modules are not loaded as part of the nucleus.
- If you are using CICS data tables, the amount of storage left after deducting the CDSA requirements must be enough for the records you want to include in the data tables.

For guidance information about virtual storage management in a VSE environment, see the *VSE/ESA Planning* guide.

Storage protection

CICS releases before CICS Transaction Server for VSE/ESA Release 1 run mainly in single storage key (partition key) for the whole of their execution. Running a single storage key means that user application programs have the same access to CICS code and control blocks as CICS itself, and there is no protection against “rogue” applications overwriting parts of storage inadvertently.

CICS Transaction Server for VSE/ESA Release 1 uses extensions to ESA/390™ storage protection facilities available with VSE/ESA Version 2 Release 4 to prevent CICS code and control blocks from being overwritten accidentally by your own user application programs. This is done by allocating separate storage areas (with separate storage keys) for your user application programs, and for CICS code and control blocks. Access to a storage area is not permitted unless the access key matches the key for that storage area.

The storage allocated for CICS code and control blocks is known as **CICS-key** storage, and the storage allocated for your user application programs is known as **user-key** storage. In addition to CICS-key and user-key storage, CICS Transaction Server for VSE/ESA Release 1 can also use **key-0 storage** for separate dynamic storage areas below and above the 16MB boundary called the **read-only** DSAs (RDSA and ERDSA). The ERDSA is used for eligible re-entrant CICS and user application programs link-edited with the SVA and RMODE(ANY) attributes. The allocation of key-0 storage for the read-only DSAs is from the same storage limit as the other DSAs, as specified by the DSALIM and EDSALIM system initialization parameters.

Use of the storage protection facilities are optional. You can enable them by coding options on new storage protection system initialization parameters. Between them, these new parameters enable you to define or control:

- The storage key for the common work area (CWAKEY)
- The storage key for the terminal control table user areas (TCTUAKEY)
- A storage protection global option (STGPROT)
- A read-only program storage key option (RENTPGM)

To help you get started, CICS provides DFHSIT$$, a default system initialization table. This default table is supplied in the VSE/ESA sublibrary, PRD1.BASE, in source form, and you can modify this to suit your own requirements. When assembled and link-edited, DFHSIT$$ becomes the unsuffixed DFHSIT, which is also supplied in pregenerated form in the VSE/ESA sublibrary, PRD1.BASE.

The common work area

The common work area (CWA) is an area of storage within your CICS region that any user application can access. You determine the size of this work area by means of the WRKAREA system initialization parameter, which allows you to specify sizes up to 3584 bytes. If you omit the WRKAREA parameter, CICS allocates a 512-byte CWA by default. You specify the storage key for the CWA on the CWAKEY parameter.

Because this work area is available to all transactions in a CICS region, you should ensure that the storage key is appropriate to the use of the CWA by all transactions. If there is only one transaction that runs in user key, and which requires **write** access, you must specify user-key storage for the CWA, otherwise it will fail with a storage protection exception (an ASRA abend). CICS obtains user-key storage for the CWA by default. You must review the use of this storage by all programs before you decide to change it to CICS key.

It is possible that you might want to protect the CWA from being overwritten by applications that should not have write access. In this case, provided all the transactions that legitimately require write access to the CWA run in CICS key, you can specify CICS-key storage for the CWA.

See page 228 for details of how to specify the CWAKEY system initialization parameter.

The terminal control table user areas

A terminal control user area (TCTUA) is an optional storage area associated with a terminal control table terminal entry (TCTTE), and is available for application program use.

For VTAM terminals, you specify that you want a TCTUA by means of the USERAREALEN parameter on the RDO TYPETERM resource definition. The USERAREALEN parameter on a RDO TYPETERM resource definition determines the TCTUA sizes for all terminals that reference the RDO TYPETERM resource definition.

For sequential terminals, definitions are added to the terminal control table (TCT), and sizes are defined by means of the TCTUAL parameter on the DFHTCT TYPE=TERMINAL and TYPE=LINE entries. For information about the TCTUAL parameter, see the *CICS Resource Definition Guide*.

You specify the storage key for the TCTUAs globally for a CICS region by the TCTUAKEY system initialization parameter. By default, CICS obtains user-key storage for all TCTUAs.

You must review the use of TCTUAs in your CICS regions, and specify CICS key only for TCTUAs when you are sure that this is justified. If you specify CICS-key storage for TCTUAs, no user-key applications can write to any TCT user areas.

See page 272 for details of how to specify the TCTUAKEY parameter.

The storage protection global option

You can control whether your CICS region uses storage protection by specifying the STGPROT system initialization parameter. By default, CICS does not use storage protection, and all applications run in the same key as CICS, as in earlier releases.

The default option is suitable for pure terminal-owning regions (TORs) and data-owning regions (DORs) that do not execute user transactions. If you want storage protection in a CICS region, you must specify this on the STGPROT system initialization parameter.

See page 267 for details of how to specify the STGPROT parameter.

The read-only storage override option

CICS obtains storage for the read-only DSAs (RDSA and ERDSA) from VSE read-only storage. CICS loader automatically loads eligible modules into the RDSA and ERDSA; that is, if they are link-edited with the SVA attribute, and for the ERDSA with RMODE(ANY). If you do not want such modules to be loaded into read-only storage (perhaps because you are using a development aid package that sets break points in your application programs) you can override the selection of read-only storage for the RDSA and ERDSA by specifying NOPROTECT on the RENTPGM system initialization parameter.

Note: When you specify RENTPGM=NOPROTECT, CICS still allocates separate read-only DSAs, but obtains CICS key-storage for the RDSA and ERDSA instead of read-only storage.

You are recommended to specify RENTPGM=NOPROTECT for development regions only, and to specify RENTPGM=PROTECT for production CICS regions. See page 257 for details of how to specify the RENTPGM parameter.

The dynamic storage areas and associated storage cushions

CICS Transaction Server for VSE/ESA Release 1 supports eight dynamic storage areas (DSAs), each with its own "storage cushion". CICS always allocates eight separate DSAs, even if you are running without storage protection (either because the necessary hardware or VSE support is not available, or because you switch off storage protection by means of the STGPROT system initialization parameter). If you are running with STGPROT=NO, CICS allocates the DSAs that are controlled by the STGPROT parameter from CICS-key storage (the same storage key as in earlier releases). However, because the CICS and user DSAs are physically separate, it makes the overwriting of CICS storage less likely, even if the DSAs are all in the same storage key.

The type of storage for the read-only DSAs is controlled by the RENTPGM system initialization parameter.

The eight DSAs are as follows:

CDSA	The CICS DSA, allocated below the 16MB boundary, always from CICS-key storage
RDSA	The read-only DSA, allocated below the 16MB boundary from either read-only storage or CICS-key storage depending on the RENTPGM system initialization parameter
SDSA	The shared DSA, allocated below the 16MB boundary from either user-key or CICS-key storage depending on the STGPROT system initialization parameter.
UDSA	The user DSA, allocated below the 16MB boundary from either user-key or CICS-key storage depending on the STGPROT system initialization parameter
ECDSA	The extended CICS-key DSA, allocated above the 16MB boundary, always from CICS-key storage
ERDSA	The extended read-only DSA, allocated above the 16MB boundary, either from read-only storage or CICS-key storage, depending on the RENTPGM system initialization parameter

ESDSA The extended shared DSA, allocated above the 16MB boundary from either user-key or CICS-key storage depending on the STGPROT system initialization parameter.

EUDSA The extended user DSA, allocated above the 16MB boundary, from either user-key or CICS-key storage depending on the STGPROT system initialization parameter

Table 39 shows the type of storage allocated according to the system initialization parameters specified.

You specify the overall limits within which CICS can allocate storage for the individual DSAs both above and below the 16MB by coding values on the DSALIM and EDSALIM system initialization parameters

You can optionally specify the sizes of the CDSA, RDSA, SDSA, and UDSA using the CDSASZE, RDSASZE, SDSASZE and USDASZE system initialization parameters respectively.

You optionally specify the sizes of the ECDSA, ERDSA, ESDSA, and EUDSA using the ECDSASZE, ERDSASZE, ESDSASZE, and EUDSASZE system initialization parameters respectively.

Using these values, CICS dynamically controls the sizes of the individual DSAs and their associated cushions. You can vary these overall limits dynamically by using either the CEMT SET SYSTEM or DSAS command or an EXEC CICS SET SYSTEM command.

Table 39. Controlling the storage key for the dynamic storage areas

Dynamic storage area	STGPROT= NO	STGPROT= YES	RENTPGM= PROTECT	RENTPGM= NOPROTECT
CDSA	CICS key	CICS key	N/A[1]	N/A[1]
RDSA	N/A[2]	N/A[2]	Read-only key-0	CICS key
SDSA	CICS key	User key	N/A[1]	N/A[1]
UDSA	CICS key	User key	N/A[1]	N/A[1]
ECDSA	CICS key	CICS key	N/A[1]	N/A[1]
ESDSA	CICS key	User key	N/A[1]	N/A[1]
ERDSA	N/A[2]	N/A[2]	Read-only key-0	CICS key
EUDSA	CICS key	User key	N/A[1]	N/A[1]

Notes:

1. Not applicable. The RENTPGM option has no effect on these DSAs, for which the storage key is determined only by the STGPROT parameter.
2. Not applicable. The STGPROT option has no effect on the read-only DSAs, for which the storage key is determined only by the RENTPGM parameter.

The storage cushions

CICS reserves amounts of storage in the dynamic storage areas (DSAs) for use when processing storage stress conditions. Each reserved area, which consists of contiguous virtual storage, is called a “storage cushion.” A storage stress condition occurs when CICS cannot satisfy a GETMAIN request, or can satisfy it only by using some of the cushion storage even when all programs that are eligible for deletion, and not in use, have been deleted. This may lead to a short-on-storage condition, if CICS cannot rectify the stress condition.

CICS dynamically tunes the size of the DSA storage cushions as necessary, within the limits set by the DSALIM and EDSALIM system initialization parameters. However, if the amount of storage available for the storage cushions becomes too small, an SOS condition can still occur.

Effects: In a storage stress condition, the cushion mechanism can avert a storage deadlock condition. This prevents CICS taking on additional work by stopping most of the soliciting for new input messages. For information on the effects of stress conditions, see the *CICS Performance Guide*.

CICS sets the storage stress condition if:

- After any successful GETMAIN, the number of unallocated dynamic storage pages remaining is less than the cushion size. This is shown in the storage statistics as “Times cushion released”.
- CICS cannot satisfy an unconditional GETMAIN because there is no contiguous area large enough for it. This is shown in the storage statistics as “Times request suspended”.

When a storage stress situation exists, the loader domain attempts to alleviate it by releasing the main storage for programs with no current user. If this fails, a short-on-storage condition is indicated, and a message is issued at the console.

While the SOS condition is set, acquisition of new input message areas is prevented, and all ATTACH requests from CICS system modules are deferred.

Recommendations

To help CICS optimize its use of the DSAs and their storage cushions, you are recommended to:

- Avoid using large GETMAIN requests.

  The storage cushion is a contiguous block of storage of fixed size, and therefore may be able to satisfy a request for a large contiguous block of storage.
- Minimize the number of resident programs.

If storage cushion releases occur frequently, you need to find out why. Reduce the maximum number of user tasks (the MXT system initialization parameter) to reduce the number of tasks using main storage. You may need either to alter your application programs so that they do not issue large GETMAINs.

Only transactions defined as SPURGE(YES) and with a DTIMOUT value can be purged during an SOS condition if they have been waiting for storage for longer than the DTIMOUT value. If such transactions are too few and if storage becomes totally deadlocked, the system can stall.

How implemented

CICS allocates the initial size of the storage cushions for the DSAs from the overall storage limits defined by the DSALIM and EDSALIM system initialization parameters. CICS dynamically tunes the sizes of the DSAs and their storage cushions within these limits.

For descriptions of the DSALIM and EDSALIM system initialization parameters, see page 231 and 234 respectively.

You can change the overall storage limits while CICS is running by means of a CEMT SET SYSTEM or DSAS command or an EXEC CICS SET SYSTEM command.

How monitored

Storage stress conditions are notified in the storage statistics ("Times cushion released" and "Times request suspended"). A storage stress condition may not cause an SOS condition; CICS may be able to alleviate the condition. However, storage stress conditions are costly, and should be avoided.

The SOS condition is notified in the dynamic storage area statistics ("times went short on storage"), and is made apparent to the terminal user by external effects such as ceasing of polling and transaction initiation, and prolonged response times. In addition, a message is displayed on the operating system console when the short-on-storage (SOS) indication is detected. The SOS message, DFHSM0131 or DFHSM0133, indicates that:

- The amount of free space in a dynamic storage area is less than needed, and the associated DSA cannot be enlarged further (because the DSA limit has been reached).
- There are currently suspended GETMAIN requests waiting for large enough areas of contiguous storage to become available.

If there is insufficient storage in the relevant DSA limit to satisfy a GETMAIN request, the request is either queued (for unconditional requests) or a return code is given (for conditional requests).

Coding conventions for DSA limits

You can specify the size of the DSA limits as:

- A number of bytes
- A whole number of kilobytes
- A whole number of megabytes.

Use the letter K or M as a suffix to indicate whether the value represents a whole number of kilobytes, or a number of megabytes. For example, 2MB can be coded as either 2048K or 2M. (1KB = 1024 bytes; 1MB = 1024KB = 1048576 bytes.)

If the value you specify is not a multiple of 256KB for DSALIM, or 1MB for EDSALIM, CICS rounds up the value to the next multiple.

You cannot specify fractions of megabytes: you must code sizes in bytes or kilobytes. Some examples are shown in Table 40 on page 307:

Table 40. Examples of DSA limit values in bytes, kilobytes and megabytes

Coded as:					
bytes	2097152	3145788	3670016	4194304	4718592
kilobytes	2048K	3072K	3584K	4096K	4608K
megabytes	2M	3M	–	4M	–

For information about estimating the size of the dynamic storage areas, see the *CICS Performance Guide*.

The sample statistics program, DFH0STAT

You can use the statistics sample program, DFH0STAT, to help you determine and adjust values needed for CICS storage parameters, for example the size of the DSALIM and EDSALIM system initialization parameters. The program produces a report showing critical system parameters from the CICS Dispatcher, an analysis of the CICS Storage Manager and Loader statistics. The program demonstrates the use of EXEC CICS INQUIRE and EXEC CICS COLLECT STATISTICS commands to produce an analysis of your CICS system. You can use the sample program as provided or modify it to suit your needs.

The sample program consists of the following resources:

DFH0STAT	Statistics program, COBOL
DFH$STAS	Assembler language program called by DFH0STAT.
DFH$STCN	Assembler language program called by DFH0STAT.
DFH0STM	Mapset used by STAT transaction.
STAT	Transaction used to invoke DFH0STAT.

You can invoke the sample program during the PLT stage of CICS initialization or as a conversational transaction.

To use the sample program, you must:

1. Assemble and link-edit the BMS mapset DFH0STM. Include the physical mapset in a library that is in the LIBDEF PHASE,SEARCH chain for the CICS job. You can either include the symbolic map set in a user copy library, or insert it directly into the source for DFH0STAT.

   For more information about installing and using map sets, see Chapter 3, “Installing mapsets and partition sets” on page 15 and “Using BMS mapsets in application programs” on page 37.

2. Translate the DFH0STAT program source code, turning CICS commands into code understood by the compiler. The program source code is provided in the VSE/ESA sublibrary, PRD1.BASE.

   Note: You must use the translator option SP when translating DFH0STAT.

3. Compile the translator output for DFH0STAT to produce object code.

4. Link-edit the object module to produce a load module, which you store in an application load library that is specified in the LIBDEF PHASE,SEARCH chain for the CICS job.

5. Create resource definition entries, in the CSD, for the programs, the mapset, and the STAT transaction.

6. Define the system initialization parameter SPOOL=YES. This specifies that you need support for the system spooling interface.

For more information about installing programs, see Chapter 4, “Installing application programs” on page 33.

Part 5. Appendix

Appendix A. System initialization parameters grouped by functional area

The following table is provided as a guide to system initialization parameters, related by function (for example, XRF or intersystem communication). By using this table as a guide, you should find it easier to ensure that you code ***all*** the parameters that are needed for a particular CICS function.

Note: A check mark (√) in column two indicates that the parameters are read from the SIT directly by the CICS component that uses them, and are not obtained through the parameter manager domain interface. For more information about the parameter manager domain, see "The CICS parameter manager domain" on page 199.

Table 41 (Page 1 of 3). System initialization parameters grouped by functional area

Functional group		**System initialization keywords**
Application considerations	√	CMDPROT, CWAKEY, DISMACP, INITPARM, LGNMSG, OPERTIM, TCTUALOC, TCTUAKEY
Autoinstall for VTAM terminals and APPC connections	√	AIEXIT, AILDELAY, AIQMAX, AIRDELAY
Autoinstall for programs		PGAICTLG, PGAIEXIT, PGAIPGM
Basic mapping support	√	BMS, PGCHAIN, PGCOPY, PGPURGE, PGRET, PRGDLAY, SKRxxxx
Data interchange	√	DIP
DL/I	√	DLI|DL1, DLIOER
Dispatcher functions		ICV, ICVTSD, MXT, PRTYAGE
Dump functions		DUMP, DUMPDS, DUMPSW, SYDUMAX, TRDUMAX TRTRANSZ, TRTRANTY
Dynamic transaction backout	√	DBP, DBUFSZ
Exits	√	TBEXITS, TRAP
Extended recovery facility	√	ADI, AUTCONN, CLT, PDI, TAKEOVR, XRF, XRFSOFF, XRFSTME, XSWITCH, XRFTODI
Files (user)		FCT
Front end programming interface		FEPI
Intersystem communication and multiregion operation	√	APPLID, DTRPGM, DTRTRAN, DTRPGM, FSSTAFF, IRCSTRT, ISC, MROBTCH, MROLRM, SYSIDNT, (For ISC, see also VTAM group.)
Journaling	√	AKPFREQ, JCT, JSTATUS
Language Environment support	√	RUWAPOOL

Table 41 (Page 2 of 3). *System initialization parameters grouped by functional area*

Functional group		**System initialization keywords**
Loading programs		PGAICTLG, PGAIPGM, PGAIEXIT, PLTPI, PLTPISEC, PLTPIUSR, PRVMOD, SVA
Miscellaneous	√	DATFORM, FLDSEP, FLDSTRT, MSGCASE, MSGLVL, PRINT, SPOOL, STATRCD
Monitoring		MCT, MN, MNCONV, MNFREQ, MNSYNC, MNTIME, MNEXC, MNPER
National Language support		NATLANG
Persistent Session support	√	PSDINT, RMTRAN
RDO: file control attributes for the CSD		CSDACC, CSDBUFND, CSDBUFNI, CSDFRLOG, CSDJID, CSDLSRNO, CSDRECOV, CSDSTRNO
Security	√	CMDSEC, CMDPROT, CONFDATA, CONFTXT, DFLTUSER, ESMEXITS, PLTPISEC, PLTPIUSR, RESSEC, SEC, SECPRFX, XAPPC, XCMD, XDCT, XFCT, XJCT, XPCT, XPPT, XPSB, XTRAN, XTST, XUSER
Signon		SNSCOPE, USRDELAY (See also Security, and Terminal and LU management.)
Storage management		CDSASZE, CHKSTRM, CHKSTSK, CMDPROT, CWAKEY, DSASLIM, ECDSASZE, EDSALIM, ERDSASZE, ESDSASZE, EUDSASZE, RDSASZE, RENTPGM, SDSASZE, STGPROT, STGRCVY, TCTUALOC, TCTUAKEY, UDSASZE
System initialization		GRPLIST, INITPARM, NEWSIT, PARMERR, PLTPI, PLTPIUSR, PLTPISEC, SIT, START, STARTER, SUFFIX, WRKAREA
System recovery	√	SRT
System termination	√	PLTSD, XLT
Temporary storage	√	TS, TSMGSET, TST

Table 41 (Page 3 of 3). System initialization parameters grouped by functional area

Functional group		System initialization keywords
Terminal and LU management	√	APPLID, CLSDSTP, DSHIPIDL, DSHIPINT, EODI, FLDSEP, FLDSTRT, GMTEXT, GMTRAN, GNTRAN, ICVTSD, LGNMSG, OPERTIM, OPNDLIM, PRINT, PVDELAY RAMAX, RAPOOL, RESP, RMTRAN, TCP, TCSACTN, TCSWAIT, TCT, TCTUALOC, TCTUAKEY, VTAM, VTPREFIX (See also autoinstall for VTAM terminals.)
Trace		AUXTR, AUXTRSW, INTTR, TRTABSZ, SPCTR, SPCTRxx, STNTR, STNTRxx, SYSTR, USERTR
Transient data	√	DCT, TD
Timer		ICP, ICVR, OPERTIM

Appendix B. CICS modules eligible for the VSE Shared Virtual Area (SVA)

This chapter provides information about the CICS modules that are required in the VSE Shared Virtual Area (SVA), and the other CICS modules that are eligible for the SVA. This information is intended to help you plan for and install CICS modules in the SVA for the functions that your CICS regions use.

The following terms are used in this chapter:

VSE SVA	The VSE shared virtual area
SVA	The area of the SVA below the 16MB line
ESVA	The area of the SVA above the 16MB line

For more information about installing CICS modules into the SVA, and about controlling their use from the SVA, see Chapter 5, "CICS programs in the VSE shared virtual area" on page 59.

Modules eligible for SVA residence

The following information is provided in Table 42 on page 316 and Table 43 on page 316. Some of the information applies only to modules listed in Table 43 on page 316.

Term	**Meaning**
Name	The name of the module
Description	A brief description of the module. This gives some clues to the associated function which is useful if the module does not have a controlling function.
SVA/ESVA	(Table 43 on page 316 only). In this column, the terms SVA and ESVA are used to indicate whether a module will be loaded into the part of the VSE shared virtual area below the 16MB line(the SVA) or above the 16MB line (the ESVA).
Priority	(Table 43 on page 316 only.) A nominal "priority" is assigned in this column to help you to decide whether a module should reside in the SVA, or to help you choose between modules should you be short of space.
Size	The size of the module.
Notes	Information about matters such as the use of modules from the VSE shared virtual area, associated CICS options that need to be specified for the function that uses that particular module, and so on will be noted in the list starting on page 333. Unless otherwise stated, the options are specified by system initialization parameters as defined in the *CICS System Definition Guide*.

Priority

A nominal “priority” is assigned to modules eligible for residence in the VSE shared virtual area. The meanings of these priorities are as follows:

1. Mandatory. Modules assigned a priority of 1 **must** reside the VSE shared virtual area. Information about these modules is given in Table 42 on page 316.
2. Modules assigned a priority of 2 are generally good candidates for inclusion in the VSE shared virtual area. You should include these modules in the SVA to support the associated function.
3. Modules assigned a priority of 3 are good candidates for inclusion in the VSE shared virtual area. You should include these modules in the SVA if you use the associated function heavily.

Size

The module sizes are taken from the latest information available at the time of publishing, However, these sizes may be different in your CICS environment depending on the options you have selected, and if any PTFs you have applied affect the modules. The sizes are quoted here to help you calculate the amount of storage you need for the modules you want to install in the SVA. You can obtain the actual sizes of these modules from:

- A directory listing of the modules
- The module index provided at the back of a formatted SDUMP (taken with the SVA=NO system initialization parameter specified).

Table 42. Modules that must reside in the VSE shared virtual area

Name	Description	SVA/ ESVA	Size	Option/Note (See page 333)
DFHCDDAN	The CPC dead data anchor block	ESVA	8	-
DFHCSEOT	CICS EOJ clean up routine	SVA	400	-
DFHDSPEX	DS domain - VSE POST exit stub	ESVA	200	-
DFHCSVC	CICS SVC startup	ESVA	2224	(3)
DFHDTSAN	Shared data tables: system anchor block	ESVA	32	-
DFHDTSVC	Shared data tables: SVC module	ESVA	10592	-
DFHIRP	Interregion communication program	SVA	16504	(6)
DFHIRW10	IRC work delivery exit program	ESVA	1328	
DFHSCTE	The subsystem control table extension	SVA	16	-

Table 43 (Page 1 of 18). Modules eligible for inclusion in VSE shared virtual area

Name	Description	SVA/ ESVA	Priority	Size	Option/Note (See page 333)
DFHAFMT	AFCT manager	ESVA	3	7808	(2)
DFHAIIN	AITM Manager initialization	ESVA	3	2224	AIEXIT

Table 43 (Page 2 of 18). Modules eligible for inclusion in VSE shared virtual area

Name	**Description**	**SVA/ ESVA**	**Priority**	**Size**	**Option/Note (See page 333)**
DFHAIIQ	AITMM - locate/unlock/inquire/browse	ESVA	2	1520	AIEXIT
DFHAIP	Application Interface program	SVA	2	11280	-
DFHAIRP	AITMM - initialization/recovery	ESVA	3	1616	-
DFHAITM	AITMM - add replace/delete	ESVA	3	3256	AIEXIT
DFHALP	Terminal allocation	ESVA	2	22928	AIEXIT
DFHAPATT	AP domain - entrypoint attach	ESVA	2	744	-
DFHAPDM	AP domain - initialization/termination	ESVA	3	5432	-
DFHAPDN	AP domain - transaction definition notify	ESVA	3	2928	-
DFHAPEP	AP domain - user exit service	ESVA	2	11224	-
DFHAPIN	AP domain - special initialization for programs and user-replaceable modules	ESVA	2	184	-
DFHAPIQ	AP domain - user exit data access service	ESVA	3	1256	-
DFHAPJC	AP domain - journal control gate service	ESVA	3	2504	-
DFHAPLI	AP domain - language interface program	ESVA	2	24536	-
DFHAPNT	AP domain - MXT notify gate	ESVA	3	1128	-
DFHAPPG	AP domain - optimize initial_link for	EVPA	2	1928	-
DFHAPRM	AP domain - transaction attach/detach	ESVA	2	3088	-
DFHAPRT	AP Domain - route transaction gate	ESVA	3	9384	-
DFHAPSI	AP domain - initialization	ESVA	2	8184	
DFHAPSTL	AP domain - statistics collection program	ESVA	2	42928	-
DFHAPTD	AP domain - transient data gate service	ESVA	2	2800	DCT=YES|xx
DFHAPTI	AP domain - timer notify gate	ESVA	2	1112	-
DFHAPTIM	AP domain - interval control midnight task	ESVA	3	1632	

Table 43 (Page 3 of 18). Modules eligible for inclusion in VSE shared virtual area

Name	Description	SVA/ ESVA	Priority	Size	Option/Note (See page 333)
DFHAPTIX	AP domain - expiry analysis task	ESVA	2	1104	-
DFHAPXM	AP domain - transaction initialization and termination services	SVA	2	4592	-
DFHAPXME	AP domain - XM exception handler	ESVA	3	2704	-
DFHASV	Authorized services interface	SVA	2	1976	-
DFHCCNV	Data conversion for CICS OS/2 ISC users	ESVA	2	79312	(1)
DFHCEGN	Goodnight transaction stub	ESVA	3	2784	(1)
DFHCETRA	CETR transaction - main program	ESVA	3	12416	(1)
DFHCETRB	CETR transaction - component flag INQ/SET	ESVA	3	13976	(1)
DFHCETRC	CETR transaction - terminal and transaction	ESVA	3	14312	(1)
DFHCETRD	CETR transaction - subroutines	ESVA	3	4128	(1)
DFHCHS	CICS/VSE mirror for CICS OS/2 and CICS/VM™	ESVA	2	9128	(1)
DFHCLS3	LU6.2 services manager	ESVA	3	3864	(1)
DFHCLS4	LU6.2 password expiration	ESVA	3	5088	(1)
DFHCMP	CICS monitoring compatibility interface	ESVA	2	504	-
DFHCNVJP	Data conversion table - Japanese	ESVA	3	159088	(1)
DFHCNVKO	Data conversion table - Korean	ESVA	3	406176	(1)
DFHCNVSC	Data conversion table - Simplified Chinese	ESVA	3	233,504	(1)
DFHCNVTC	Data conversion table - Traditional Chinese	ESVA	3	262432	(1)
DFHCPIC	SAA communications interface program	ESVA	2	177720	-
DFHCPIN	CPI initialization program	ESVA	3	2792	-
DFHCPIRR	SAA resource recovery interface program	ESVA	2	1200	-
DFHCPSM	CICS translator special case code for CICSPLex SM	SVA	3	3296	
DFHCRC	Interregion cleanup program	EVA	3	352	ISC=YES

Table 43 (Page 4 of 18). Modules eligible for inclusion in VSE shared virtual area

Name	Description	SVA/ ESVA	Priority	Size	Option/Note (See page 333)
DFHCRNP	Interregion connection manager	ESVA	2	10616	(1)
DFHCRQ	ATI purge program	ESVA	2	880	(1)
DFHCRR	Interregion session recovery program	ESVA	3	2160	(1)
DFHCRS	Remote scheduler program	ESVA	2	5848	(1)
DFHCRSP	CICS IRC startup module	ESVA	3	3072	(1)
DFHCRT	Transaction routing relay program for APPC devices	ESVA	2	800	(1)
DFHCXCU	XRF catchup transaction	ESVA	3	712	XRF=YES (1)
DFHCXPA	Report Controller - Notify transaction	ESVA	3	800	SPOOL=YES
DFHCXPB	Report Controller - Connect any transaction	ESVA	3	1160	SPOOL=YES
DFHDBP1$	Dynamic transaction backout program	ESVA	2	5024	(1)
DFHDCP	Dump control program	ESVA	3	856	-
DFHDES	DES data encryption module	ESVA	3	4928	ISC=YES
DFHDIP	Data interchange program	ESVA	2	4048	DIP=YES
DFHDIPDY	Data interchange program (dummy)	ESVA	2	176	DIP=NO
DFHDLRP	DL/I restart program	SVA	3	1096	DLI=YES
DFHDMRM	CSD open/close program	ESVA	3	824	-
DFHDSAUT	DS domain - authorized services	ESVA	2	1992	-
DFHDSBA$	BMS data stream build (standard)	ESVA	2	1600	BMS= STANDARD
DFHDSB1$	BMS data stream build (full)	ESVA	2	1600	BMS=FULL
DFHDTAM	Shared data tables: access manager	ESVA	2	11984	(7)
DFHDTAOR	Shared data tables: AOR module	ESVA	2	3320	(7)
DFHDTCV	Shared data tables connection validation	ESVA	2	296	(7)
DFHDTFOR	Shared data tables: FOR module	ESVA	2	13992	(7)
DFHDTINS	Shared data tables: initialization	ESVA	3	760	(7)
DFHDTXS	Shared data tables connection security	ESVA	3	1888	(7)
DFHDUIO	DU domain - open/close/switch/write	SVA	2	6104	-

Table 43 (Page 5 of 18). Modules eligible for inclusion in VSE shared virtual area

Name	Description	SVA/ ESVA	Priority	Size	Option/Note (See page 333)
DFHDUSVC	DU domain - SVC processing routine	ESVA	2	2408	-
DFHDYP	Dynamic routing program	ESVA	2	320	DTRPGM= DFHDYP (1)
DFHEBF	EXEC interface for built-in functions	ESVA	3	320	
DFHEBU	EXEC FMH construction	ESVA	2	440	ISC=YES|xx
DFHECID	CECI service program	ESVA	3	78808	(1)
DFHECIP	Command interpeter (CECI) program	ESVA	3	3064	(1)
DFHECSP	Command syntax check (CECS) program	ESVA	3	3064	(1)
DFHEDAD	RDO (CEDA) service program	ESVA	3	121600	(1)
DFHEDAP	RDO (CEDA) program	ESVA	3	3200	(1)
DFHEDC	EXEC interface for dump control	ESVA	2	160	-
DFHEDCP	EXEC interface for dump system/transaction	ESVA	3	3816	-
DFHEDFBR	Temporary-storage browse transaction, CEBR	ESVA	3	12480	(1)
DFHEDFD	EDF display program	ESVA	3	65832	(1)
DFHEDFE	EDF attach error handler	ESVA	3	1376	(1)
DFHEDFP	EDF control program	ESVA	3	7592	(1)
DFHEDFR	EDF response table	ESVA	3	600	(1)
DFHEDFX	EDF task switch program	ESVA	3	4400	(1)
DFHEDI	EXEC interface for data interchange	ESVA	2	1360	DIP=YES
DFHEEI	EXEC interface for HANDLE, ADDRESS, ASSIGN	ESVA	2	6904	-
DFHEEX	EXEC FMH extraction	ESVA	2	760	-
DFHEFRM	EXEC file control syncpoint processor	ESVA	2	1248	-
DFHEGL	EXEC interface for unmapped LU6.2 commands	ESVA	2	3672	VTAM=YES
DFHEIACQ	EXEC ACQUIRE TERMINAL	ESVA	3	1472	-
DFHEICRE	EXEC CICS CREATE	ESVA	2	66976	-
DFHEIDLI	DL/I load table	SVA	3	9680	DLI

Table 43 (Page 6 of 18). Modules eligible for inclusion in VSE shared virtual area

Name	Description	SVA/ ESVA	Priority	Size	Option/Note (See page 333)
DFHEIDTI	EXEC ask-time, format-time program	ESVA	2	3344	-
DFHEIGDS	Translator table (GDS commands)	SVA	3	2784	(1)
DFHEIGDX	EXEC interface load table	SVA	3	3136	-
DFHEIIC	EXEC interface IC module	ESVA	2	9096	-
DFHEIPRT	EXEC interface for perform resettime	ESVA	3	688	-
DFHEIPSE	EXEC interface for perform security	ESVA	3	904	SEC=YES
DFHEIPSH	EXEC interface for perform shutdown	ESVA	3	2296	-
DFHEIQDN	EXEC inquire/set for external data sets	ESVA	3	4000	-
DFHEIQDS	EXEC inquire/set/discard for files	ESVA	3	14888	-
DFHEIQDU	EXEC inquire/set for dump data sets and dump codes	ESVA	3	7944	-
DFHEIQIR	EXEC inquire/set for IRC	ESVA	3	1944	-
DFHEIQMS	EXEC inquire/set for monitor and stats	ESVA	3	13472	-
DFHEIQMT	EXEC inquire/set for CEMT-only commands	ESVA	3	3432	-
DFHEIQPF	EXEC inquire/discard for profiles	ESVA	3	1888	
DFHEIQPN	EXEC inquire/discard for partner	ESVA	3	2536	
DFHEIQRQ	EXEC inquire for queued requests (REQIDs)	ESVA	3	3240	-
DFHEIQSA	EXEC inquire/set for system attributes	ESVA	3	7984	-
DFHEIQSC	EXEC inquire/set for connections	ESVA	3	7544	-
DFHEIQSJ	EXEC inquire/set for journals	ESVA	3	3952	-
DFHEIQSK	EXEC inquire/set for tasks	ESVA	3	13848	-
DFHEIQSM	EXEC inquire/set for modenames	ESVA	3	4056	-
DFHEIQSP	EXEC inquire/set/discard for programs	ESVA	3	6588	-
DFHEIQSQ	EXEC inquire/set for TD queues	ESVA	3	6360	-
DFHEIQST	EXEC inquire/set for terminals	ESVA	3	18408	-

Table 43 (Page 7 of 18). Modules eligible for inclusion in VSE shared virtual area

Name	Description	SVA/ ESVA	Priority	Size	Option/Note (See page 333)
DFHEIQSV	EXEC inquire/set for volumes	ESVA	3	312	
DFHEIQSX	EXEC inquire/set/discard for transactions	ESVA	3	7160	-
DFHEIQSZ	EXEC CICS SPI commands for FEPI	ESVA	3	3800	-
DFHEIQTM	EXEC inquire for autoinstmodel	ESVA	3	2032	
DFHEIQTR	EXEC inquire/set for trace	ESVA	3	10128	-
DFHEIQTS	EXEC inquire for tsqueue	ESVA	3	3256	
DFHEIQUE	EXEC inquire for exit programs	ESVA	3	5648	-
DFHEIQVT	EXEC inquire/set for VTAM and autoinstall	ESVA	3	5808	-
DFHEITAB	Translator table (API commands)	SVA	3	48448	(1)
DFHEITBS	Translator table (SPI commands)	SVA	3	41256	(1)
DFHEITHG	EXEC interface hired gun lookup table	SVA	2	14440	-
DFHEITMT	Command language table for CEMT	ESVA	3	27112	(1)
DFHEITOT	Command language table for CEOT	ESVA	3	1264	(1)
DFHEITST	CEST language definition table	ESVA	3	3896	(1)
DFHEITSZ	EXEC CICS language definition table	SVA	3	8696	(1)
DFHEJC	EXEC interface for journal control	ESVA	2	768	JCT=YES
DFHEKC	EXEC interface for task control	ESVA	2	1096	-
DFHEMEX	EXEC interface for ME domain	ESVA	3	2800	-
DFHEMS	EXEC interface for BMS	ESVA	2	4360	BMS
DFHEMSJB	CEMS/CEOS transaction: jobs	ESVA	3	18600	SPOOL=YES (1)
DFHEMSP	CEMS/CEOS transaction: main program	ESVA	3	11704	SPOOL=YES (1)
DFHEMSPR	CEMS/CEOS transaction: printers	ESVA	3	21888	SPOOL=YES (1)
DFHEMSRE	CEMS/CEOS transaction: reports	ESVA	3	42000	SPOOL=YES (1)

Table 43 (Page 8 of 18). Modules eligible for inclusion in VSE shared virtual area

Name	Description	SVA/ ESVA	Priority	Size	Option/Note (See page 333)
DFHEMSTD	CEMS/CEOS transaction: td queues	ESVA	3	17216	SPOOL=YES (1)
DFHEMTA	Programmable interface to Master terminal program	ESVA	3	3288	(1,4)
DFHEMTD	Master terminal (CEMT) service program	ESVA	3	94448	(1)
DFHEMTP	Master terminal (CEMT) program	ESVA	3	3288	(1)
DFHEOP	EXEC interface for write operator	ESVA	3	2792	-
DFHEOTP	CEOT service program	ESVA	3	3288	(1)
DFHEPC	EXEC interface for program control	ESVA	2	8584	-
DFHEPS	System spooling interface stub	ESVA	2	2856	SPOOL=YES
DFHERM	Resource manager interface (RMI) module	SVA	2	12944	-
DFHESC	EXEC interface for storage control	ESVA	2	1328	-
DFHESE	EXEC interface for query security	ESVA	2	4552	-
DFHESN	EXEC interface for signon and sign-off	ESVA	2	5032	-
DFHESP	EXEC interface for syncpoint control	ESVA	2	856	-
DFHESTP	CEST service program	ESVA	3	3288	(1)
DFHESZ	EXEC CICS API commands for FEPI	ESVA	3	1088	-
DFHETC	EXEC interface for terminal control	ESVA	2	7488	-
DFHETD	EXEC interface for transient data	ESVA	2	2400	DCT=YES|xx
DFHETL	LU6.2 EXEC interface stub	ESVA	2	7928	-
DFHETR	EXEC interface for trace control	ESVA	2	800	(5)
DFHETRX	EXEC interface for enter tracenum, monitor	ESVA	2	1240	USERTR
DFHETS	EXEC interface for temporary storage	ESVA	2	2520	TST=YES|xx
DFHEVAS	CICS macro emulation	ESVA	2	6088	
DFHEVBBF	CICS closely coupled interface module	ESVA	2	7536	
DFHEVCL	CICS sequential I/O close simulation	ESVA	2	15864	

Table 43 (Page 9 of 18). Modules eligible for inclusion in VSE shared virtual area

Name	Description	SVA/ ESVA	Priority	Size	Option/Note (See page 333)
DFHEVID0	JC subtask stub	ESVA	2	56	
DFHEVID1	KETCB subtask stub	ESVA	2	56	
DFHEVID2	TRTCB subtask stub	ESVA	2	56	
DFHEVOP	CICS sequential I/O OPEN simulation	ESVA	2	101040	
DFHEVORJ	CICS RDJFCB macro simulation	ESVA	2	624	
DFHEVSDM	CICS sdumpx module	ESVA	2	832	
DFHFCAT	File control catalog manager	ESVA	2	4272	-
DFHFCBD	File control DAM request processor	SVA	2	5776	FCT=YES|xx
DFHFCBF	FILE control backout failure	ESVA	2	3824	
DFHFCDN	File control DSN block manager	ESVA	3	8352	FCT=YES|xx
DFHFCD2	File control shared data tables record request handler	ESVA	2	13680	FCT=YES|xx (7)
DFHFCEI	File control EXEC interface module	ESVA	2	10528	FCT=YES|xx
DFHFCFR	File control file request handler	ESVA	2	6760	FCT=YES|xx
DFHFCFS	File control file state program	ESVA	2	54040	FCT=YES|xx
DFHFCIN	File control initialization program	ESVA	3	1632	FCT=YES|xx
DFHFCMT	File control table manager	ESVA	3	11224	FCT=YES|xx
DFHFCRL	File control VSAM SHRCTL block manager	ESVA	3	3160	FCT=YES|xx
DFHFCRM	File control syncpoint processor	ESVA	2	2960	FCT=YES|xx
DFHFCRP	File control restart program	ESVA	3	15456	FCT=YES|xx
DFHFCSD	File control shutdown program	ESVA	2	1464	FCT=YES|xx
DFHFCST	File control statistics program	ESVA	3	7464	FCT=YES|xx
DFHFCU	File open utility program	SVA	3	592	FCT=YES|xx (1)
DFHFCVS	File access VSAM request processor	ESVA	2	31608	FCT=YES|xx
DFHFDP	Formatted dump program	ESVA	3	792	DUMP=YES
DFHGMM	VTAM LU startup message	ESVA	2	1704	(1)
DFHICP	Interval control program	ESVA	2	12760	-

Table 43 (Page 10 of 18). Modules eligible for inclusion in VSE shared virtual area

Name	Description	SVA/ ESVA	Priority	Size	Option/Note (See page 333)
DFHICXM	AP domain - bind, inquire, and release facility IC functions	ESVA	2	5840	-
DFHIIPA$	BMS non-3270 input mapping (standard)	ESVA	3	2056	BMS= STANDARD
DFHIIP1$	BMS non-3270 input mapping (full)	ESVA	3	2056	BMS=FULL
DFHISP	Intersystem communication program	ESVA	2	3336	ISC=YES
DFHJAP	Journal archive program	ESVA	3	12872	JCT=YES|xx
DFHJASP	Journal archive submission program	SVA	3	8368	JCT=YES|xx
DFHJCBSP	JOURNAL control bootstrap progam	ESVA	2	1064	JCT=YES|xx
DFHJCC	Journal control close	ESVA	3	2472	JCT=YES|xx
DFHJCOAT	Journal control tape open-ahead	ESVA	3	272	JCT=YES|xx
DFHJCP	JOURNAL control program	SVA	2	12216	JCT=YES|xx
DFHJCPDY	Journal control program (dummy)	ESVA	2	472	JCT=NO
DFHJCRM	Journal control syncpoint processor	ESVA	2	2056	JCT=YES|xx
DFHJCSDJ	JOURNAL control shutdown program	ESVA	2	1712	JCT=YES|xx
DFHKCP	Transaction manager startup routine	ESVA	2	12896	-
DFHKCSC	DFHKCQ chain scanning for discard	ESVA	3	1096	-
DFHLDDMI	LD domain - secondary initialization	ESVA	3	18920	-
DFHLDNT	LD domain - storage notify handler	ESVA	2	2552	-
DFHLDST	LD domain - statistics collection	ESVA	3	3632	-
DFHLDSVC	LD domain - authorized service routine	SVA	2	2080	-
DFHLIRET	Language interface return program	SVA	2	176	-
DFHMCPA$	BMS mapping control program (standard)	ESVA	2	8568	BMS= STANDARD
DFHMCPE$	BMS mapping control program (minimum)	ESVA	2	8208	BMS= MINIMUM
DFHMCP1$	BMS mapping control program (full)	ESVA	2	13480	BMS=FULL

Table 43 (Page 11 of 18). Modules eligible for inclusion in VSE shared virtual area

Name	Description	SVA/ ESVA	Priority	Size	Option/Note (See page 333)
DFHMCX	BMS fast path module	ESVA	2	8280	BMS
DFHMCY	BMS mapping control(MAPPINGDEV requests)	ESVA	3	6904	BMS
DFHMET1C	BASE messages link-edit module - Chinese	ESVA	3	237952	NATLANG=C
DFHMET1E	Base messages link-edit module - English	ESVA	2	260528	NATLANG=E
DFHMET1G	BASE messages link-edit module - German	ESVA	3	262776	NATLANG=G
DFHMET1K	BASE messages link-edit module - Kanji	ESVA	3	270872	NATLANG=K
DFHMGP	Message writer program	ESVA	3	14624	-
DFHMGT	Message generation table	ESVA	3	23288	-
DFHMIRS	Function Shipping mirror program	ESVA	2	4744	ISC=YES (1)
DFHML1	BMS LU1 printer mapping program	ESVA	2	5168	BMS
DFHMNDML	MN domain - initialization/termination	ESVA	2	72008	-
DFHMNSVC	MN domain - authorized service routine	ESVA	2	5016	-
DFHMROQP	IRC work queue manager	ESVA	3	1320	ISC=YES
DFHMSP	Message switching program	ESVA	2	12648	(1)
DFHMXP	Local queuing shipper	ESVA	2	1184	(1)
DFHM32A$	BMS 3270 mapping (standard)	ESVA	2	7752	BMS= STANDARD
DFHM321$	BMS 3270 mapping (full)	ESVA	2	7752	BMS=FULL
DFHPBPA$	BMS page and text build (standard)	ESVA	2	8704	BMS= STANDARD
DFHPBP1$	BMS page and text build (full)	ESVA	2	9512	BMS=FULL
DFHPCP	Program control program	ESVA	2	2512	-
DFHPCPC2	DOS/VS COBOL interface	SVA	3	104	
DFHPGADX	Program autoinstall exit - Assembler	ESVA	2	200	(1)
DFHPGDM	PG domain - initialize, quiesce, and terminate domain functions	ESVA	2	155976	-
DFHPGRP	PG domain - recovery program	ESVA	2	13112	-
DFHPHP	Partition handling program	ESVA	2	2272	BMS

Table 43 (Page 12 of 18). Modules eligible for inclusion in VSE shared virtual area

Name	Description	SVA/ ESVA	Priority	Size	Option/Note (See page 333)
DFHPRCM	Partner resource manager command interface	ESVA	3	1384	-
DFHPRFS	Partner resource manager interface to SAA communications interface	ESVA	3	656	-
DFHPRIN	Partner initialization load program	ESVA	3	3384	-
DFHPRPT	Partner resource table (PRT) manager	ESVA	3	3080	-
DFHPRRP	Partner recovery program	ESVA	3	1528	
DFHPSBP	Report Controller: backout program	ESVA	3	2304	SPOOL=YES (1)
DFHPSEC	Report Controller error messages: Chinese	ESVA	3	8104	SPOOL=YES (1)
DFHPSEE	Report Controller error messages: English	ESVA	3	8104	SPOOL=YES (1)
DFHPSEG	Report Controller error messages: German	ESVA	3	8104	SPOOL=YES (1)
DFHPSEK	Report Controller error messages: Kanji	ESVA	3	8104	SPOOL=YES (1)
DFHPSIP	Report Controller initialization program	ESVA	3	752	SPOOL=YES (1)
DFHPSJC	Report Controller JCL report panels: Chinese	ESVA	3	13403	SPOOL=YES (1)
DFHPSJE	Report Controller JCL report panels: English	ESVA	3	13084	SPOOL=YES (1)
DFHPSJG	Report Controller JCL report panels: German	ESVA	3	13084	SPOOL=YES (1)
DFHPSJK	Report Controller JCL report panales: Kanji	ESVA	3	13343	SPOOL=YES (1)
DFHPSMC	Report Controller printer control panels: Chinese	ESVA	3	13068	SPOOL=YES (1)
DFHPSME	Report Controller printer control panels: English	ESVA	3	12776	SPOOL=YES (1)
DFHPSMG	Report Controller printer control panels: German	ESVA	3	12777	SPOOL=YES (1)
DFHPSMK	Report Controller printer control panels: Kanji	ESVA	3	13076	SPOOL=YES (1)
DFHPSNC	Report Controller print report panels: Chinese	ESVA	3	26933	SPOOL=YES (1)
DFHPSNE	Report Controller print report panels: English	ESVA	3	26351	SPOOL=YES (1)
DFHPSNG	Report Controller print report panels: German	ESVA	3	26351	SPOOL=YES (1)

Table 43 (Page 13 of 18). Modules eligible for inclusion in VSE shared virtual area					
Name	**Description**	**SVA/ ESVA**	**Priority**	**Size**	**Option/Note (See page 333)**
DFHPSNK	Report Controller print report panels: Kanji	ESVA	3	26820	SPOOL=YES (1)
DFHPSOP	Report Controller printer spooler output program	ESVA	3	43232	SPOOL=YES (1)
DFHPSP	System spooling interface program	SVA	2	69312	SPOOL=YES
DFHPSQC	Report Controller TD queue panels: Chinese	ESVA	3	7333	SPOOL=YES (1)
DFHPSQE	Report Controller TD queue panels: English	ESVA	3	7176	SPOOL=YES (1)
DFHPSQG	Report Controller TD queue panels: German	ESVA	3	7176	SPOOL=YES (1)
DFHPSQK	Report Controller TD queue panels: Kanji	ESVA	3	7338	SPOOL=YES (1)
DFHPSTEP	Report Controller writer task error Handling	ESVA	3	816	SPOOL=YES (1)
DFHPS0C	Report Controller generic operator panels: Chinese	ESVA	3	6708	SPOOL=YES (1)
DFHPS0E	Report Controller generic operator panels: English	ESVA	3	6540	SPOOL=YES (1)
DFHPS0G	Report Controller generic operator panels: German	ESVA	3	6540	SPOOL=YES (1)
DFHPS0K	Report Controller generic operator panels: Kanji	ESVA	3	6712	SPOOL=YES (1)
DFHQRY	Query transaction	ESVA	2	4032	(1)
DFHRCP	Recovery control program	ESVA	3	7560	-
DFHRLRA$	BMS route list resolution (standard)	ESVA	2	2048	BMS= STANDARD
DFHRLR1$	BMS route list resolution (full)	ESVA	2	3840	BMS=FULL
DFHRTC	CRTE cancel command processor	ESVA	2	872	(1)
DFHRTE	Transaction routing program	ESVA	2	2688	(1)
DFHSAIQ	AP domain - system data inquire & set	ESVA	2	2408	-
DFHSCAA	Language Environment for VSE/ESA - get common anchor area	SVA	2	128	-
DFHSFP	Sign-off program	ESVA	2	4688	(1)
DFHSIPLT	System initialization PLT processor	ESVA	2	5088	
DFHSIP31	System initialization program	ESVA	2	804496	

Table 43 (Page 14 of 18). Modules eligible for inclusion in VSE shared virtual area

Name	Description	SVA/ ESVA	Priority	Size	Option/Note (See page 333)
DFHSKP	Subtask management program	ESVA	2	6512	-
DFHSKTSK	General purpose subtask entry point	ESVA	3	48	-
DFHSMSVC	SM domain - authorized service routine	ESVA	3	11544	-
DFHSMTAB	CICSPLex SM language definition table	ESVA	3	17016	
DFHSNP	Signon program	ESVA	2	14704	(1)
DFHSNUS	US domain - local and remote signon	ESVA	2	52648	-
DFHSPP	Syncpoint program	ESVA	2	7536	-
DFHSPZ	Syncpoint resource manager	ESVA	2	24576	-
DFHSTDML	ST domain - initialization/termination	ESVA	3	30688	-
DFHSUSX	XRF signon	ESVA	2	9312	XRF=YES
DFHSUWT	WTO/WTOR interface subroutine	ESVA	3	7416	-
DFHSUZX	ZC trace controller	ESVA	3	6824	-
DFHSZATR	FEPI adaptor program	ESVA	3	17512	-
DFHSZRMP	FEPI resource manager	ESVA	3	219344	(1)
DFHTBSSP	Builder syncpoint processor	ESVA	2	14352	-
DFHTCRP	Terminal control recovery program	ESVA	3	24232	-
DFHTDP	Transient data program (macro entry)	SVA	2	16600	DCT=YES|xx
DFHTDQ	Transient data program (internal entry)	ESVA	2	19040	DCT=YES|xx
DFHTDRM	Transient data recovery manager processor	ESVA	2	1232	DCT=YES|xx
DFHTDRP	Transient data recovery program	ESVA	3	10864	DCT=YES|xx
DFHTDXM	XM domain - TD facility management services	ESVA	2	2984	-
DFHTFBF	Terminal facility manager bind facility functions	ESVA	2	11448	-
DFHTFIQ	Terminal facility manager inquire/set functions	ESVA	2	4712	-
DFHTFRF	Terminal facility manager release function	ESVA	2	4904	-
DFHTIDM	TI domain - initialization/termination	ESVA	3	9264	-

Table 43 (Page 15 of 18). Modules eligible for inclusion in VSE shared virtual area

Name	**Description**	**SVA/ ESVA**	**Priority**	**Size**	**Option/Note (See page 333)**
DFHTMP	Table manager program	ESVA	2	14824	-
DFHTON	Terminal object resolution module	ESVA	2	832	-
DFHTORP	Terminal object recovery program	ESVA	3	1288	-
DFHTPPA$	BMS terminal page processor (standard)	ESVA	2	3312	BMS= STANDARD
DFHTPP1$	BMS terminal page processor (full)	ESVA	2	4328	BMS=FULL
DFHTPQ	BMS terminal page cleanup program	ESVA	2	4072	BMS (1)
DFHTPR	BMS terminal page retrieval program	ESVA	2	22088	BMS (1)
DFHTPS	BMS terminal page scheduling program	ESVA	2	4640	BMS (1)
DFHTRAO	TR domain - auxiliary trace output	SVA	3	2008	AUXTR=ON
DFHTSP	Temporary-storage control program	ESVA	2	22792	TST=YES|xx
DFHTSUT	Temporary-storage unit table abstract type	ESVA	2	8288	-
DFHUCNV	User data conversion program	ESVA	3	440	(1)
DFHUEH	User exit handler (AP domain)	ESVA	2	8296	-
DFHUEM	User exit manager	ESVA	3	7880	-
DFHUSDM	US domain - initialize, quiesce, and terminate domain functions	ESVA	3	52256	-
DFHWKP	Warm Keypoint program	ESVA	3	7112	
DFHXCEIX	EXCI API handler	ESVA	3	9128	
DFHXCI	External CICS interface (EXCI) program	SVA	3	3264	-
DFHXCPRX	EXCI program request handler	ESVA	3	29640	
DFHXCSVC	EXCI SVC services	ESVA	3	640	-
DFHXCTAB	EXCI language table	SVA	3	552	-
DFHXFP	Online data transformation program	SVA	2	24136	ISC=YES
DFHXFX	Optimized data transformation program	ESVA	2	8136	ISC=YES
DFHXJCC	User-replaceable journal close exit	ESVA	3	224	JCT=YES|xx

Table 43 (Page 16 of 18). Modules eligible for inclusion in VSE shared virtual area

Name	**Description**	**SVA/ ESVA**	**Priority**	**Size**	**Option/Note (See page 333)**
DFHXJCO	User-replaceable journal open exit	ESVA	3	224	JCT=YES|xx
DFHXMAB	XM domain - abend handler	ESVA	2	320	-
DFHXMSG	Persistent session recovery message Map	ESVA	3	594	(1)
DFHXRCP	XRF console communication program	ESVA	3	8264	XRF=YES
DFHXRP	XRF request program	ESVA	2	9104	XRF=YES
DFHXRSP	XRF surveillance program	ESVA	2	4984	XRF=YES
DFHXSEAI	EXEC CICS early verification stub program	ESVA	3	34	SEC=YES
DFHXSS	XS domain - supervisor request services	ESVA	3	32608	SEC=YES
DFHXSWM	XRF message manager for security manager	ESVA	2	1744	XRF=YES
DFHXTP	Terminal sharing transformation program	ESVA	2	12008	ISC=YES
DFHZATA	Autoinstall program	ESVA	2	18568	(1)
DFHZATD	Autoinstall delete program	ESVA	2	6072	(1)
DFHZATDX	User-replaceable autoinstall exit	ESVA	2	392	AIEXIT (1)
DFHZATDY	User-replaceable autoinstall exit with APPC	ESVA	2	544	AIEXIT (1)
DFHZATMD	Autoinstall remote terminal time-out mass delete program	ESVA	3	3288	(1)
DFHZATMF	Autoinstall remote terminal mass flag program	ESVA	3	1672	(1)
DFHZATR	Autoinstall restart terminal deletion program	ESVA	3	1976	(1)
DFHZATS	Autoinstall remote install/delete program	ESVA	3	10616	(1)
DFHZBAN	Terminal control bind analysis	SVA	2	10344	-
DFHZCA	VTAM working set module	ESVA	2	10480	VTAM=YES
DFHZCB	VTAM working set module	ESVA	2	40072	VTAM=YES
DFHZCC	VTAM working set module	ESVA	2	59992	VTAM=YES
DFHZCGRP	APPC Persistent session initialization stub program	ESVA	3	105	VTAM=YES
DFHZCN1	CICS client CCIN transaction	ESVA	3	4816	(1)
DFHZCN2	CICS Client CCIN transaction ZC domain subroutine	ESVA	3	4696	

Table 43 (Page 17 of 18). Modules eligible for inclusion in VSE shared virtual area

Name	**Description**	**SVA/ ESVA**	**Priority**	**Size**	**Option/Note (See page 333)**
DFHZCOVR	CICS COVR (OPEN VTAM retry) transaction	ESVA	3	1544	VTAM=YES
DFHZCP	Terminal management program	ESVA	2	31208	VTAM=YES
DFHZCSTP	CSTP transaction attach stub	ESVA	3	656	
DFHZCT1	CICS Client CTIN transaction	ESVA	3	11016	
DFHZCUT	Persistent verification signed-on-from list management program	ESVA	2	5424	VTAM=YES
DFHZCW	VTAM nonworking set module	ESVA	3	7512	VTAM=YES
DFHZCX	LOCATE, ISC/IRC request	ESVA	2	32488	ISC=YES
DFHZCXR	Transaction routing module address list	ESVA	2	29032	ISC=YES
DFHZCY	VTAM nonworking set module	ESVA	3	69376	VTAM=YES
DFHZCZ	VTAM nonworking set module	ESVA	3	23112	VTAM=YES
DFHZGAI	APPC autoinstall - create APPC clones	ESVA	2	8096	AIEXIT
DFHZGBM	APPC manipulate bitmap	ESVA	2	5672	VTAM=YES
DFHZGCA	LU6.2 CNOS actioning	ESVA	3	6200	VTAM=YES
DFHZGCC	Catalog CNOS services	ESVA	3	2528	VTAM=YES
DFHZGCN	LU6.2 CNOS negotiation	ESVA	3	12360	VTAM=YES
DGHZGDA	APPC persistent session deallocate abend program	ESVA	3	5504	VTAM=YES
DFHZGPC	APPC CNOS value recovery	ESVA	3	6528	VTAM=YES
DFHZGPR	VTAM persistent sessions resource handler	ESVA	3	2896	VTAM=YES
DFHZGRP	APPC persistent session initialization program	ESVA	3	15664	VTAM=YES
DFHZGSL	APPC persistent session SETLOGON processing	ESVA	3	1896	VTAM=YES
DFHZGUB	APPC persistemt session termination program	ESVA	3	3336	VTAM=YES
DFHZLS1	LU6.2 CNOS request transaction program	ESVA	3	2200	VTAM=YES (1)
DFHZRSP	TermiNAl control RESYNC SEND program	ESVA	3	680	VTAM=YES (1)
DFHZXCU	XRF catch-up program	ESVA	3	11600	XRF=YES (1)

Table 43 (Page 18 of 18). Modules eligible for inclusion in VSE shared virtual area

Name	Description	SVA/ ESVA	Priority	Size	Option/Note (See page 333)
DFHZXRE	XRF terminal reconnection program	ESVA	3	3584	XRF=YES (1)

Notes:

(Module references to this note are generated automatically)

1. The program is used from the VSE shared virtual area if you set the USESVACOPY option of its program resource definition to YES.
2. The DFHAFMT program is used by ADD, DELETE, UPDATE, and INQUIRE commands for FILE resource definitions.
3. You must always install the latest service level of the CICS SVC module, DFHCSVC. You should install the DFHCSVC module into the VSE shared virtual area.
4. The use of this pre-CICS Transaction Server for VSE/ESA Release 1 programmable interface to the master terminal program, DFHEMTA, is supported for compatibility reasons only. You are strongly recommended to use the equivalent EXEC CICS INQUIRE|SET commands instead, and retain the old programmable interface only where no equivalent API command is available (for example, for CEMT INQUIRE|SET DLIDATABASE). The documentation for this interface is available only in the CICS libraries for the releases prior to CICS Transaction Server for VSE/ESA Release 1.
5. You can set the system tracing status by coding appropriate system initialization parameters, and you can also set it dynamically by using the CETR transaction.

   The system initialization parameters that you can use are:

Parameter	Use
AUXTR	Activate auxiliary trace.
AUXTRSW	Define the auxiliary switch status.
INTTR	Activate CICS internal tracing.
TRTABSZ	Specify the size of the internal trace table.
USERTR	Set the master user trace flag on or off.

   For information about using CICS trace, and using the CETR transaction to control the tracing status, see the *CICS Problem Determination Guide*.
6. The DFHIRP module needs to be in the VSE shared virtual area only if you are using MRO.

   You must always install the latest service level of the DFHIRP (if needed).

   If you are running CICS with MRO at different release levels, all regions in the same VSE-image must use the latest DFHIRP module.
7. The following modules, used by the Shared Data Tables facility, are eligible for residence in the VSE shared virtual area:

   DFHDTAM
   DFHDTAOR
   DFHDTCV
   DFHDTFOR

DFHDTINS
DFHDTSAN
DFHDTSVC
DFHDTXS
DFHFCD2

DFHDTSVC and DFHDTSAN **must** reside in the VSE shared virtual area. In addition, only DFHDTAM, DFHDTAOR, DFHDTFOR, DFHFCD2, and possibly DFHDTCV are used sufficiently frequently to be worth considering for the VSE shared virtual area.

Bibliography

CICS Transaction Server for VSE/ESA Release 1 library

Evaluation and planning	
Release Guide	GC33-1645
Migration Guide	GC33-1646
Report Controller Planning Guide	GC33-1941
General	
Master Index	SC33-1648
Trace Entries	SC34-5556
User's Handbook	SC34-5555
Glossary (softcopy only)	GC33-1649
Administration	
System Definition Guide	SC33-1651
Customization Guide	SC33-1652
Resource Definition Guide	SC33-1653
Operations and Utilities Guide	SC33-1654
CICS-Supplied Transactions	SC33-1655
Programming	
Application Programming Guide	SC33-1657
Application Programming Reference	SC33-1658
Sample Applications Guide	SC33-1713
Application Migration Aid Guide	SC33-1943
System Programming Reference	SC33-1659
Distributed Transaction Programming Guide	SC33-1661
Front End Programming Interface User's Guide	SC33-1662
Diagnosis	
Problem Determination Guide	GC33-1663
Messages and Codes Vol 3 (softcopy only)	SC33-6799
Diagnosis Reference	LY33-6085
Data Areas	LY33-6086
Supplementary Data Areas	LY33-6087
Communication	
Intercommunication Guide	SC33-1665
CICS Family: Interproduct Communication	SC33-0824
CICS Family: Communicating from CICS on System/390	SC33-1697
Special topics	
Recovery and Restart Guide	SC33-1666
Performance Guide	SC33-1667
Shared Data Tables Guide	SC33-1668
Security Guide	SC33-1942
External CICS Interface	SC33-1669
XRF Guide	SC33-1671
Report Controller User's Guide	GC33-1940
CICS Clients	
CICS Clients: Administration	SC33-1792
CICS Universal Clients Version 3 for OS/2: Administration	SC34-5450
CICS Universal Clients Version 3 for Windows: Administration	SC34-5449
CICS Universal Clients Version 3 for AIX: Administration	SC34-5348
CICS Universal Clients Version 3 for Solaris: Administration	SC34-5451
CICS Family: OO programming in C++ for CICS Clients	SC33-1923
CICS Family: OO programming in BASIC for CICS Clients	SC33-1671
CICS Family: Client/Server Programming	SC33-1435
CICS Transaction Gateway Version 3: Administration	SC34-5448

Books from VSE/ESA 2.4 base program libraries

VSE/ESA Version 2 Release 4

Book title	Order number
Administration	SC33-6705
Diagnosis Tools	SC33-6614
Extended Addressability	SC33-6621
Guide for Solving Problems	SC33-6710
Guide to System Functions	SC33-6711
Installation	SC33-6704
Licensed Program Specification	GC33-6700
Messages and Codes Volume 1	SC33-6796
Messages and Codes Volume 2	SC33-6798
Messages and Codes Volume 3	SC33-6799
Networking Support	SC33-6708
Operation	SC33-6706
Planning	SC33-6703
Programming and Workstation Guide	SC33-6709
System Control Statements	SC33-6713
System Macro Reference	SC33-6716
System Macro User’s Guide	SC33-6715
System Upgrade and Service	SC33-6702
System Utilities	SC33-6717
TCP/IP User's Guide	SC33-6601
Turbo Dispatcher Guide and Reference	SC33-6797
Unattended Node Support	SC33-6712

High-Level Assembler Language (HLASM)

Book title	Order number
General Information	GC26-8261
Installation and Customization Guide	SC26-8263
Language Reference	SC26-8265
Programmer’s Guide	SC26-8264

Language Environment for VSE/ESA (LE/VSE)

Book title	Order number
C Run-Time Library Reference	SC33-6689
C Run-Time Programming Guide	SC33-6688
Concepts Guide	GC33-6680
Debug Tool for VSE/ESA Fact Sheet	GC26-8925
Debug Tool for VSE/ESA Installation and Customization Guide	SC26-8798
Debug Tool for VSE/ESA User's Guide and Reference	SC26-8797
Debugging Guide and Run-Time Messages	SC33-6681
Diagnosis Guide	SC26-8060
Fact Sheet	GC33-6679
Installation and Customization Guide	SC33-6682
LE/VSE Enhancements	SC33-6778
Licensed Program Specification	GC33-6683
Programming Guide	SC33-6684
Programming Reference	SC33-6685
Run-Time Migration Guide	SC33-6687
Writing Interlanguage Communication Applications	SC33-6686

VSE/ICCF

Book title	Order number
Adminstration and Operations	SC33-6738
User's Guide	SC33-6739

VSE/POWER

Book title	Order number
Administration and Operation	SC33-6733
Application Programming	SC33-6736
Networking Guide	SC33-6735
Remote Job Entry User's Guide	SC33-6734

VSE/VSAM

Book title	Order number
Commands	SC33-6731
User's Guide and Application Programming	SC33-6732

VTAM for VSE/ESA

Book title	Order number
Customization	LY43-0063
Diagnosis	LY43-0065
Data Areas	LY43-0104
Messages and Codes	SC31-6493
Migration Guide	GC31-8072
Network Implementation Guide	SC31-6494
Operation	SC31-6495
Overview	GC31-8114
Programming	SC31-6496
Programming for LU6.2	SC31-6497
Release Guide	GC31-8090
Resource Definition Reference	SC31-6498

Books from VSE/ESA 2.4 optional program libraries

C for VSE/ESA (C/VSE)

Book title	Order number
C Run-Time Library Reference	SC33-6689
C Run-Time Programming Guide	SC33-6688
Diagnosis Guide	GC09-2426
Installation and Customization Guide	GC09-2422
Language Reference	SC09-2425
Licensed Program Specification	GC09-2421
Migration Guide	SC09-2423
User’s Guide	SC09-2424

COBOL for VSE/ESA (COBOL/VSE)

Book title	Order number
Debug Tool for VSE/ESA Fact Sheet	GC26-8925
Debug Tool for VSE/ESA Installation and Customization Guide	SC26-8798
Debug Tool for VSE/ESA User’s Guide and Reference	SC26-8797
Diagnosis Guide	SC26-8528
General Information	GC26-8068
Installation and Customization Guide	SC26-8071
Language Reference	SC26-8073
Licensed Program Specifications	GC26-8069
Migration Guide	GC26-8070
Migrating VSE Applications To Advanced COBOL	GC26-8349
Programming Guide	SC26-8072

DB2 Server for VSE

Book title	Order number
Application Programming	SC09-2393
Database Administration	GC09-2389
Installation	GC09-2391
Interactive SQL Guide and Reference	SC09-2410
Operation	SC09-2401
Overview	GC08-2386
System Administration	GC09-2406

DL/I VSE

Book title	Order number
Application and Database Design	SH24-5022
Application Programming: CALL and RQDLI Interface	SH12-5411
Application Programming: High-Level Programming Interface	SH24-5009
Database Administration	SH24-5011
Diagnostic Guide	SH24-5002
General Information	GH20-1246
Guide for New Users	SH24-5001
Interactive Resource Definition and Utilities	SH24-5029
Library Guide and Master Index	GH24-5008
Licensed Program Specifications	GH24-5031
Low-level Code and Continuity Check Feature	SH20-9046
Library Guide and Master Index	GH24-5008
Messages and Codes	SH12-5414
Recovery and Restart Guide	SH24-5030
Reference Summary: CALL Program Interface	SX24-5103
Reference Summary: System Programming	SX24-5104
Reference Summary: HLPI Interface	SX24-5120
Release Guide	SC33-6211

PL/I for VSE/ESA (PL/I VSE)

Book title	Order number
Compile Time Messages and Codes	SC26-8059
Debug Tool For VSE/ESA User’s Guide and Reference	SC26-8797
Diagnosis Guide	SC26-8058
Installation and Customization Guide	SC26-8057
Language Reference	SC26-8054
Licensed Program Specifications	GC26-8055
Migration Guide	SC26-8056
Programming Guide	SC26-8053
Reference Summary	SX26-3836

Screen Definition Facility II (SDF II)

Book title	Order number
VSE Administrator's Guide	SH12-6311
VSE General Introduction	SH12-6315
VSE Primer for CICS/BMS Programs	SH12-6313
VSE Run-Time Services	SH12-6312

Notices

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

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

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

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

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

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

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

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

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

Trademarks and service marks

Index

Sending your comments to IBM

CICS® Transaction Server for VSE/ESA™

System Definition Guide

SC33-1651-05

If you want to send to IBM any comments you have about this book, please use one of the methods listed below. Feel free to comment on anything you regard as a specific error or omission in the subject matter, and on the clarity, organization or completeness of the book itself.

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:

  IBM UK Laboratories
  Information Development
  Mail Point 095
  Hursley Park
  Winchester, SO21 2JN
  England

- 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)
  - Email: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

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

Program Number: 5648-054

SC33-1651-05