CICS Transaction Server for z/OS



Configuring CICS

Version 5 Release 4

CICS Transaction Server for z/OS



Configuring CICS

Version 5 Release 4

Note
Before using this information and the product it supports, read the information in "Notices" on page 343.

This edition applies to the IBM CICS Transaction Server for z/OS Version 5 Release 4 (product number 5655-Y04) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 1989, 2017.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this PDF vii

Chapter 1. Setting up shared data sets, CSD and SYSIN 1

Planning your CSD configuration 1
Calculating CSD disk space 2
Initializing the CSD 3
Creating a larger CSD 5
Defining CSD attributes 6
Shared user access from the same CICS region . . . 7
Sharing user access from several CICS regions . . 7
Multiple users of the CSD within a CICS region (non-RLS) 8
Sharing a CSD by CICS regions within a single MVS image (non-RLS) 9
Sharing a CSD in a multi-MVS environment (non-RLS) 10
Multiple users of one CSD across CICS or batch regions (non-RLS) 10
Sharing the CSD between different releases of CICS 11
CICS regions that use DB2 11
CICS-supplied compatibility groups 12
Other factors restricting CSD access 12
Differences in CSD management between RLS and non-RLS access 12
Specifying read integrity for the CSD 13
Specifying file control attributes for the CSD . . . 13
Effect of RLS on the CSD batch utility DFHCSDUP 14
Planning for backup and recovery 14
Transaction backout during emergency restart . . 17
Dynamic backout for transactions 17
Other recovery considerations 18
CEDA command syncpoint criteria 18
Accessing the CSD by the offline utility program, DFHCSDUP 18
Logging RDO commands 18
Logging SPI commands 20
Making the CSD available to CICS 21
Installing the RDO transactions 22
Installing definitions for the Japanese language feature 22

Chapter 2. Setting up a CICS region . . 23

Defining data sets 23
Setting up CICS data sets 23
Setting up temporary storage data sets 31
Setting up data sets for transient data 34
Setting up CICS log streams 38
Setting up the catalog data sets 50
Setting up auxiliary trace data sets 60
Defining dump data sets 62
Defining user files 67
Defining the CDBM GROUP command data set 79
Defining the CMAC messages data set 81
Defining the WS-AT data set 83
Setting up the debugging profiles data sets . . . 84
Specifying CICS system initialization parameters . . 89
System initialization parameters to set up a CICS region 90
Specifying DFHSIT macro parameters 97
The default system initialization table 99
Assembling the SIT 106
Selecting versions of CICS programs and tables 107
Processing system initialization parameters . . . 108
Supplying system initialization parameters to CICS 109
Using system initialization control keywords 109
Controlling start and restart 114
Retrieving information about system initialization parameters 122
Specifying feature toggles 123
Setting up a region-level configuration file . . 125
Setting up group-level configuration files . . . 125
CICS startup 126
Setting address space storage limits for a CICS region 128
Using the sample startup job stream 129
A sample CICS startup procedure 139
Preparing CICS for using debugging tools 140
Preparing your CICS region for debugging . . 140
Starting CICS regions 141
Specifying system initialization parameters before startup 142
Starting CICS as a batch job 143
Starting CICS as a started task 144
Overriding system initialization parameters during startup 145
System console messages for CICS startup . . . 146
CICS Transaction Server resource usage collection for software pricing 149

Chapter 3. Setting up CICS data sharing servers 151

Defining and starting AXM system services . . . 151
Setting up and running a temporary storage server 152
Overview of the temporary storage data sharing server 152
Defining temporary storage pools for temporary storage data sharing 153
Defining TS server regions 156
Queue server automatic ALTER processing . . 163
Shared TS queue server commands 164
Unloading and reloading queue pools 166
Setting up and running a coupling facility data table server 168
Overview of a coupling facility data table server 168
Coupling facility data tables 169
Defining and starting a coupling facility data table server region 175

© Copyright IBM Corp. 1989, 2017

Controlling coupling facility data table server regions 186
Deleting or emptying coupling facility data table pools 192
Unloading and reloading coupling facility data table pools 192
Setting up and running a region status server . . 195
Defining a list structure for a region status server 196
Defining and starting a region status server region 198
Controlling region status servers 199
Deleting region status server pools 205
Setting up and running a named counter server 206
Named counter server overview 206
Defining a named counter options table . . . 208
Defining a list structure for a named counter server 211
Defining and starting a named counter server region 212
Controlling named counter server regions . . . 217
Deleting or emptying named counter pools . . 220
Changing the size of named counter pools . . 221
Unloading and reloading named counter pools 221
Dumping named counter pool list structures 223
Coupling facility server operations 223
Monitoring coupling facility server messages 223
Coupling facility storage management 224
Managing the pool structure 226
Server connection management 227
CICS server support for system-managed processes 230
System-managed list structure rebuild 230
System-managed list structure duplexing . . . 232

Chapter 4. Defining resources 235

Resource definition online (RDO) transaction CEDA 235
The resource definition batch utility DFHCSDUP 235
Sharing the CSD between CICS Transaction Server for z/OS, Version 5 Release 4 and earlier releases 236
Invoking DFHCSDUP as a batch program . . . 236
Invoking the DFHCSDUP program from a user program 239
Entry parameters for the DFHCSDUP program 239
Responsibilities of the user program 240
Command processing in DFHCSDUP following internal error detection 241
Autoinstall 241
Autoinstall models 241
Autoinstall control program 242
Autoinstalling z/OS Communications Server terminals 242
Deciding which terminals to autoinstall . . . 242
Autoinstall and z/OS Communications Server 244
Implementing z/OS Communications Server autoinstall 247
Recovery and restart of autoinstalled terminal definitions 250
Autoinstalling MVS consoles 253
Implementing autoinstall for MVS consoles . . 255
The autoinstall control program for MVS consoles 256
Autoinstalling APPC connections 257
Implementing APPC connection autoinstall . . 258
Model definitions for connection autoinstall . . 259
The autoinstall control program for connections 259
Recovery and restart for connection autoinstall 260
Autoinstalling IPIC connections 260
Autoinstalling programs, map sets, and partition sets 261
Implementing program autoinstall 261
Cataloging for program autoinstall 262
Model definitions for program autoinstall . . . 263
The autoinstall control program for programs 263
Program autoinstall and recovery and restart 264
Autoinstalling model terminal definitions 265
Autoinstalling journals 265
Macro resource definition 265
Introduction to CICS control tables and macros 266
Defining resources in CICS control tables . . . 270
Defining CICS bundles 275
Artifacts that can be deployed in bundles . . . 277
Characteristics of resources in CICS bundles . . 281
Referencing zFS artifacts in a bundle 288
Private resources for application versions . . . 289
Manifest contents for a CICS bundle 295
Scoping of bundles 299
OSGi bundle recovery on a CICS restart . . . 300
Security for bundles 301
Variables in a CICS project 303
Variables and properties files definition. . . . 303
Defining terminal resources 304
Defining z/OS Communications Server terminals 305
Defining sequential (BSAM) devices 307
Defining console devices 309
Defining z/OS Communications Server persistent sessions support 313
Resource definition installation 314
What happens when CICS is initialized . . . 314
What happens when you use the INSTALL command 315
How to install a limited number of data definitions 316
Duplicate resource definition names 316

Chapter 5. Setting up a platform . . . 319

Platform design 319
Sharing CICS regions between region types in platforms 323
Preparing zFS for platforms 324
Creating a platform 325
Deploying a platform. 327

Chapter 6. Setting up access for CICS Explorer 331

Setting up CMCI with CICSPlex SM. 331
Setting up CMCI in a stand-alone CICS region . . 332
Setting up CMCI in a CICS region with active security 334

Estimating storage requirements for CMCI . . . 335
Record count warnings in CMCI 336

Chapter 7. Setting up event processing 341

Notices 343

Index 349

About this PDF

This PDF describes how you set up CICS TS for z/OS. Other PDFs, listed below, describe the configuration for certain areas of CICS and you might need to refer to those as well as this PDF. (In IBM Knowledge Center, all this information is under one section called "Configuring".) You are also likely to need the reference companions to this PDF: the *System Initialization Parameter Reference* and the *Resource Reference*. Before CICS TS V5.4, the information in this PDF was in the *System Definition Guide* and the *Resource Definition Guide*.

Configuring information for areas of CICS is in the following PDFs:

- SOAP and JSON is in *Using Web Services with CICS* .
- ONC/RPC interface is in the *External Interfaces Guide* .
- EXCI is in *Using EXCI with CICS*.
- Java and Liberty are in *Java Applications in CICS*.
- Front End Programming Interface is in the *Front End Programming Interface User's Guide*.
- DB2 is in *Using DB2 with CICS*.
- DBCTL is in the *IMS DB Control Guide*.
- Shared data tables are in the *Shared Data Tables Guide*.
- CICSPlex SM is in *CICSPlex SM Administration*.
- BTS is in *Business Transaction Services*
- Connections between CICS systems is in the *Intercommunication Guide*

Reference information about the parameters used in CICS system initialization is in the *System Initialization Parameter Reference*.

For details of the terms and notation used in this book, see Conventions and terminology used in the CICS documentation in IBM Knowledge Center.

Date of this PDF

This PDF was created on June 6th 2017.

© Copyright IBM Corp. 1989, 2017

Chapter 1. Setting up shared data sets, CSD and SYSIN

The system definition data set (CSD) and the SYSIN data set are typically shared between more than one CICS® region. The CSD stores resources definitions and the SYSIN data set contains start up information for a region, such as system initialization parameter values. Use the supplied jobs to create these data sets.

About this task

A CSD is required for most resource definitions. If you are creating a CSD for the first time, go through the steps listed in "Planning your CSD configuration." The remainder of this section describes these steps in more detail.

If you are already using a CSD with a previous release of CICS, upgrade your CSD to include CICS resource definitions that are new in CICS Transaction Server for z/OS®, Version 5 Release 4 . For information about upgrading your CSD, see System definition file utility program (DFHCSDUP).

The SYSIN data set contains information that configures CICS during start up. You can create a data set to share a standard set of start up parameters across your CICS regions, and override these values as required on the start up job stream for particular CICS regions. CICS supplies the DFHCOMDS job to create a SYSIN data set. For more information, see Creating the CICS data sets in Installing.

Planning your CSD configuration

Before you can create a CSD, you must plan your configuration.

Procedure

1. Decide how much disk space you require.
2. Decide whether you want to use the CSD in RLS or non-RLS mode. Having the CSD open in RLS mode allows more than one CICS region to update the CSD concurrently. However, if your CSD is defined as a recoverable data set, and you want update it using the batch utility, DFHCSDUP, you must quiesce the CSD in the CICS regions before running DFHCSDUP.

   If you decide to use RLS for the CSD, specify `CSDRLS=YES` as a system initialization parameter. See "VSAM record-level sharing (RLS)" on page 70.
3. Decide what backup and recovery procedures you require for your CSD. The CSD can use backup-while-open (BWO), which means that DFSMS components can back up the CSD while the data set is open for update. To use BWO, ensure that DFSMShsm and DFSMSdss components of DFSMS 1.2 or later are available. The CSD must have an ICF catalog entry and be defined in SMS-managed storage.
   - Define the recovery options for a CSD accessed in RLS mode:
     - Specify BWO(TYPECICS) in the ICF catalog to make the data set eligible for backup-while open.
     - Make the CSD a recoverable data set by specifying the appropriate LOG parameter in the ICF catalog.
   - Define the recovery options for a CSD accessed in non-RLS mode:

© Copyright IBM Corp. 1989, 2017

- Make the data set eligible for backup-while-open by either specifying BWO(TYPECICS) in the catalog or setting the **CSDBKUP** system initialization parameter to DYNAMIC.
- Make the data set recoverable by setting the appropriate LOG parameter in the ICF catalog entry or specifying the appropriate option on the **CSDRECOV** system initialization parameter.

  By default, the recovery options in the catalog override the attributes in the FILE resource. If no recovery options are set in the catalog, CICS uses the attribute values of the FILE resource. You can set the **NONRLSRECOV** system initialization parameter to FILEDEF, if you want CICS to always use the recovery options on the FILE resource instead of the catalog.

4. Define and initialize the CSD.
5. Decide what CICS file processing attributes you want for your CSD. Although the CSD is a CICS file-control-managed data set, you define file control resource definitions for the CSD by specifying CSDxxxxx system initialization parameters (see "Defining CSD attributes" on page 6).
6. Decide if you want to use command logs for RDO. See "Logging RDO commands" on page 18 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, either by using dynamic allocation or by including the necessary DD statement in the CICS startup job stream. For dynamic allocation of the CSD, you name the fully qualified data set name, and the disposition, on the CSDDSN and the CSDDISP system initialization parameters respectively.

   When you have started CICS, test the RDO transactions CEDA, CEDB, and CEDC.

Calculating CSD disk space

Before you can create the CSD, you must calculate the amount of space you need in your CSD for definition records.

Procedure

1. In your calculation, allow for approximately 1800 CICS-supplied resource definitions of various types, which are loaded into the CSD when you initialize the CSD with the utility program, DFHCSDUP. You need to consider:
   a. Each resource definition (for example each program, transaction and terminal) needs one record. The sizes of these definition records are:

Resource	Definition record size (maximum)
ATOMSERVICE	720 bytes
BUNDLE	698 bytes
CONNECTION	260 bytes
CORBASERVER	1375 bytes
DB2CONN	308 bytes
DB2ENTRY	236 bytes
DB2TRAN	198 bytes
DJAR	445 bytes
DOCTEMPLATE	567 bytes
ENQMODEL	447 bytes
FILE	369 bytes
IPCONN	468 bytes
JOURNALMODEL	222 bytes
JVMSERVER	208 bytes

Resource	Definition record size (maximum)
LIBRARY	925 bytes
LSRPOOL	425 bytes
MAPSET	190 bytes
MQCONN	240 bytes
PARTITIONSET	190 bytes
PARTNER	408 bytes
PIPELINE	959 bytes
PROCESSTYPE	206 bytes
PROFILE	231 bytes
PROGRAM	499 bytes
REQUESTMODEL	1211 bytes
SESSION	296 bytes
TCPIPSERVICE	571 bytes
TDQUEUE	331 bytes
TERMINAL	327 bytes
TRANCLASS	192 bytes
TRANSACTION	545 bytes
TSMODEL	308 bytes
TYPETERM	402 bytes
URIMAP	1443 bytes
WEBSERVICE	708 bytes

   b. Each group requires two 122-byte records
   c. Each group list requires two 122-byte records
   d. Each group name within a list requires one 68-byte record
2. Add a suitable contingency (approximately 25%) to the size that you have calculated so far.

What to do next

Use your final figure when you define the VSAM cluster for the CSD. (See the sample job in "Initializing the CSD.")

Initializing the CSD

The **INITIALIZE** command initializes your CSD with definitions of the resources that are supplied by CICS. After initialization, you can migrate resource definitions from your CICS control tables and begin defining your resources. You use the **INITIALIZE** command only once in the lifetime of the CSD.

About this task

Before you can use the CSD, you must define it as a VSAM KSDS data set, and initialize it using the DFHCSDUP utility program. Use the sample job to define and initialize the CSD.

Procedure

1. Code the **KEYS** parameter as shown in the sample job. The key length is 22 bytes.
2. Calculate the CSD disk space that is required:
   - The **RECORDS** parameter defines how many records will be allocated to the CSD. The parameter takes two values: `n1` is the primary extent, and `n2` is the secondary extent, which will only be used if the allocated number of records

defined by n1 is exceeded. For guidance on setting this parameter, see "Calculating CSD disk space" on page 2.

- The **RECORDSIZE** parameter defines the average record size, in bytes, as well as the maximum record size. The average record size is 200 bytes for a CSD that contains only the supplied resource definitions (generated by the **INITIALIZE** and **UPGRADE** commands). If you create a larger proportion of terminal resource definition entries than are defined in the initial CSD, then the average record size will be higher. The maximum record size is 2000, as shown in the sample job.

3. Code the **SHAREOPTIONS** parameter as shown in the sample job.
4. Optional: You can specify the recovery attributes for the CSD in the ICF catalog instead of using the **CSD** system initialization parameters. If you decide to use the CSD in RLS mode, you must define the recovery attributes in the ICF catalog:
   - LOG(NONE) (Nonrecoverable data set)
   - LOG(UNDO) (For backout only)
   - LOG(ALL) (For both backout and forward recovery)

   If you specify LOG(ALL), you must also specify LOGSTREAMID to define the 26-character name of the MVS™ log stream to be used as the forward recovery log. If you specify recovery attributes in the ICF catalog, and also want to use BWO, specify LOG(ALL) and BWO(TYPECICS).
5. You must specify the DDNAME for the CSD as DFHCSD.

Example

```
//DEFINIT  JOB  accounting information
//DEFCSD   EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//AMSDUMP  DD   SYSOUT=A
//SYSIN    DD   *
   DEFINE CLUSTER -
            (NAME(CICSTS54.CICS.applid.DFHCSD) -
            VOLUMES(volid)               -
            KEYS(22 0)                   -                        1
            INDEXED                      -
            RECORDS(n1 n2)               -
            RECORDSIZE(200 2000)         -                        2
            FREESPACE(10 10)             -
            SHAREOPTIONS(2)              -                        3
            LOG(ALL)                     -                        4
            LOGSTREAMID(CICSTS54.CICS.CSD.FWDRECOV)   -           4
            BWO(NO))                     -                        4

          DATA                           -
            (NAME(CICSTS54.CICS.applid.DFHCSD.DATA)   -
            CONTROLINTERVALSIZE(8192))          -
          INDEX                                 -
            (NAME(CICSTS54.CICS.applid.DFHCSD.INDEX))
/*
//INIT     EXEC PGM=DFHCSDUP,REGION=300K
//STEPLIB  DD DSN=CICSTS54.CICS.SDFHLOAD,DISP=SHR
//DFHCSD   DD DSN=CICSTS54.CICS.applid.DFHCSD,DISP=SHR            5
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSIN    DD *
            INITIALIZE
            LIST ALL OBJECTS
/*
//
```

Figure 1. Sample job to define and initialize the CSD

What to do next

The command **LIST ALL OBJECTS** lists the CICS-supplied resources that are now in the CSD.

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 AMS 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 z/OS DFSMS Access Method Services for Catalogs .

Defining CSD attributes

File processing attributes for the CSD are defined in a number of system initialization parameters.

About this task

Define suitable definitions for the following system initialization parameters:

Procedure

1. Define the type of access that is allowed using the **CSDACC** parameter.
2. Define whether the CSD is eligible for BWO using the **CSDBKUP** parameter. This parameter is ignored if you specify CSDRLS=YES. CICS uses the BWO parameter in the ICF catalog instead. By default, CICS also uses the BWO parameter in the ICF catalog for non-RLS mode CSDs if the LOG parameter in the ICF catalog specifies either UNDO or ALL. You can set the **NONRLSRECOV** system initialization parameter to FILEDEF if you want CICS to always use the **CSDBKUP** parameter over the BWO attribute.
3. Define the number of buffers for CSD data using the **CSDBUFND** parameter. This parameter is ignored if you specify CSDRLS=YES.
4. Define the number of buffers for the CSD index using the **CSDBUFNI** parameter. This parameter is ignored if you specify CSDRLS=YES.
5. Define the disposition of the CSD data set using the **CSDDISP** parameter.
6. Define the JCL data set name (DSNAME) of the CSD using the **CSDDSN** parameter.
7. Define a forward recovery journal identifier using the **CSDFRLOG** parameter. This parameter is ignored if you specify CSDRLS=YES, or if the recovery attributes are defined in the ICF catalog on the **LOG** parameter, in which case LOGSTREAMID from the ICF catalog is used instead. You can set the **NONRLSRECOV** system initialization parameter to FILEDEF if you want CICS to always use the **CSDFRLOG** parameter over the LOGSTREAMID attribute.
8. Define the level of read integrity for a CSD accessed in RLS mode using the **CSDINTEG** parameter.
9. Define an identifier for automatic journaling using the **CSDJID** parameter.
10. Define the VSAM local shared resource pool using the **CSDLSRNO** parameter. This value is ignored if you specify CSDRLS=YES.
11. Define whether or not the CSD is recoverable using the **CSDRECOV** parameter. This parameter is ignored if you specify CSDRLS=YES and CICS uses the **LOG** parameter from the ICF catalog instead. If **LOG** is “undefined”, any attempt to open the CSD in RLS mode fails.

    If CSDRLS=NO, this parameter is used only if **LOG** in the ICF catalog is “undefined.” By default, if **LOG** in the ICF catalog specifies NONE, UNDO, or ALL, the **LOG** parameter overrides the **CSDRECOV** value. You can set the **NONRLSRECOV** system initialization parameter to FILEDEF if you want CICS to always use the CSDRECOV parameter over the LOG attribute.
12. Define whether the CSD is accessed in RLS or non-RLS mode using the **CSDRLS** parameter.
13. Define the number of strings for concurrent requests using the **CSDSTRNO** parameter. This value is ignored and a value of 1024 is assumed if you specify CSDRLS=YES.

What to do next

These parameters are described in greater detail in "Specifying CICS system initialization parameters" on page 89.

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 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 (non-RLS)" on page 8.

Sharing user access from several CICS regions

Several users in different CICS regions can access the CSD at the same time.

1. Give one CICS region read and 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 MVS image or different MVS images.
2. 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.
3. 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:
   a. 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.
   b. Define the CSD as read-only to other CICS regions.
   c. For all regions other than the CSD-owning region:
      1) Redefine the CEDB transaction as a remote transaction (to be run in the CSD-owning region).
      2) 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. In this case, you must not specify CSDBKUP=DYNAMIC, because the CSD would not be suitable for BWO.

4. You can define several CICS regions with read/write access to the CSD, but this should only be considered if the CICS regions run in the same MVS image, and all are at the latest CICS level.
5. If you give several CICS regions read/write access to the same CSD, and those regions are in the same MVS image, integrity of the CSD is maintained by the SHAREOPTIONS(2) operand of the VSAM definition, as shown in the Figure 1 on page 5.
6. If you give several CICS regions read/write access to the same CSD, and those regions are in different MVS images, the VSAM SHAREOPTIONS(2) operand does not provide CSD integrity, because the VSAMs for those MVS images do not know about each other.

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 12.

For information about the system initialization parameters for controlling access to the CSD, see "Defining CSD attributes" on page 6.

Multiple users of the CSD within a CICS region (non-RLS)

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.

CEDA issues 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 MVS image (non-RLS)

The CSD may be shared by a number of CICS regions within the same MVS image.

You can maintain the integrity of the CSD in this situation by coding SHAREOPTIONS(2) on the VSAM definition, as shown in the Figure 1 on page 5. 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 in "Sharing user access from several CICS regions" on page 7.

  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. See "Specifying read integrity for the CSD" on page 13 for information about read integrity in a data set accessed in RLS mode.

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, CEDB, or CEDC 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 14 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 parameters allow the following operations on the CSD:

READONLY
: Read and browse.

READWRITE
: Add, delete, update, read and browse.

Sharing a CSD in a multi-MVS environment (non-RLS)

If you need to share a CSD between CICS regions that are running in different MVS images, you should ensure that only one region has read/write access.

The VSAM SHAREOPTIONS(2) offers no integrity, because the VSAMs running in a multi-MVS environment do not know of each other.

Because of these limitations, active and alternate CICS regions running in different MVS images must not share the CSD with other CICS regions, unless you are using some form of global enqueuing (for example, with global resource serialization (GRS)).

These multi-MVS restrictions also apply to running the offline utility, DFHCSDUP.

Multiple users of one CSD across CICS or batch regions (non-RLS)

There are four types of activity that require access to the CSD.

The types of access needed in the four situations where the CSD are used are shown in Table 1.

Table 1. CSD access

	Type of activity	**Access**
1	CICS region performing initialization (cold or initial 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 parameter)
3	Batch region running utility program DFHCSDUP	Read/write or read only, depending on PARM parameter
4	CICS regions performing emergency restart, and CSD file backout is required	Read/write

Note the following limitations when the activities listed in Table 1 are attempted concurrently:

1. You cannot run DFHCSDUP in read/write mode in a batch region if any CICS region using the same CSD is running one of the CEDA, CEDB, or CEDC transactions. (The exception is when the CED*x* transactions accessing the CSD are in a region (or regions) 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 CED*x* transactions runs in a CICS region whose CSD is defined for read-write access if any of the RDO transactions are running in another CICS region that has the CSD defined for read-write access.

A CICS region starting with an initial or cold start opens the CSD for read access only during initialization, regardless of the CSDACC operand. This enables a CICS region to be initialized even if a user on another region or the DFHCSDUP utility program is updating the CSD at the same time. After the group lists are installed, CICS leaves the CSD in a closed state.

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 different releases of CICS

You can share the CSD between different releases of CICS so that common resource definitions are shared. When you update resource definitions, you must update the CSD from the highest release level, and you must consider the effect on obsolete resource attributes.

Resource attributes become obsolete when they are not relevant to a new release of CICS. Such attributes are displayed on CED*x* panels as protected fields, which indicates that they are not supported by this release.

When you update resource definitions by using the ALTER command, any obsolete attributes on definitions are kept, so you can update resource definitions by using the current release without affecting obsolete attributes.

If you are sharing the CSD with CICS regions that are at an earlier release, you can update the obsolete attributes by removing the protection from the protected fields when in ALTER mode. To do this, use the F2 function key, which is the compatibility key (COM) on the CEDA or CEDB display panels. The F2 function key converts protected fields to unprotected fields that you can modify. If you want to use this protection removal facility to share common resource definitions between different release levels of CICS, you must update the CSD from the highest release level.

For information about using the CEDA and CEDB ALTER commands to update resource definitions in compatibility mode, see Resource management transaction CEDA commands.

You can also use the CSD utility program, DFHCSDUP, to update resources that specify obsolete attributes. You must specify the compatibility option, COMPAT or NOCOMPAT, on the **PARM** parameter of the EXEC PGM=DFHCSDUP statement. The default is NOCOMPAT, which means that the program does not update obsolete attributes. For more information, see Entry parameters for the DFHCSDUP program.

CICS regions that use DB2

If you share your CSD between different releases of CICS that use DB2®, you must use the DB2 resource definitions appropriate for each release of CICS.

With those releases of CICS that ship the CICS DB2 attachment facility, you must use the CICS-supplied group called DFHDB2. This group is included in the CICS-supplied startup list, DFHLIST, and specifies different program names from the attachment facility provided by DB2.

For earlier releases of CICS that do not provide the DFHDB2 group, you must use your own resource definitions that specify the resource names appropriate for the release of CICS and DB2.

CICS-supplied compatibility groups

If you share the CSD between CICS Transaction Server for z/OS, Version 5 Release 4 and an earlier release of CICS, 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 TS for z/OS, Version 5.4 level, some IBM® groups that are referenced by your group list are deleted and the contents are transferred to one of the compatibility groups, DFHCOMP*x*. To ensure that these groups continue to be available to your CICS regions at earlier releases, add the compatibility groups *after* all the other CICS-supplied definitions.

For information about upgrading your CSD, and about the compatibility groups in CICS TS for z/OS, Version 5.4, see CSD compatibility between different CICS releases in Upgrading.

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 (non-RLS)” on page 8, 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” on page 14.

Differences in CSD management between RLS and non-RLS access

Although a CSD accessed in RLS mode is protected by VSAM RLS locking, this operates at the CICS file control level. It does not change the way the CEDA and CEDB transactions manage the integrity of CSD groups.

The CED*x* transactions protect resource definitions in the same way for RLS mode and non-RLS mode CSDs. They protect individual resource definitions against concurrent updates by a series of internal locks on the CSD. The RDO transactions apply these locks at the group level. While RDO transactions are executing a command that updates any element in a group, they use the internal lock to prevent other RDO transactions within a CICS region from updating the same group. The locks are freed only when the updating command completes execution. Operations on lists are protected in the same way. However, in an RLS environment, these internal locks affect all CICS regions that open the CSD in RLS mode. In the non-RLS case they apply only to the CICS region that has the data set open for update (which can only be a single region).

The use of a single buffer pool by the SMSVSAM server removes some of the problems of sharing data that you get with a non-RLS CSD.

Some other points to note are:

- If a CSD is defined with CSDACC=READWRITE and CSDRLS=YES, more than one CICS region can open the CSD concurrently. However, file control closes the CSD opened in RLS mode at termination of each CED*x* transaction, in the same way as for a non-RLS CSD. CSDACC=READONLY is not necessary for a CSD accessed in RLS mode.
- The number of concurrent requests that can be processed against the CSD, is always 1024 for RLS. Diagnostic messages about CSDSTRNO value do not occur for RLS-mode CSDs.
- The VSAM cluster definition SHAREOPTIONS parameter is ignored by SMSVSAM when an application, such as CICS, opens a data set in RLS mode.
- A CSD accessed in RLS mode could give rise to RDO transaction failures that do not occur for a non-RLS mode CSD: For example:
  - An RDO transaction could be holding an RLS exclusive lock while it updates a record, which causes another RDO transaction to time out.
  - If the CSD is recoverable and CICS or MVS fails, update locks of failed-inflight RDO transactions are converted into retained locks. This could result an RDO transaction receiving a LOCKED response from VSAM which, in turn, would cause the RDO transaction to fail.

Specifying read integrity for the CSD

You can specify that you want read integrity for a CSD opened in RLS mode.

This ensures that CED*x* INSTALL command always installs the latest version of a resource definition. The CEDA INSTALL command has to wait for a lock on any CSD record that it is trying to install if another CED*x* transaction is updating the record. The installation completes only when the updating task has finished updating the record and released its exclusive lock.

Although the CSDINTEG system initialization parameter supports both consistent and repeatable read integrity, consistent read integrity should provide all the benefit you need for your RDO operations.

Specifying file control attributes for the CSD

You specify file control attributes for the CSD using the CSD*xxxxx* system initialization parameters, with a few exceptions.

These exceptions are as follows:

CSDBKUP
You specify backup-while-open support for the CSD using the VSAM **BWO** parameter in the ICF catalog.

CSDBUFND
Ignored.

CSDBUFNI
Ignored.

CSDFRLOG
You specify the forward recovery log stream for the CSD using the VSAM **LOGSTREAMID** parameter in the ICF catalog.

CSDINTEG
You specify read integrity for RDO transactions (CED*x*) using this system initialization parameter.

CSDLSRNO
Ignored.

CSDRECOV
You specify the recovery attributes for the CSD using the VSAM **LOG** parameter in the ICF catalog. If **LOG** is "undefined", any attempt to open the CSD in RLS mode will fail.

CSDSTRNO
For RLS, the number of strings defaults to 1024.

Effect of RLS on the CSD batch utility DFHCSDUP

You can use DFHCSDUP to update a **non-recoverable** CSD in RLS mode while CICS also has the CSD open for update in RLS mode.

To enable DFHCSDUP to update the CSD in RLS mode, specify RLS=NRI or RLS=CR in the DD statement for the CSD in the DFHCSDUP JCL. Generally, DFHCSDUP does not perform as well in RLS mode as in non-RLS mode.

You cannot run DFHCSDUP while CICS regions have the CSD open in RLS mode if the CSD is defined as recoverable. This is because a non-CICS job, such as DFHCSDUP, is not allowed to open a recoverable data set for output in non-RLS mode while it is already open in RLS mode. Therefore, before you can run DFHCSDUP, you must quiesce the CSD by issuing a CEMT, or an EXEC CICS, SET DSNAME(...) QUIESCED command.

A recoverable CSD is unavailable to all CICS regions while DFHCSDUP is running until it is unquiesced, which makes it available again in RLS mode. To unquiesce the CSD at the end of the DFHCSDUP run, issue a CEMT, or an EXEC CICS, DSNAME(...) UNQUIESCED command.

For a recoverable CSD, the main factor to consider when planning whether to use RLS is how much you use DFHCSDUP compared with the CED*x* transactions. If you use DFHCSDUP frequently to update your production CSD, you may decide that it is better to use the CSD in non-RLS mode. On the other hand, if you use DFHCSDUP only occasionally, and you want the ability to update the CSD online from any CICS region, use RLS.

Planning for backup and recovery

To guard against system failures that affect your CSD, take a backup 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.

About this task

To keep the backup of the CSD as up-to-date as possible, make an image copy of your CSD before each period of update activity, either by an RDO transaction or DFHCSDUP.

Alternatively, because the CSD is open for update whenever RDO work is taking place, it is a good candidate for eligibility for BWO. If the CSD is specified as eligible for BWO, and the data set is corrupted, you can restore a BWO image of the CSD using DFSMSdss, then run forward recovery to the point of corruption using a forward recovery utility.

For a CSD opened in RLS mode, the recovery attributes must be defined in the ICF catalog entry for the CSD, and CICS uses the forward recovery log's log stream name (LSN) from the ICF catalog.

For a CSD opened in non-RLS mode, the recovery attributes can be defined in the ICF catalog entry for the CSD, or on the CSD system initialization parameters. The forward recovery log stream name (LSN) is retrieved from either CSDFRLOG or the ICF catalog. If LOG is defined in the catalog, the forward recovery log stream specified in the catalog is used. If LOG is not defined, the CSDFRLOG journal ID is used to determine the log stream name. If the **NONRLSRECOV** system initialization parameter is set to FILEDEF, the CSDFRLOG journal ID is always used to determine the log stream name. Any recovery attributes specified on the ICF catalog are ignored.

For a CSD opened in non-RLS mode, you can use the system initialization parameter CSDBKUP=DYNAMIC|STATIC to indicate whether the CSD is eligible for BWO. Specify CSDBKUP=DYNAMIC for BWO support, or STATIC (the default) for a "normal" quiesced backup. If you specify BWO support for the CSD you must also define it as forward recoverable. For more information about BWO, see "Defining backup while open (BWO) for VSAM files" on page 28.

For a CSD opened in RLS mode, you must specify all recovery attributes, which includes backup, in the ICF catalog. BWO backup eligibility is specified using BWO(TYPECICS).

If you specify forward recovery for the CSD, changes (after images) made by CICS to the CSD are logged in the forward recovery log stream. Using the latest backup, and the after images from forward recovery log stream, you can recover all the changes made by running a recovery program, such as the CICS VSAM forward recovery utility. After performing forward recovery, you must reenter any 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 "Logging RDO commands" on page 18 for more information.

Recoverability, forward recovery log stream names, and BWO eligibility can be defined optionally in the ICF catalog for a non-RLS accessed CSD, but must be defined the ICF catalog if the CSD is accessed in RLS mode.

The CSDBKUP, CSDRECOV, and CSDFRLOG system initialization parameters interact according to how they are specified. Table 2 on page 16 and Table 3 on page 16 summarize their effects when the SIT is assembled and during CICS override processing, respectively.

Table 2. CSDBKUP and related parameters at SIT assembly time for CSDRLS=NO

CSDRECOV	CSDFRLOG	CSDBKUP	Result
ALL	FRLOG from 01 through 99.	Either DYNAMIC or STATIC	OK
ALL	NO	Either DYNAMIC or STATIC	SIT assembly fails with MNOTE stating that CSDRECOV=ALL implies that the CSDFRLOG option must be specified.
BACKOUTONLY or NONE	FRLOG from 01 through 99.	DYNAMIC	SIT assembly fails with assembler MNOTES stating that CSDBKUP=DYNAMIC requires CSDRECOV=ALL and that CSDFRLOG requires CSDRECOV=ALL.
BACKOUTONLY or NONE	NO	DYNAMIC	SIT assembly fails with an assembler MNOTE stating that CSDBKUP=DYNAMIC requires CSDRECOV=ALL.
BACKOUTONLY or NONE	NO	STATIC	OK
BACKOUTONLY or NONE	FRLOG from 01 through 99.	STATIC	SIT assembly warning MNOTE stating that CSDFRLOG is ignored unless CSDRECOV=ALL.

Note:

1. When CSDBKUP=DYNAMIC, the CSD is eligible for BWO.
2. Backup and recovery attributes must be specified in the ICF catalog for a CSD opened in RLS mode (CSDRLS=YES).
3. Backup and recovery attributes can optionally be specified in the ICF catalog for a CSD opened in non-RLS mode (CSDRLS=NO), but you must still have a consistent set of parameters as defined in the table above.

Table 3. CSDBKUP and related system initialization parameters during CICS override processing (CSDRLS=NO)

CSDRECOV	CSDFRLOG	CSDBKUP (see Notes)	Result
ALL	FRLOG from 01 through 99.	Either DYNAMIC or STATIC.	OK
ALL	NO	Either DYNAMIC or STATIC.	Message DFHPA1944 is issued stating that CSDRECOV=ALL cannot be specified without a CSDFRLOG if CSDRLS=NO. CICS initialization is terminated.
BACKOUTONLY or NONE	FRLOG from 01 through 99.	DYNAMIC	Processing continues and messages DFHPA1929, stating that CSDBKUP has defaulted to STATIC, and DFHPA1930, stating that CSDFRLOG has been ignored, are issued.

Table 3. CSDBKUP and related system initialization parameters during CICS override processing (CSDRLS=NO) (continued)

CSDRECOV	CSDFRLOG	CSDBKUP (see Notes)	Result
BACKOUTONLY or NONE	NO	DYNAMIC	Processing continues and message DFHPA1929 is issued, stating that CSDBKUP has defaulted to STATIC.
BACKOUTONLY or NONE	NO	STATIC	OK
BACKOUTONLY or NONE	FRLOG from 01 through 99.	STATIC	Processing continues and message DFHPA1930 stating that CSDFRLOG has been ignored is issued.

Note:

1. When CSDBKUP=DYNAMIC, the CSD is eligible for BWO.
2. Backup and recovery attributes must be specified in the ICF catalog for a CSD opened in RLS mode (CSDRLS=YES).
3. Backup and recovery attributes can optionally be specified in the ICF catalog for a CSD opened in non-RLS mode (CSDRLS=NO), but you must still have a consistent set of parameters as defined in the table above.

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. To enable recovery of the updates made outside CICS, you need to use an image copy. If you update the CSD from outside CICS, do not use CEDA to update the CSD until an image copy has been made.

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. (Defining the CSD as non-recoverable has the effect of preventing backout, but this is not recommended.)

Other recovery considerations

When you are deciding what recoverability options to specify, you must consider a number of factors.

These factors are as follows:
- 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." For information about sharing the CSD between CICS regions, see Sharing user access from several CICS regions. For information about using the DFHCSDUP utility to access the CSD, see "Accessing the CSD by the offline utility program, DFHCSDUP."

CEDA command syncpoint criteria

You can issue a CEDA command on the command line, or issue the command as a series of single commands from 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 installation 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 installation fails:
- AUTOINSTALL MODEL,FILE, LSRPOOL, MAPSET, PARTITIONSET, PARTNER, PROFILE, PROGRAM, TDQUEUE, and TRANSACTION,

Changes made to the following resource definitions by an INSTALL command are committed at the group level and are backed out if the installation 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.

Logging RDO commands

If you want to record RDO commands, use the sample job to create definitions for the extrapartition queues CADL, CAIL, CRDI, CSDL, CSFL, CSKL, CSLB, CSPL, and CSRL.

About this task

These queues are used as follows:

CADL Logs z/OS Communications Server resources installed in the active CICS region. CICS records in this log all z/OS Communications Server resources installed, z/OS Communications Server resources deleted, and dynamically installed z/OS Communications Server resources 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 the dynamic installation and deletion of autoinstall terminal models.

CRDI Logs installed resource definitions, such as 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 resource definitions that are installed or deleted, dynamically installed file resource definitions 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 resource definitions that are installed or deleted, and dynamically installed transaction and profile resource definitions that are discarded.

CSLB Logs changes to the dynamic LIBRARY resources installed in the active CICS region. That is, install and discard changes, and any changes to the enablement, ranking or critical status of the LIBRARY. If no dynamic LIBRARY resources are installed, no audit logging will take place.

CSPL Logs program resources installed in the active CICS region. That is, all program resource definitions that are installed or deleted, and dynamically installed program resource definitions 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, you can use the definitions shown in Figure 2 on page 20. If you like, you can direct these logs to any other transient data queue, or define them as extrapartition data sets.

Note: VTAM® is now z/OS Communications Server.

```
*
DEFINE TDQUEUE (CSSL)         GROUP(DFHDCTG)
       DESCRIPTION(USED FOR MESSAGES)
       TYPE(EXTRA)            TYPEFILE(OUTPUT)
       RECORDSIZE(132)        BLOCKSIZE(136)
       RECORDFORMAT(VARIABLE) BLOCKFORMAT(UNBLOCKED)
                              DDNAME(MSGUSR)
*
DEFINE TDQUEUE (CADL)         GROUP(DFHDCTG)
       DESCRIPTION(CEDA VTAM RESOURCE LOGGING)
       TYPE(INDIRECT)         INDIRECTNAME(CSSL)
*
DEFINE TDQUEUE (CAIL)         GROUP(DFHDCTG)
       DESCRIPTION(AITM MESSAGES)
       TYPE(INDIRECT)         INDIRECTNAME(CSSL)
*
DEFINE TDQUEUE (CRDI)         GROUP(DFHDCTG)
       DESCRIPTION(RDO INSTALL LOG)
       TYPE(INDIRECT)         INDIRECTNAME(CSSL)
*
DEFINE TDQUEUE (CSLB)         GROUP(DFHDCTG)
       DESCRIPTION(CICS LD Domain LIBRARY Audit Trail)
       TYPE(INDIRECT)         INDIRECTNAME(CSSL)
*
DEFINE TDQUEUE (CSDL)         GROUP(DFHDCTG)
       DESCRIPTION(CEDA COMMAND LOGGING)
       TYPE(INDIRECT)         INDIRECTNAME(CSSL)
*
DEFINE TDQUEUE (CSFL)         GROUP(DFHDCTG)
       DESCRIPTION(FILE ALLOCATION MESSAGES)
       TYPE(INDIRECT)         INDIRECTNAME(CSSL)
*
DEFINE TDQUEUE (CSKL)         GROUP(DFHDCTG)
       DESCRIPTION(TRANSACTION MANAGER MESSAGES)
       TYPE(INDIRECT)         INDIRECTNAME(CSSL)
*
DEFINE TDQUEUE (CSPL)         GROUP(DFHDCTG)
       DESCRIPTION(PROGRAM MANAGER MESSAGES)
       TYPE(INDIRECT)         INDIRECTNAME(CSSL)
*
DEFINE TDQUEUE (CSRL)         GROUP(DFHDCTG)
       DESCRIPTION(PARTNER RESOURCE MANAGER)
       TYPE(INDIRECT)         INDIRECTNAME(CSSL)
```

Figure 2. Definitions for RDO command logs sent to CSSL

Logging SPI commands

System programming interface commands are audited to aid system administration and auditing.

About this task

Auditing system programming interface commands generates messages which are directed to the CADS transient data queue. The messages are written in a human readable form and can be redirected to another queue to aid problem determination. You can also redirect them to a separate queue if the volume of messages is high. See SPI commands that can be audited .

The CADS queue is defined in the DFHDCTG group which is part of DFHLIST. CADS is defined as an indirect queue and redirected to the CSSL queue by default. The definitions are shown in Figure 3. However, you can redirect CADS to an extrapartition queue specially defined for the purpose. For example, you can define an extrapartition queue named "SPIAUDIT" as shown in Figure 4

```
*
DEFINE TDQUEUE (CSSL)         GROUP(DFHDCTG)
       DESCRIPTION(USED FOR MESSAGES)
       TYPE(EXTRA)            TYPEFILE(OUTPUT)
       RECORDSIZE(132)        BLOCKSIZE(136)
       RECORDFORMAT(VARIABLE) BLOCKFORMAT(UNBLOCKED)
                              DDNAME(MSGUSR)
*
DEFINE TDQUEUE (CADS)         GROUP(DFHDCTG)
       DESCRIPTION(SPI AUDIT MESSAGES)
       TYPE(INDIRECT)         INDIRECTNAME(CSSL)
*
```

Figure 3. Definitions for SPI audit messages redirected to CSSL

```
*
DEFINE TDQUEUE (SPIA)     GROUP(DFHDCTG)
       DESCRIPTION(USED FOR MESSAGES)
       TYPE(EXTRA)            TYPEFILE(OUTPUT)
       RECORDSIZE(132)        BLOCKSIZE(136)
       RECORDFORMAT(VARIABLE) BLOCKFORMAT(UNBLOCKED)
                              DDNAME(SPIAUDIT)
*
DEFINE TDQUEUE (CADS)         GROUP(DFHDCTG)
       DESCRIPTION(SPI AUDIT MESSAGES)
       TYPE(INDIRECT)         INDIRECTNAME(SPIA)
*
```

Figure 4. Definitions for SPI audit messages redirected to extrapartition queue

Making the CSD available to CICS

To make your CSD available to CICS, you can either include a DD statement in the CICS startup job or use dynamic allocation.

About this task

Procedure

1. Include the following DD statement in your CICS startup job stream:
   ```
   //DFHCSD  DD  DSN=CICSTS54.CICS.applid.DFHCSD,DISP=SHR
   ```
   You usually need the CSD DD statement to include DISP=SHR. (See Sharing user access from several CICS regions.) If you include a DD statement for the CSD in the CICS startup job, the CSD is allocated at the time of CICS job step initiation and remains allocated for the duration of the CICS job step.
2. To dynamically allocate the CSD, specify the data set name (DSNAME) and disposition (DISP) of the CSD, using one of the following methods:
   - The **CSDDSN** and **CSDDISP** system initialization parameters
   - The **CEMT SET FILE** command

- The **EXEC CICS SET FILE** command

Do not provide a DD statement for the CSD in the startup job stream. If there is a CSD DD statement, it is used instead of dynamic allocation. CICS uses the full data set name (DSNAME) to allocate the CSD as part of OPEN processing. The CSD is automatically deallocated when the last entry associated with it is closed.

What to do next

For more information about OPEN processing, see "Defining user files" on page 67. For information about the parameters that you can code for the CSD in the SIT, see "Specifying CICS system initialization parameters" on page 89.

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 the group list that you use for your CICS startup. You specify the group list on the GRPLIST system initialization parameter.

For information about the CEDA, CEDB, and CEDC transactions, see CEDA - resource definition online.

Installing definitions for the Japanese language feature

If you have the Japanese language feature, install the definitions for the feature in the CSD, by running the DFHCSDUP utility.

When you run the DFHCSDUP utility, specify the following option:

```
UPGRADE USING(DFHRDJPN)
```

If you previously used a language-specific mapset for sign on (DFHSNLK), this is no longer supported as part of the DFHRDJPN group. Instead, you can modify the CESL and CESN maps that are supplied in the sample library SDFHSAMP. For details, see and .

If you already have a customized mapset DFHSNLK that you have used in previous releases, you can rename it to DFHSNLE to avoid having to recode and regenerate the module.

For information about the DFHCSDUP utility program and the available commands, see System definition file utility program (DFHCSDUP).

Chapter 2. Setting up a CICS region

To set up a CICS region, create the data sets for the CICS region, edit the JCL to configure the services that the CICS region can use, and start the CICS region.

Defining data sets

CICS system data sets are used by a number of CICS processes including temporary storage, transient data, transaction dump and trace. Some data sets are optional, others are required for running CICS. If a data set needs pre-formatting, a job that you can use for this purpose is shown.

For more information about CICS and data sets see:

- "Setting up CICS data sets"
- "Setting up temporary storage data sets" on page 31
- "Setting up data sets for transient data" on page 34
- "Setting up CICS log streams" on page 38
- "Setting up the catalog data sets" on page 50
- "Setting up auxiliary trace data sets" on page 60
- "Defining dump data sets" on page 62
- "Defining user files" on page 67
- "Defining the CDBM GROUP command data set" on page 79
- "Defining the CMAC messages data set" on page 81
- "Setting up the debugging profiles data sets" on page 84
- "CICS server support for system-managed processes" on page 230

Setting up CICS data sets

You must define a number of data sets to run CICS. Some of these data sets are mandatory, whereas others are needed only if you are using the corresponding facilities. You might also need to provide data set definitions for user files, DL/I databases, and terminals other than z/OS Communications Server terminals.

About this task

Procedure

1. Plan what CICS data sets you require.
   a. Review the CICS facilities that you want in your CICS regions and their data set requirements.
   b. Define a naming convention for your data sets.
   c. Calculate the space to allocate to the data sets and the data definition statements to define them to the running CICS region.
2. Define and catalog the data sets that are used by CICS.
3. If required, initialize or preformat the data sets that are used by CICS.
4. Protect the data sets with an external security manager, such as RACF®, to suit your security requirements.
5. Define the DD statements for the required data sets in the CICS startup job stream. You do not have to define DD statements for the following data sets:

© Copyright IBM Corp. 1989, 2017

- User files for which you are using CICS dynamic allocation facilities
- CICS system data sets that are managed by CICS file control and for which you are using CICS dynamic allocation facilities
- DL/I databases you are accessing through CICS remote DL/I support or DBCTL

For more information about user file definitions, see "Defining user files" on page 67.

What to do next

When you have defined the CICS data sets, you can use CICS utility programs to perform postprocessing on the data sets. These utilities are described in theCICS utility programs.

Reviewing the CICS data sets

Review the list of mandatory and optional data sets to select which ones to create in your CICS regions.

Procedure

1. You must define the following CICS data sets:

Data set	DDNAME	Block or CI size (bytes)	Record format	Data set organization
Auxiliary trace (See Setting up and using auxiliary trace data sets)	DFHAUXT DFHBUXT	4096	F	Sequential
BTS local request queue. (See Local request queue data set)	DFHLRQ	1024 and 2560	VB	VSAM KSDS
Catalogs (See Setting up and using catalog data sets)	DFHGCD DFHLCD	8192 & 2048	VB	VSAM KSDS
CSD (See Setting up the CICS system definition data set)	DFHCSD	8192	VB	VSAM KSDS
Dump (See Defining dump data sets)	DFHDMPA DFHDMPB	32 760 (tape) or 1 track (DASD)	V	Sequential
Messages and codes (See Defining the CMAC messages data set)	DFHCMACD	—	V	VSAM KSDS
Temporary storage (See Setting up the temporary storage data set)	DFHTEMP	See Control interval size of temporary data set	N/A	VSAM ESDS
Transient data extrapartition (See "Defining extrapartition data sets" on page 36)	From the DDNAME option in the resource definition	From the BLOCKSIZE option in the resource definition	From the RECORD FORMAT option in the resource definition	Sequential
Transient data intrapartition (See "Defining the intrapartition data set" on page 34)	DFHINTRA	See "Size of the intrapartition data set" on page 36	N/A	VSAM ESDS

Temporary storage and transient data intrapartition data sets use control interval (CI) processing and therefore the record format is not relevant.

2. You must define the following CICS data sets if you plan to use the equivalent functions:

Data set	DDNAME	More information
CDBM group command	DFHDBFK	Defining the CDBM GROUP command data set
CICSPlex SM WUI server HTML template data set	DFHHTML	Specifying the WUI customizable view and menu help data set
Debugging profiles data sets	DFHDPFMB, DFHDPFMP, DFHDPFMX	Setting up the debugging profiles data sets
Link3270 bridge	DFHBRNSF	Defining DFHBRNSF
Sample file FILEA for sample application programs	FILEA	The FILEA sample application programs
WS-AT directory data set	DFHPIDIR	Defining the WS-AT data set

Defining a data set naming convention

There are no restrictions on the data set names you choose for CICS data sets, other than MVS constraints. In this information, CICSTS54.CICS is used as the high-level qualifier, and the DD name as the lowest level. If you are running multiple CICS regions, you can use the CICS APPLID as a second level qualifier.

About this task

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.

Procedure

- For 4-character names, use the *CTGI* naming convention, as described in *z/OS Parallel Sysplex Application Migration*. It is based on the 4-character *CTGI* symbol, where:
  - C identifies an entire CICSplex
  - T identifies the type of region
  - G identifies a group of regions
  - I identifies iterations of regions within a group
- For 8-character names, as for CICS APPLIDs, use the letters CICS for the first four characters, particularly for production regions. For example, if CICSHTH1 is the APPLID, the data set name for the CSD would be:

```
DFHCSD   DD  DSN=CICSTS54.CICS.CICSHTH1.DFHCSD,DISP=SHR
```

Defining data sets with multiple extents and 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. However, to cater for exceptional cases in which the data set size might have to be much larger than your average, you can define data sets with multiple extents and volumes.

For example, 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 save space until it is needed. When each extra extent becomes full, VSAM creates another. VSAM continues to create extra extents, as needed, up to a maximum of 123 extents. The use of multiple volumes has no effect on this limit.

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.

Use multiple extents over multiple volumes 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.

Creating the CICS data sets

You can use CICS-supplied jobs to create the CICS data sets.

About this task

When you ran the DFHISTAR job as part of the CICS installation, these jobs were tailored to your environment and stored in the library that you specified on the **LIB** parameter of the DFHISTAR job. The default is CICSTS54.XDFHINST).

You can generate several copies of these jobs by rerunning the DFHISTAR job, selecting the jobs that you want to copy. To generate new copies of these jobs, edit the DFHISTAR job to specify new values for the **DSINFO** and **SELECT** parameters. Only those jobs that you name by the **SELECT** parameter are regenerated.

Procedure

1. Run the DFHCOMDS job to create the CICS region definition data set, DFHCSD, and the SYSIN data set.
2. Run the DFHDEFDS job to create data sets used only by one CICS region. You must run a separate copy of this job to create the data sets for each CICS region. This job creates the following data sets:

Data set	Description
DFHAUXT	Non-VSAM auxiliary trace (A) data set
DFHBRNSF	Bridge
DFHBUXT	Non-VSAM auxiliary trace (B) data set
DFHDMPA	Non-VSAM dump (A) data set
DFHDMPB	Non-VSAM dump (B) data set
DFHDPFMB	The debugging profiles base data set
DFHDPFMP	The debugging profiles path data set

Data set	Description
DFHDPFMX	The debugging profiles path data set
DFHGCD	CICS global catalog
DFHHTML	HTML template data set
DFHINTRA	Intrapartition transient data set
DFHLCD	CICS local catalog
DFHLRQ	BTS local request queue
DFHPIDIR	WS-AT directory data set
DFHTEMP	Temporary storage data set
FILEA	Sample program file

3. Run the DFHCMACI job to create the CICS messages data set, DFHCMACD, and load it with the data from the CICS-supplied file, DFHCMACD, in the CICSTS54.CICS.SDFHMSGS target library.

Sizing the MVS system data sets

Besides its own system data sets, CICS also uses some MVS data sets.

About this task

These data sets are:

Data set	Owned or used by	Other comments
SDUMP data sets	MVS SDUMP macro	Used by CICS for system dumps through the MVS SDUMP macro.
SMF data sets	System management facility	Used by CICS monitoring and statistics domains for monitoring and statistics records.
GTF data sets	Generalized trace facility	Used by CICS trace domain for CICS trace entries.

Procedure

1. Recalculate the size of these system data sets, taking into account the increased volumes of data that CICS generates. For example, for an SDUMP data set you must have at least 25 cylinders of a 3380 device, or the equivalent. For guidance information about calculating the size of SDUMP data sets, see the *z/OS MVS Initialization and Tuning Guide*.
2. To prevent the SDUMP data sets from becoming full of unwanted SDUMPs, suppress the SDUMPs as described in "Suppressing system dumps that precede ASRx abends" on page 64. SDUMPs can precede ASRA, ASRB, and ASRD abends after the message DFHAP0001.
3. If you are collecting CICS interval statistics frequently, or the volume of statistics at each interval is high, then you must take this into account when sizing your SMF data sets.

   CICS can write records to SMF of up to 32 756 bytes, resulting in SMF writing spanned records to the SMF data sets. For more efficient use of DASD, you should consider creating the SMF data sets to be used by CICS with a control interval size of either 16 384 bytes (16 KB) or 8192 bytes (8 KB). If you use

other control interval sizes you must consider the trade-off between efficient use of DASD, SMF data set I/O performance, and the possibility of data being lost due to insufficient SMF buffers.

4. If you are collecting CICS monitoring data, you must size the amount of data that is written when the monitoring classes are active. For background information about SMF, and about other SMF data set considerations, see the *z/OS MVS System Management Facilities (SMF)*.
5. If you are running CICS with GTF trace on, make allowance for CICS trace entries in the GTF data sets. For background information about GTF, see *z/OS MVS Diagnosis: Tools and Service Aids*.

Defining backup while open (BWO) for VSAM files

CICS supports the backup while open (BWO) facility provided by DFSMSdss and DFSMShsm. This support enables some types of VSAM data sets to be backed up by DFSMSdss while CICS is currently updating these data sets.

At the same time, CICS logs forward recovery images of any changes to these data sets on a forward recovery journal. At a later date the backup of the data set can be restored using DSMShsm and brought to a point of consistency by applying the forward recovery logs using a forward recovery utility such as CICS VSAM Recovery.

Before you begin

BWO is available only for data sets accessed by CICS file control, which includes the CICS system definition (CSD) data set. You must decide which VSAM user data sets are eligible for BWO, subject to the restrictions detailed in "Restrictions on BWO" on page 29 that are applicable to heavily-updated KSDS data sets.

VSAM data sets that are to use this facility must reside on SMS-managed DASD, and must have an ICF catalog structure. Only VSAM ESDS, RRDS (both fixed and variable), and KSDS data sets are supported.

About this task

For DFSMS 1.3, there are two ways of defining BWO:

Procedure

- Define the cluster with parameter **BWO**=TYPECICS. This value means that the cluster is eligible for BWO in a CICS region. The file resource definition is ignored, even if it conflicts.

  Clusters with data sets that are to be opened in RLS mode must have BWO specified in the cluster definition.
- If the BWO parameter is not defined, it defaults to UNDEFINED. In this case, CICS looks at the file resource definition.

  CICS defines a data set as eligible for BWO when a file is defined using RDO. If BACKUPTYPE=DYNAMIC is specified for a VSAM file, the file is defined as eligible for BWO when the data set is opened.

  If DFSMSdss is to back up a data set that is specified with BACKUPTYPE=STATIC, all CICS files currently open for update against that data set must be closed before the backup can start.

Results

CICS records the fact that a VSAM base cluster data set is eligible for BWO in its base cluster block. This is remembered when all files have closed against the VSAM base cluster and across CICS warm and emergency restarts. It is not remembered across CICS cold or initial starts.

What to do next

You must put appropriate procedures into place for BWO and for forward recovery. These procedures must include:

- Restoring the BWO backup and running the forward recovery utility to bring the data set to a point of consistency. The restore requires that users do not have access to the file during the recovery process.
- Restoring and forward recovery of data sets that might have been damaged while allocated to CICS. This operation might require backout of partially committed units of work, by CICS emergency restart.

Effect of disabling activity keypointing:

If you disable activity keypointing in your CICS region by specifying the system initialization parameter **AKPFREQ**=0, there is a serious effect on BWO support for non-RLS activities.

When activity keypointing is disabled, no tie-up records (TURs) are written to the forward recovery logs and the data set recovery point is not updated. Therefore, forward recovery of a BWO backup must take place from the time that the data set was first opened for update. This requires that all forward recovery logs are kept since that time so that forward recovery can take place. If there are many inserts or records that change length, a lot of forward recovery could be required. If, however, a record is just updated and the length is unchanged, there is no CI split.

For information about TURs and recovery points, see BWO and concurrent copy .

Restrictions on BWO:

The following restrictions apply to VSAM KSDS data set types only.

If a VSAM control interval or control area split occurs while a BWO is in progress, the backup is unreliable and is discarded by DFHSM and DFDSS. During such a split, certain portions of the data set may be duplicated or not represented at all in the backup as DFDSS copies sequentially. MVS/DFP 3.2 indicates that a split has occurred in the ICF catalog. At the end of the backup, DFHSM and DFDSS check the ICF catalog, and if a split has occurred, or is still in progress, discard the backup. For this reason, certain heavily-updated VSAM KSDS data sets may not be eligible for BWO, or might be eligible only during periods of reduced activity (for example, overnight). For a KSDS data set to be eligible for BWO, the typical time between control interval or control area splits must be greater than the time taken for DFHSM and DFDSS to take a backup of the data set.

Using storage management facilities

BWO requires a number of storage management facilities.

About this task

These storage management facilities are :

- Storage management subsystem (SMS), part of MVS/DFP Version 3 Release 2 or later, product number 5665-XA3.
- Data facility hierarchical storage manager (DFHSM), product number 5665-329.
- Data facility data set services (DFDSS), product number 5665-327.

For more information about these facilities see the following sections.

Storage management subsystem (SMS):

SMS is the approach to DASD storage management in which CICS, by means of the storage management subsystem, determines data placement. In addition, an automatic data manager handles data backup, movement, space, and security.

This is sometimes referred to as DFSMS and complements functions of MVS/DFP and other individual products of the Data Facility product family. For more details about SMS, see the following publications:

- *MVS/DFP Version 3 Release 2: Storage Administration Reference*, SC26-4566
  This describes storage administrator applications.
- *MVS/DFP Version 3 Release 2: General Information*, SC26-4552
  This gives an overview of MVS/DFP and its requirements and describes concepts of SMS-managed storage.

Message on recall of backed up data sets:

If you use DFHSMS to manage your VSAM data sets, you should consider carefully the period after which your CICS VSAM data sets are migrated to primary or auxiliary storage.

If a migrated data set has to be recalled for CICS, it can take several minutes from primary storage, or longer from auxiliary storage. While the recall is taking place, the user is locked out, and no other opens or closes on that data set can be performed until the data set has been recalled.

If a migrated data set has to be recalled, CICS issues message DFHFC0989 to the system console, to notify the user that a recall is taking place, and to indicate whether it is from primary or auxiliary storage.

Data facility hierarchical storage manager (DFHSM):

DFHSM is an IBM-licensed program to manage volumes and data sets.

For more details about DFHSM, see the *Data Facility Hierarchical Storage Manager Version 2 Release 5.0: General Information*, GH35-0092. This manual discusses the main features, options, and potential benefits of DFHSM, and addresses people who want to know what the DFHSM program can do for them.

Data facility data set services (DFDSS):

DFDSS is an IBM-licensed program used to copy, move, dump, and restore data sets and volumes.

For more details about DFDSS, see the *Data Facility Data Set Services Version 2 Release 5.0: General Information*, GC26-4123. This manual introduces DFDSS and helps you evaluate its use.

Setting up temporary storage data sets

Each CICS region uses an area of 64-bit (above the bar) or 31-bit (above the line) storage in the CICS region as main temporary storage. You can define some auxiliary temporary storage for the CICS region in a nonindexed VSAM data set. Applications in a CICS region can also use shared temporary storage pools in a z/OS coupling facility.

About this task

For an overview of the different temporary storage locations and the features available for temporary storage queues in each location, see CICS temporary storage overview in Improving performance.

Compared to main temporary storage, auxiliary temporary storage is typically used by applications that store data for longer periods, or access the data less frequently. Temporary storage queues in auxiliary storage can also be defined as recoverable, which enables you to maintain data from one CICS run to the next. Temporary storage queues in main storage cannot be defined as recoverable.

Shared temporary storage pools in a coupling facility require temporary storage servers, typically one server in each z/OS image in the sysplex. However, they do not use any storage in the CICS region. Access to temporary storage queues in shared temporary storage pools is quicker than access to temporary storage queues held by a remote CICS region (queue-owning region). Shared temporary storage pools do not cause intertransaction affinities. For more information, see "Defining temporary storage pools for temporary storage data sharing" on page 153.

Defining the auxiliary temporary storage data set

You can either define a VSAM data set for auxiliary temporary storage as a single extent data set on a single volume using the sample job described here, or use the CICS-supplied job DFHDEFDS. DFHDEFDS creates the DFHTEMP data set as one of the data sets for a CICS region.

About this task

You must not define any extra associations for a temporary storage data set. For example, do not define a PATH. Doing so causes CICS startup to fail. Do not allocate the DFHTEMP data set from an SMS data class using extended addressability because CICS does not support this.

Procedure

1. Consider adding multiple extents and multiple volumes for the data set. The sample job produces a single-extent data set defined on a single volume, but you might experience channel and arm contention if auxiliary temporary storage is used heavily. To use DASD space more efficiently, you could define the DFHTEMP data set with a primary extent large enough for normal activity, and with secondary extents for exceptional circumstances, such as unexpected peaks in activity. For instructions to define further extents or volumes, see "Defining data sets with multiple extents and volumes" on page 25.
2. Specify a **RECORDSIZE** value that is 7 bytes less than the **CONTROLINTERVALSIZE**. You must specify the amount of space for temporary storage in two values:
   a. The control interval size. See "Control interval size for auxiliary temporary storage" on page 32 for information about how to calculate the space.

b. The number of control intervals in the data set. See "Number of control intervals for auxiliary temporary storage" on page 33 for information about how to set the correct number of control intervals.

3. Add a data definition statement for the DFHTEMP data set to the startup job stream. The temporary storage data set is a passively shared data set, owned by the active CICS region, but allocated to both the active and alternate CICS regions. Although the alternate CICS region does not open this data set before takeover, it is allocated at job step initiation, so you must specify DISP=SHR on the DD statement to enable the alternate CICS region to start.

```
//DFHTEMP  DD  DSN=CICSTS54.CICS.applid.DFHTEMP,DISP=SHR
```

Example

```
//DEFTS    JOB accounting info,name
//AUXTEMP  EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  *
   DEFINE CLUSTER(NAME(CICSTS54.CICS.CNTL.CICSqualifier.DFHTEMP)-
          RECORDSIZE(4089,4089)           -
          RECORDS(200)                    -
          NONINDEXED                      -
          CONTROLINTERVALSIZE(4096)       -
          SHAREOPTIONS(2 3)               -
          VOLUMES(volid))                 -
        DATA(NAME(CICSTS54.CICS.CNTL.CICSqualifier.DFHTEMP.DATA) -
          UNIQUE)
/*
```

Figure 5. Sample job defining an auxiliary temporary storage data set

What to do next

Use the **TS** system initialization parameter to specify an appropriate number of VSAM buffers and strings for auxiliary temporary storage. CICS uses each VSAM buffer to make a control interval from DFHTEMP available in CICS storage, and uses a VSAM string for each VSAM I/O request between a buffer and DFHTEMP. Typically, the default setting of three buffers and three strings is sufficient. For information about the performance considerations, see Auxiliary temporary storage: monitoring and tuning.

Control interval size for auxiliary temporary storage:

You specify the control interval size for the auxiliary temporary storage data set 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.

A temporary storage record is a single numbered item in a temporary storage queue, which might be written by CICS or an application. A temporary storage record must have the following space:

- 36-bytes for the temporary storage header.
- The length of the data in the temporary storage record (the item in the temporary storage queue). If you are using 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. Make sure that you allow sufficient space for the data length used by large-screen devices.

The total number of bytes allocated for a temporary storage record (including the 36-byte header) must be rounded up to a multiple of 64 for control interval sizes less than, or equal to, 16 KB (16 384-bytes), or a multiple of 128 for larger control interval sizes.

CICS can process temporary storage records that exceed the control interval size, but performance might degrade in this situation. Choose a control interval size large enough to hold at least one instance of the largest normally occurring temporary storage record, together with the VSAM control information for the control interval.

Typically, a control interval contains more than one temporary storage record, and the records might be of different sizes. The control interval size affects transfer efficiency: a smaller size can improve performance if access to temporary storage is random, and a larger size can improve performance if applications tend to use items in temporary storage in a sequential way. In general, the larger the queues and write to read ratio, the more sequential the usage tends to be.

Select your exact control interval size following these rules:

- The maximum control interval size is 32 KB.
- A control interval size less than, or equal to, 16 KB (16 384-bytes) must include space for 64-bytes of VSAM control information in addition to the space allowed for temporary storage records.
- A control interval size greater than 16 KB (16 384-bytes) must include space for 128-bytes of VSAM control information in addition to the space allowed for temporary storage records.
- A control interval size smaller than 8 KB must be a multiple of 512-bytes.
- A control interval size equal to or larger than 8 KB must be a multiple of 2 KB.

Example

If you use BMS to write a 24 x 80 character screen to temporary storage, the data written occupies 1920-bytes. You require 36-bytes for the CICS temporary storage header, giving a total of 1956-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 to hold a single record. You can select a control interval size larger than 2048-bytes to hold several records that might differ in size.

Number of control intervals for auxiliary temporary storage:

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.

Setting up data sets for transient data

Data sets that are used for transient data queues can be intrapartition or extrapartition. The transient data intrapartition data set is a VSAM entry-sequenced data set (ESDS) that is used for queuing messages and data within the CICS region. Transient data extrapartition data sets are sequential files, normally on disk or tape; you can use the queue to send and receive data outside the CICS region.

About this task

Messages or other data are addressed to a symbolic queue that you define as either intrapartition or extrapartition using the CEDA transaction. The queues can be used as indirect destinations to route messages or data to other queues. System messages that CICS produces are commonly sent to transient data queues, either intrapartition or extrapartition.

Procedure

1. Define all of the queues that CICS uses in your CICS region. 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.
   a. Use the sample definitions in group DFHDCTG to define the required queues. See TDQUEUE resources for a summary.
   b. Take a backup copy of the changes that you made to DFHDCTG. When maintenance is applied, the DFHDCTG group might be refreshed, overwriting your changes. Taking a backup avoids this problem.
2. Define the intrapartition data set using job control statements. See "Defining the intrapartition data set" for details on how to do this step.
3. Define the extrapartition data sets using job control statements. See "Defining extrapartition data sets" on page 36 for details on how to do this step.
4. To print CICS system messages on a local printer as they occur, use the transient data write-to-terminal sample program, DFH$TDWT. This sample program is supplied with the CICS in CICSTS54.CICS.SDFHLOAD and the assembler source is in CICSTS54.CICS.SDFHSAMP.

Defining the intrapartition data set

You can either use the sample job described here to define the transient data intrapartition data set, or you can use the CICS-supplied job, DFHDEFDS. DFHDEFDS creates the DFHINTRA data set as one of the data sets for a CICS region

About this task

The intrapartition data set must be big enough to hold all the data for intrapartition queues. If you are using the sample job to define the transient data intrapartition data set, perform the following steps:

Procedure

1. Decide if a single extent data set on a single volume is appropriate. If you define one extent on one volume, you might require a much larger data set than your average requirements to cater for exceptional cases. You can define multiple extents or multiple volumes or both for the data set. For details, see "Using multiple extents and multiple volumes" on page 36.
2. Specify a **CONTROLINTERVALSIZE** parameter that is large enough to hold the longest data record, in addition to the 32 bytes that CICS requires for its own

purposes. The maximum control interval size is 32 KB. Space is allocated to queues in units of a control interval (CI). 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.

3. If you allocate space in records, rather than tracks or cylinders, you must specify a RECORDSIZE value. The value must be 7 bytes less than the **CONTROLINTERVALSIZE**.
4. Add the data definition statement for the intrapartition data set to the CICS startup job stream. The DD name for the intrapartition data set is DFHINTRA, and the DSN operand must be the name of the VSAM entry-sequenced data set. For example, you could specify:

   ```
   //DFHINTRA  DD  DSNAME=CICSTS54.CICS.applid.DFHINTRA,DISP={OLD|SHR}
   ```

Example

```
//DEFDS     JOB  accounting info,name,MSGCLASS=A
//TDINTRA   EXEC PGM=IDCAMS
//SYSPRINT  DD   SYSOUT=A
//SYSIN     DD   *
       DEFINE CLUSTER -
                ( NAME(CICSTS54.CICS.applid.DFHINTRA)    -
                  RECORDSIZE(1529,1529)            -
                  RECORDS(100)                     -
                  NONINDEXED                       -
                  CONTROLINTERVALSIZE(1536)        -
                  VOL(volid))                      -
              DATA  -
                ( NAME(CICSTS54.CICS.applid.DATA.DFHINTRA))
/*
//
```

Figure 6. Sample job to define a transient data intrapartition data set

What to do next

Use the **TD** system initialization parameter to specify an appropriate number of VSAM buffers and strings for the transient data intrapartition data set. CICS uses buffers to make control intervals from the data set available in CICS storage, and uses strings for VSAM I/O requests between a buffer and the data set. Typically, the default setting of three buffers and three strings is sufficient.

What happens if the intrapartition data set fails to open:

The DFHINTRA data set is opened during CICS initialization, regardless of the presence or absence of any intrapartition queue definitions that might become active during GRPLIST installation.

If DFHINTRA fails to open during an initial or cold start of CICS, a message is issued informing you of this, and asks if you want to continue or if you want to cancel CICS.

If DFHINTRA opened successfully during a previous startup but fails to open during a subsequent warm or emergency restart, CICS is terminated.

If CICS initializes without a DFHINTRA data set, any attempts to install intrapartition data destinations for that run of CICS fails and appropriate error messages are issued.

Using multiple extents and multiple volumes:

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.

The job control statements in Figure 6 on page 35 are for a single extent data set defined on a single volume. That data set must be big enough to hold all your data. For more information about defining these, see "Defining data sets with multiple extents and volumes" on page 25.

Size of the intrapartition data set:

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 READQ TD or DELETEQ TD requests.

The intrapartition data set should hold at least two control intervals. When a logically recoverable queue is read until a QZERO condition is returned, and the request is committed, CICS retains the last CI used by the queue (unless there is no further space between the end of the last record and the end of the CI). Retaining the final CI used by the queue, means that subsequent requests to write to the queue can be accommodated in the remaining space within the CI if they can fit there. This avoids the need to acquire a new CI whenever the first request is made to write to the queue following a QZERO condition, which benefits performance on subsequent write requests. However, the CI is left allocated to the queue, and so increases the usage of the data set, and the possibility of a NOSPACE condition being returned by CICS.

Intrapartition data set restrictions:

An intrapartition data set used as a transient data queue must be associated with only one CICS region.

CICS stores the relative byte addresses (RBAs) of records written to an intrapartition data set used as a transient data queue, and you must take care 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.

Data can be corrupted or lost if you use VSAM export or import services to increase the available space by compressing the data set, or to increase the control interval size.

Do not use the extended addressing entry-sequenced data set (ESDS) format for an intrapartition data set.

Defining extrapartition data sets

You can define each extrapartition data set either for input only or for output only, but not for both.

About this task

You should define transient data extrapartition data sets used as queues for CICS messages with a record length of 132 bytes and a record format of V or VB. If you

use the FREE=CLOSE parameter for an input extrapartition entry, be aware that it will only be usable once in the CICS session. Any attempt to read the queue after it has been CLOSEd and OPENed again will result in the IOERR condition being raised.

Should you decide to specify VBS for the extrapartition data set, be aware that BFTEK=A should be specified on the definition in the JCL. When variable-length spanned records are processed without the extended logical record interface (XLRI), and a record area is provided for a logical record interface (LRI) (BFTEK=A has been specified in the data control clock), QSAM will provide an area large enough to contain the maximum record size (up to 32756 bytes). Failure to specify BFTEK=A on such data set definitions can result in run time errors such as abend 002 returned from the access method.

Example

```
//LOGUSR  DD   DSN=CICSTS54.CICS.applid.LOGUSR,DISP=(NEW,KEEP),
//         DCB=(DSORG=PS,RECFM=V,BLKSIZE=136),
//         VOL=SER=volid,UNIT=3380,SPACE=(CYL,2)
//MSGUSR  DD   SYSOUT=A
```

Figure 7. Sample JCL to define transient data extrapartition data sets

The DFHCXRF data set:

Besides any extrapartition data sets that you might define, there is a special extrapartition queue that CICS creates dynamically. This queue has the identifier CXRF and is created by CICS early in the initialization process.

The DD name for this extrapartition data set is DFHCXRF. You cannot define CXRF or DFHCXRF. If you code DFHCXRF as a DSCNAME, or code CXRF as a destination identifier, an error message is issued. If you create a definition for CXRF in the CSD, CICS does not install the definition. This is because the CICS entry is hardcoded and cannot be removed or replaced.

If an attempt is made to write to a CICS-defined transient data queue before CICS is fully initialized, a message is written to CXRF.

If, on an initial or cold start, a request is received to write a record to a queue that has not yet been installed as part of GRPLIST, the record is written to CXRF.

If an attempt is made to write to an intrapartition queue after the warm keypoint has been taken, the record is written to CXRF.

Active CICS regions

CICS uses the CXRF queue during CICS initialization as some CICS components might need to write to transient data queues.

If the queues are not available at initialization time, the request to write to these queues is redirected to CXRF. Any requests from CICS components to write to transient data before the CXRF queue definition has been installed fails with a QIDERR condition.

Any requests to write to an intrapartition transient data queue after the warm keypoint has been taken during a normal shutdown are routed to CXRF.

If you want to take advantage of the special CXRF queue, you must include a DD statement for DFHCXRF. If you omit the DD statement, transient data write requests redirected to CXRF fail with a NOTOPEN condition.

DFHCXRF DD statements

You can define the DFHCXRF data set to MVS in the same way as other transient data extrapartition data sets, either to a SYSOUT data set or a sequential data set on disk (or tape). For example, you could use either of the DD statements shown in the following example in a startup job stream for a CICS region:

```
//DFHCXRF  DD SYSOUT=*

or
//DFHCXRF  DD DSN=CICSTS54.CICS.applid.DFHCXRF,DISP=(NEW,KEEP),
//                        DCB=(DSORG=PS,RECFM=V,BLKSIZE=136),
//                        VOL=SER=volid,UNIT=3380,SPACE=(TRK,5)
```

Figure 8. Sample DD statements for DFHCXRF

Setting up CICS log streams

Create CICS log streams to use the MVS system logger to record journaling and logging information.

About this task

CICS log manager supports:

- The CICS system log, which is also used for dynamic transaction backout.
- User journals, forward recovery logs, and auto-journals. These are general logs.

Defining CICS system logs

Each CICS region requires its own system log. The system log is implemented as two MVS system logger log streams - a primary and a secondary - but, together, they form a single logical log stream.

About this task

The system log is used for recovery purposes - for example, during dynamic transaction backout, or during emergency restart, and is not meant to be used for any other purpose.

CICS connects to its system log automatically during initialization (unless you specify a journal model definition that defines the system log as type DUMMY).

You must define a system log if you want to preserve data integrity in the event of unit of work failures and CICS failures. CICS needs a system log in order to perform:

- The backout of recoverable resources changed by failed units of work.
- Cold starts when CICS needs to recover conversation state data with remote partners.
- Warm restarts, when CICS needs to restore the region to its pre-shutdown state.
- Emergency restarts, when CICS needs to restore the region to its pre-shutdown state as well as recovering transactions to perform the backout of recoverable resources changed by units of work that were in-flight at the time of shutdown.

For information on how to define CICS system log streams, see Coupling facility log streams and DASD-only log streams.

Planning your CICS system log streams:

A CICS system log (which comprises two physical log streams) is unique to the region and must not be used by any other CICS region. The default log stream names *region_userid.applid*.DFHLOG and *region_userid.applid*.DFHSHUNT are designed to ensure unique names.

Using JOURNALMODELs to define the system log:

You might want to use journal models for system logs if the CICS region userid changes between runs (for example, where a test CICS region is shared between application developers).

It would be wasteful to create log streams with a different high level qualifier for each user. Using the same system log stream regardless of the which programmer starts up the CICS region keeps the number of log streams to a minimum. The following example uses a specific JOURNALNAME, with symbols in the STREAMNAME, making this an explicit model for the primary log stream.

```
DEFINE GROUP(TEST) DESC('System logs for test CICS regions')
       JOURNALMODEL(DFHLOG) JOURNALNAME(DFHLOG) TYPE(MVS)
       STREAMNAME(TESTCICS.&APPLID..&JNAME.)
```

If you define JOURNALMODEL resource definitions to define log stream names for DFHLOG and DFHSHUNT, ensure that the resulting log stream names are unique. If you have some CICS regions that use the same applid, you must use some other qualifier in the log stream name to ensure uniqueness.

If you use JOURNALMODEL resource definitions for the system log, these resource definitions must be defined and added to the appropriate group list (using the CSD utility program, DFHCSDUP) before INITIAL-starting CICS.

System logs cannot be TYPE(SMF).

DFHLOG can be TYPE(DUMMY), but you can use this only if you always INITIAL start your CICS regions and there are no recoverable resources requiring transaction backout. CICS cannot perform a cold start, or warm or emergency restart if TYPE(DUMMY) is specified on the JOURNALMODEL definition.

If you do not want to use a system log, perhaps in a test or development region, define a JOURNALMODEL for DFHLOG with type DUMMY, as shown in the following example:

```
DEFINE  JOURNALMODEL(DFHLOG)   GROUP(CICSLOGS)
        JOURNALNAME(DFHLOG)
        TYPE(DUMMY)
```

To start a CICS region without a system log, you must ensure that a JOURNALMODEL definition, such as the one shown above, is included in the start-up group list. Use the DFHCSDUP batch utility program to define the required JOURNALMODEL and to add the group to the group list.

DFHSHUNT can be TYPE(DUMMY). This is not recommended, however, because it impairs the ability of CICS to manage the system log.

Effects of the AKPFREQ parameter:

Review the activity keypoint frequency (AKPFREQ) defined for each CICS region. The larger the value, the more coupling facility space you need for the system logs, but you should not set AKPFREQ too low so that transactions last longer than an activity keypoint interval.

Defining CICS general logs

Journals on general log streams comprise user journals, forward recovery logs, and autojournals.

Planning log streams for use by your user journals and autojournals:

General logs are identified as such by their MVS log stream names, which are differentiated from system log streams in that their names do not end with DFHLOG or DFHSHUNT.

Using JOURNALMODELs to define general logs:

If you are running multiple cloned copies of your application-owning regions (AORs), it is probable that the logged data is common and that you would want to merge the data from all of the AORs to the same log stream.

The following JOURNALMODEL resource definition maps CICS journals of the same journal identifier to a shared log stream:

```
DEFINE GROUP(MERGE) DESC('Merge journals across cloned CICS regions')
       JOURNALMODEL(JRNLS) JOURNALNAME(DFHJ*) TYPE(MVS)
       STREAMNAME(&USERID..SHARED.&JNAME.)
```

In this example, the literal SHARED is used in place of the default CICS applid, which would require a unique log stream for each region.

You might want to use JOURNALMODELs to map journals to log streams if the CICS region userid changes between runs. This could be the case, for example, where CICS test regions are shared between groups of developers. It would be wasteful to create log streams with a different high level qualifier for each user and you might prefer to use the same log streams regardless of which developer starts up a CICS region. For example, the following generic JOURNALMODEL definition maps all journals not defined by more explicit definitions to the same log stream

```
DEFINE GROUP (TEST) DESC('Journals for test CICS regions')
       JOURNALMODEL(JRNLS) JOURNALNAME(*) TYPE(MVS)
       STREAMNAME(TESTCICS.&APPLID..&JNAME.)
```

You might want to merge data written by CICS regions using different journal names to a single log stream.

```
DEFINE GROUP (TEST) DESC('Merging journals 10 to 19')
       JOURNALMODEL(J10TO19) JOURNALNAME(DFHJ1*) TYPE(MVS)
       STREAMNAME(&USERID..MERGED.JNLS)
DEFINE GROUP (TEST) DESC('Merging journalnames JNLxxxxx')
       JOURNALMODEL(JNLXXXXX) JOURNALNAME(JNL*) TYPE(MVS)
       STREAMNAME(&USERID..MERGED.JNLS)
```

The last qualifier of the stream name is used as the CICS resource name for dispatcher waits. Therefore, if it is self-explanatory, it can be helpful when interpreting monitoring information and CICS trace entries.

Planning log streams for use by your forward recovery logs:

CICS performs the logging for RLS and non-RLS data sets.

You can share a forward recovery log stream between multiple data sets - you do not need to define a log stream for each forward-recoverable data set. Your decision is a balance of transaction performance, rapid recovery, and the work involved in managing a large number of log streams.

For a data set open in RLS mode, the MVS logger merges all the forward recovery log records from the various CICS systems on to the shared forward recovery log.

Some points to consider are:

- All data sets used by one transaction should use the same log stream (to reduce the number of log streams written to at syncpoint).
- A good starting point is to use the same forward recovery log ID that you use in an earlier CICS release.
- Share a forward recovery log stream between data sets that:
  - Have similar security requirements
  - Have similar backup frequency
  - Are likely to need restoring in their entirety at the same time
- Log stream names should relate to the data sets. For example, PAYROLL.data_sets could be mapped to a forward recovery log named PAYROLL.FWDRECOV.PAYLOG.
- The last qualifier of the stream name is used as the CICS resource name for dispatcher waits. Therefore, if it is self-explanatory, it can be helpful when interpreting monitoring information and CICS trace entries.
- Don't mix high update frequency data sets with low update frequency data sets, because this causes a disproportionate amount of unwanted log data to be read during recovery of low frequency data sets.
- Don't put all high update frequency data sets on a single log stream because you could exceed the throughput capacity of the stream.
- If you define too many data sets to a single log stream, you could experience frequent structure-full events when the log stream can't keep up with data flow.
- Redundant data should be deleted from log streams periodically. On OS/390® Release 2 or earlier, this is a user responsibility, and must be done before the system logger inventory entry exceeds the limit of 168 data sets per log stream. On OS/390 Release 3 or later, you can specify that redundant data is to be deleted automatically from general log streams, by defining them with AUTODELETE(YES) RETPD(dddd). For information about the AUTODELETE and RETPD MVS parameters, see Definitions required for VSAM RLS support>.

  Typically, for a forward recovery log, deletion of old data is related to the data backup frequency. For example, you might keep the 4 most recent generations of backup, and when you delete a redundant backup generation you should also delete the corresponding redundant forward recovery log records. These are the records older than the redundant backup because they are no longer needed for forward recovery.

  For information about IBM CICS VSAM Recovery, see CICS VSAM Recovery for z/OS.

Planning log streams for use by your log of logs (DFHLGLOG):

The log of logs is written by CICS to provide information to forward recovery programs such as CICS VSAM Recovery.

The log of logs is a form of user journal containing copies of the tie-up records written to forward recovery logs. Thus it provides a summary of which recoverable VSAM data sets CICS has used, when they were used, and to which log stream the forward recovery log records were written.

If you have a forward recovery product that can utilize the log of logs, you should ensure that all CICS regions sharing the recoverable data sets write to the same log of logs log stream.

```
DEFINE GROUP(JRNL) DESC('Merge log of logs')
       JOURNALMODEL(DFHLGLOG) JOURNALNAME(DFHLGLOG) TYPE(MVS)
       STREAMNAME(&USERID..CICSVR.DFHLGLOG)
```

Note: Note that this definition is supplied in group DFHLGMOD in DFHLIST.

If you don't have a forward recovery product that can utilize the log of logs you can use a dummy log stream:

```
DEFINE GROUP(JRNL) DESC('Dummy log of logs')
       JOURNALMODEL(DFHLGLOG) JOURNALNAME(DFHLGLOG) TYPE(DUMMY)
```

In addition to forward recovery, the log of logs is also used for recording log stream errors. When a log stream failure occurs, CICS tries to update the log of logs with the error and diagnostic information, unless DFHLGLOG is defined as TYPE(DUMMY).

Do not share the log of logs between test and production CICS regions, because it could be misused to compromise the contents of production data sets during a restore.

Naming journals

The journals have the following naming conventions.

About this task

System logs:

DFHLOG and DFHSHUNT are the journal names for the CICS system log.

CICS automatically creates journal table entries for DFHLOG and DFHSHUNT during initialization as shown in Table 4.

Table 4. Journal name entry for the CICS primary system log

Journal table entry - CICS system log	**Created during system initialization**
Name: DFHLOG	Always DFHLOG for the primary log
Status: Enabled	Set when journal entry created
Type: MVS	The default, but it can be defined as DUMMY on JOURNALMODEL definition (DUMMY = no output)
LSN: log_stream_name	By default, log_stream_name resolves to &reguserid..&applid..DFHLOG, but this can be user-defined on a JOURNALMODEL definition

Forward recovery logs:

For non-RLS data sets that have not specified their recovery attributes in the VSAM catalog, forward recovery log names are of the form DFHJ*nn* where *nn* is a number in the range 1–99.

You define the name of the forward recovery log in the forward recovery log id (FWDRECOVLOG) in the FILE resource definition.

User applications can use a forward recovery log through a user journal name that maps on to the same log stream name. In this case, the user records are merged on to the forward recovery log. See Table 5 for an example of this.

Table 5. Example of journal name entry for non-RLS mode forward recovery log

Journal table entry - forward recovery log	**Entry created during file-open processing**
Name: DFHJ01	Name derived from FWDRECOVLOG identifier. For example, FWDRECOVLOG(01) = DFHJ01, thus FWDRECOVLOG(nn) = DFHJnn
Status: Enabled	Set when journal entry created
Type: MVS	The default, but it can be defined as DUMMY on JOURNALMODEL definition (DUMMY = no output)
LSN: log_stream_name	By default, log_stream_name resolves to &reguserid..&applid..DFHJ01, but this can be user-defined on a JOURNALMODEL definition

Note: There is no journal table entry for the forward recovery log of an RLS file. The recovery attributes and LSN are obtained directly from the VSAM catalog, and the LSN is referenced directly by CICS file control. Therefore there is no need for indirect mapping through a journal name.

You can also choose to specify the recovery attributes and LSN for a non-RLS file in the VSAM catalog.

User journals:

CICS user journals are identified by their journal names (or number, in the case of DFHJ*nn* names), which map on to MVS log streams.

You name your user journals using any 1-8 characters that conform to the rules of a data set qualifier name. Apart from the user journal names that begin with the letters DFHJ, followed by two numeric characters, you should avoid using names that begin with DFH. User journal names of the form DFHJ*nn* are supported for compatibility with earlier CICS releases.

Although 8-character journal names offer considerable flexibility compared with the DFHJnn name format of previous releases, you are recommended not to create large numbers of journals (for example, by using the terminal name or userid as part of a program-generated name).

Journal name DFHLOG (on an EXEC CICS WRITE JOURNALNAME command) indicates that you want to write to the CICS system log.

When used in FILE and PROFILE resource definitions, the journal numbers 1 through 99 map on to the user journal names DFHJ01–99. You can map these journal names to specific MVS log stream names by specifying JOURNALMODEL resource definitions, or allow them to default. If you do not specify matching JOURNALMODEL definitions, by default user journals are mapped to LSNs of the form *userid.applid*.DFHJ*nn*.

Table 6 shows an example of a user journal name table entry.

Table 6. Example of a user journal name entry for output to MVS

Journal table entry - user journals	**Entry created on first reference**
Name: JRNL001	Name derived from API WRITE JOURNALNAME command
Status: Enabled	Set when journal entry created
Type: MVS	This journal is defined for MVS output by a JOURNALMODEL that references the JRNL001 name
LSN: log_stream_name	By default, log_stream_name resolves to &reguserid..&applid..JRNL001, but this can be user-defined on a JOURNALMODEL definition

Installing system log and journal names:

The journal control table of earlier releases is obsolete, and is replaced by a journal names table that CICS creates dynamically.

The CICS log manager needs the name of the log stream that corresponds to a CICS system log or general log, and the type - whether it is MVS, SMF, or a dummy. Except for VSAM forward recovery log stream names taken directly from the ICF catalog, CICS maintains this information in the journal names table, together with the corresponding log stream token that is returned by the MVS system logger when CICS successfully connects to the log stream.

Defining JOURNALMODELs

CICS uses JOURNALMODEL definitions to resolve log stream names.

About this task

CICS resolves log stream names at the following times:

System log
During initialization, on an initial start only.

On a cold, warm or emergency restart, CICS retrieves the log stream name from the CICS global catalog.

General logs
When a journal name is first referenced after the start of CICS, or when it is first referenced again after its log stream has been disconnected. Log stream disconnection, requiring further reference to a matching JOURNALMODEL resource definition, occurs as follows:

User journals
As soon as you issue a **DISCARD JOURNALNAME** command.

Any further references to the discarded journal name means that CICS must again resolve the log stream name by looking for a matching

JOURNALMODEL resource definition. You can change the log stream name for a user journal by installing a modified JOURNALMODEL definition.

Auto journals for files
All files that are using the log stream for autojournaling are closed.

Forward recovery logs
All files that are using the log stream for forward recovery logging are closed.

A JOURNALMODEL definition generally specifies a generic journal name, thereby mapping to the same MVS log stream any journal names that match on the generic name. JOURNALMODEL definitions can also be specific models and, using JOURNALMODEL definitions, you can map many journals or forward recovery logs to the same MVS log stream, or assign them to SMF (see Figure 9).

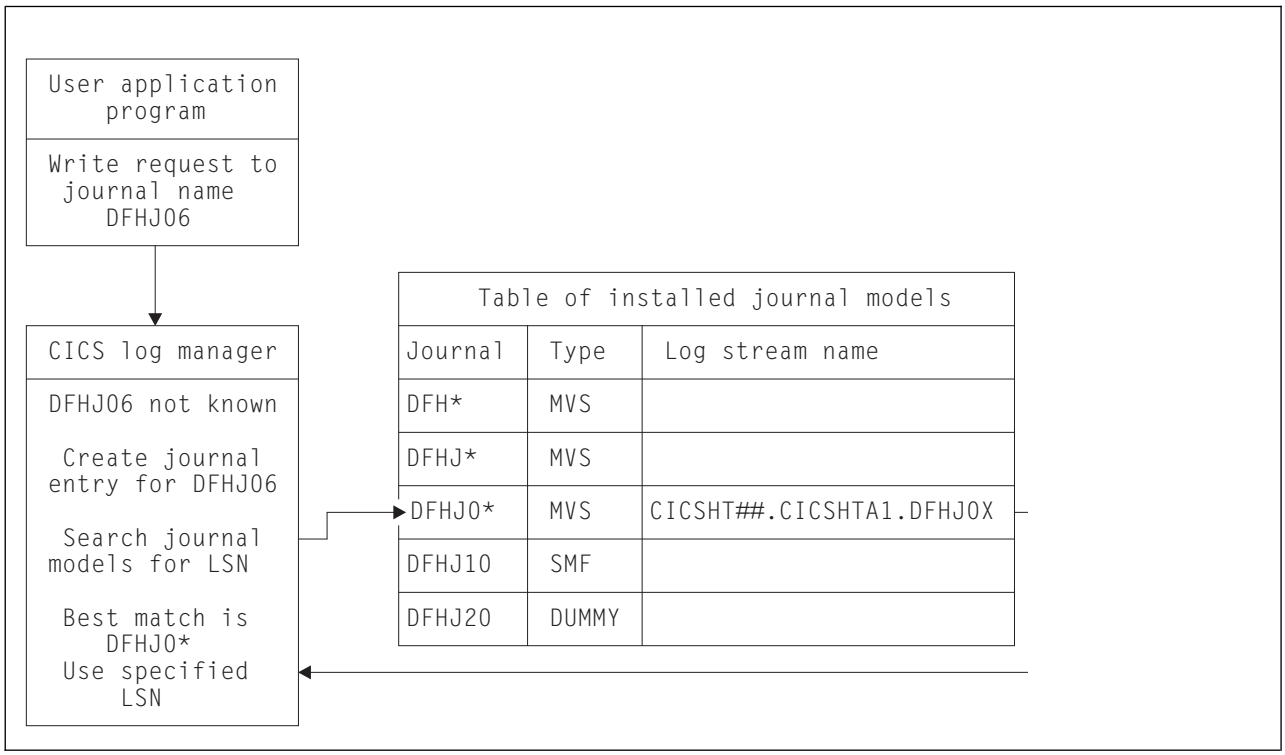


Figure 9. Looking for a journal model that matches a journal name. CICS searches the Table of installed journal models to find the log stream name that corresponds to the journal name, using a "best-match" algorithm.

Mapping log streams

Except for VSAM RLS forward recovery log stream names, which are obtained directly from the VSAM catalog, CICS maps the system log name or journal name to a corresponding log stream name.

About this task

CICS does this either by using a user-defined JOURNALMODEL resource definition (if one exists), or by using default names created by resolving symbolic names.

Mapping of the system log stream:

On a cold start, or warm or emergency restart, CICS retrieves the system log stream name from the CICS global catalog.

CICS uses the default log stream names unless you provide a JOURNALMODEL resource definition for the system log.

If there are JOURNALMODEL definitions for your system logs (CICS finds JOURNALMODEL definitions with JOURNALNAME(DFHLOG) and JOURNALNAME(DFHSHUNT)), it attempts to connect to the system log streams named in these definitions. System log stream names must be unique to the CICS region.

If you define JOURNALMODEL resource definitions for your system logs, ensure that:

- The log streams named in the definition are defined to the MVS system logger, or
- Suitable model log streams are defined so that they can be created dynamically.

If there are no suitable JOURNALMODEL definitions, CICS attempts to connect to system log streams, using the following default names:

- userid.applid.DFHLOG
- userid.applid.DFHSHUNT

where 'userid' is the RACF userid under which the CICS address space is running, and 'applid' is the region's z/OS Communications Server APPL name. The CSD group DFHLGMOD holds the CICS-supplied JOURNALMODEL definitions for the default DFHLOG and DFHSHUNT log streams.

Before you try to use these default log stream names, ensure that

- The default log streams are defined explicitly to the MVS system logger, or
- Suitable model log streams are defined so that they can be created dynamically.

If these log streams are not available (perhaps they have not been defined to MVS) or the definitions are not found (perhaps they have not been installed), CICS attempts to create system log streams using a model log stream named &*sysname*.*LSN_last_qualifier*.MODEL, where:

- &*sysname* is the MVS symbol that resolves to the system name of the MVS image
- *LSN_last_qualifier* is the last qualifier of the log stream name as specified on the JOURNALMODEL resource definition.

If you do not provide a JOURNALMODEL resource definition for DFHLOG and DFHSHUNT, or if you use the CICS definitions supplied in group DFHLGMOD, the model names default to &*sysname*.DFHLOG.MODEL and &*sysname*.DFHSHUNT.MODEL. Once these log streams have been created, CICS then connects to them.

Figure 10 on page 47 shows a graphical representation of the system log mapping process during an INITIAL start.

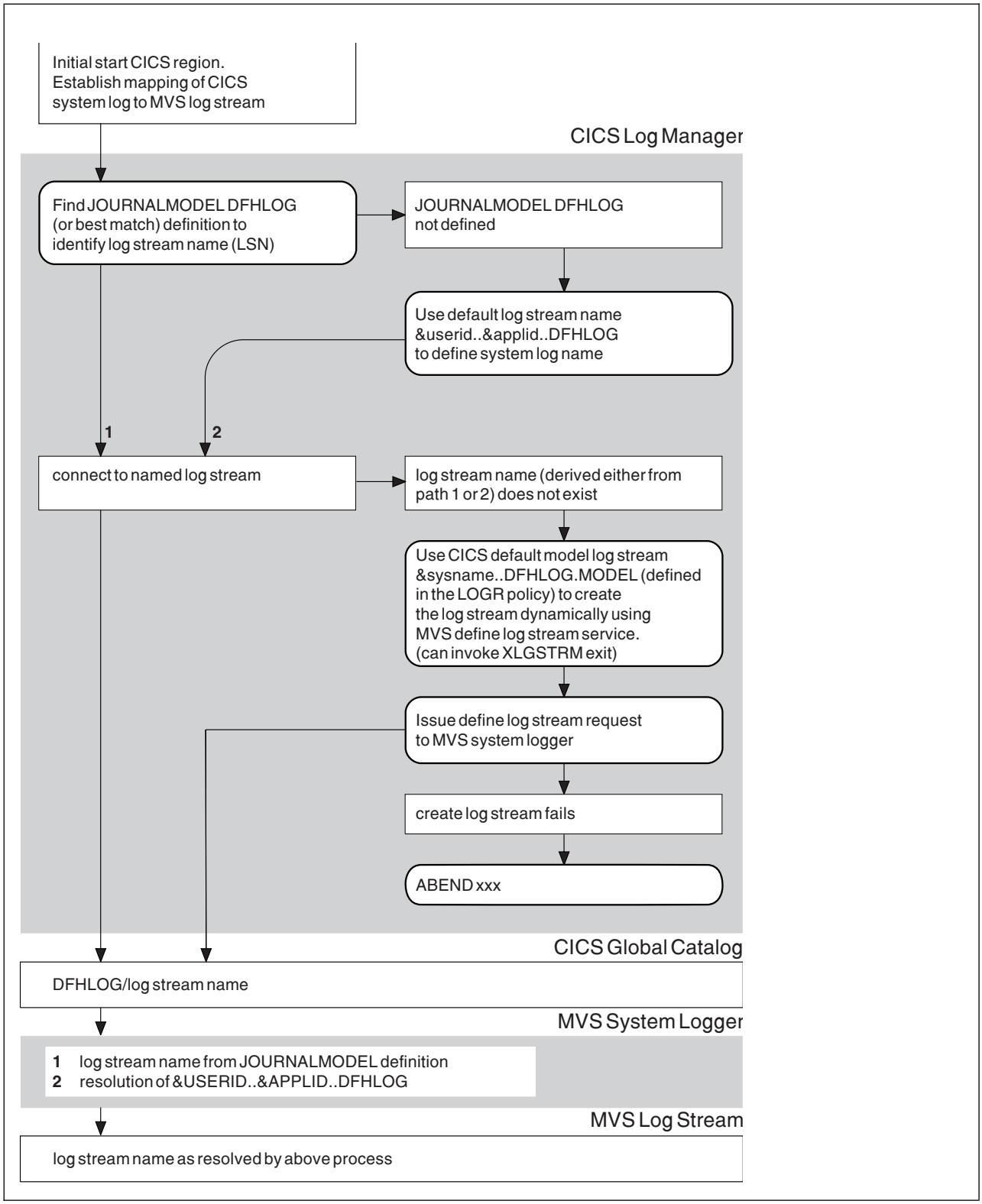


Figure 10. How CICS maps the system log (DFHLOG) to a log stream name (LSN) during an INITIAL start. CICS uses the same process for the secondary system log, DFHSHUNT.

Mapping of general log streams

CICS uses the default log stream names unless you provide a JOURNALMODEL resource definition for the journal or log.

If there is a JOURNALMODEL definition for the log, CICS attempts to connect to the log stream named in the definition.

If you define JOURNALMODEL resource definitions for your system logs, ensure that:

- The log streams named in the definition are defined to the MVS system logger, or
- Suitable model log streams are defined so that they can be created dynamically.

If CICS cannot connect to the log stream named in the JOURNALMODEL definition, it attempts to connect to a log stream, using the default name:

`userid.applid.`*`journalname`*

Before you try to use this default log stream name, ensure that

- The default log stream is defined explicitly to the MVS system logger, or
- A suitable model log stream is defined so that it can be created dynamically.

If the log stream is not available (perhaps it has not been defined to MVS) or the definition is not found (perhaps it has not been installed), CICS attempts to create a log stream using the default name:

`LSN_QUALIFIER1.LSN_QUALIFIER2.MODEL`

where the qualifier fields are based on the JOURNALMODEL definition streamname attribute, as follows:

- If the log stream being created has a qualified name consisting of only two names (*qualifier1.qualifier2*) or has an unqualified name, CICS constructs the model name as *qualifier1*.MODEL or *name*.MODEL.
- If the log stream being created has a qualified name consisting of 3 or more names (*qualifier1.qualifier2....qualifier_n*), CICS constructs the model name as *qualifier1.qualifier2*.MODEL.

Once the log stream has been created, CICS connects to it.

Figure 11 on page 49 shows a graphical representation of the mapping process for general logs.

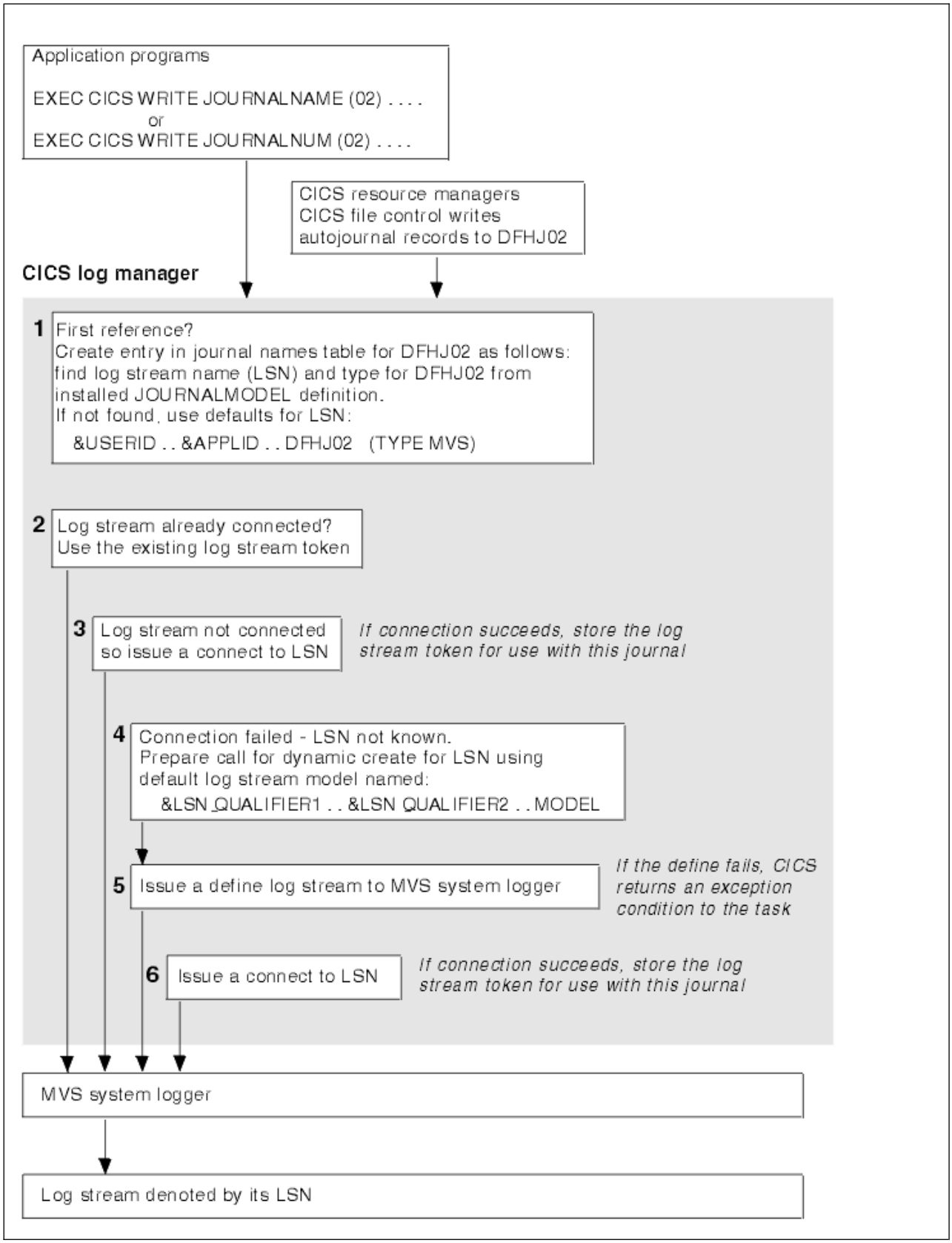


Figure 11. How a CICS journal is mapped to its log stream name (LSN). The name is DFHJ02, used here for user journaling and file control autojournaling.

Printing or copying data on MVS system logger log streams

You can use the CICS-supplied journal utility program DFHJUP to select, print, or copy data held on MVS system logger log streams. Alternatively, you can use your own utility.

About this task

The journal utility program DFHJUP uses the SUBSYS=(LOGR... facility. If you want to your own utility, your utility should use the SUBSYS=(LOGR... facility.

For information about running DFHJUP, and the SUBSYS=(LOGR.. , facility, see Reading log streams using batch jobs (DFHJUP).

Setting up the catalog data sets

You must define and initialize new CICS catalogs for CICS Transaction Server for z/OS, Version 5 Release 4 .

About this task

You must define the CICS global catalog data set and the CICS local catalog data set, which CICS requires to catalog CICS system information. Throughout the documentation, these data sets are referred to as the global catalog and the local catalog. The CICS catalog data sets are not connected with MVS system catalogs and contain data that is unique to CICS.

For information on how CICS uses the catalogs, including startup and restart, see "The role of the CICS catalogs" on page 114

To estimate the amount of space needed in your global catalog, see "Space calculations" on page 52

To set up the catalog data sets:

Procedure

1. Define the global catalog.
2. Define the local catalog.

Defining the global catalog

The global catalog is a VSAM key-sequenced data set (KSDS) is used to store start type information, location of the CICS system log, resource definitions, terminal control information and profiles.

About this task

CICS uses the global catalog to perform the following activities:

- To record information that governs the possible types of start and the location of the CICS system log.
- During the running of CICS, to hold the resource definitions that are installed during initialization when CICS installs the group list, by the RDO **CEDA INSTALL** command or by the **EXEC CICS CREATE** command.
- During a normal shutdown, to record terminal control information and profiles. All other warm keypoint information is written to the CICS system log.

You must ensure that the REGION parameter on your CICS jobs is high enough to cope with the increase in buffer storage used for the global catalog, because this storage comes out of region storage not EDSA.

You can define and initialize the CICS global catalog in two ways. You can use the sample job as described below, or you can use the CICS-supplied job, DFHDEFDS.

Edit the sample job:

Procedure

1. Edit the data set name in the CLUSTER definition to be the same as the DSN parameter in the DD 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 requirements, and substitute your values for *n1* and *n2*. Whichever **IDCAMS** parameter you use for the global catalog data set space allocation (CYLINDERS, TRACKS, or RECORDS), make sure that you specify a secondary extent. CICS abends if your global catalog data set fills and VSAM cannot create a secondary extent. For information about record sizes, see Table 7 on page 53
3. Specify the REUSE option on the **DEFINE CLUSTER** command. This option enables the global catalog to be opened repeatedly as a reusable cluster. Also specify REUSE if you intend to use the COLD_COPY input parameter of the DFHRMUTL utility.
4. Edit the *CONTROLINTERVALSIZE* values for the VSAM definition if required. This job does not specify a minimum or maximum buffer size explicitly, but accepts the default that is set by VSAM. You can code an explicit value if you want to define buffers of a specific size. See "Buffer space sizings" on page 55 for more information.
5. Use the recovery manager utility program, DFHRMUTL, to initialize the data set. Specify this utility in the job step INITGCD. DFHRMUTL writes a record to the data set, specifying that on its next run using this global catalog, if START=AUTO is specified, CICS is to perform an initial start and not prompt the operator for confirmation. This record is called the autostart override record.
6. Add a job step to run the DFHCCUTL utility. Adding this step means that the global and local catalogs do not get out of step.
7. Define the data definition statement for CICS as:

   ```
   //DFHGCD  DD  DSN=CICSTS54.CICS.applid.DFHGCD,DISP=OLD
   ```

   This example shows the minimum specification for a global catalog for use by a single CICS region.
8. Add the relevant **AMP** subparameters to the DD statement to help improve restart and shutdown time. The **AMP** parameter is described in *z/OS MVS JCL Reference* and an example is shown in the CICS startup job stream in "CICS startup" on page 126.

Example

```
//GLOCAT   JOB accounting info,,CLASS=A
//DEFGCD   EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
      DEFINE CLUSTER -
               (NAME(CICSTS54.CICS.applid.DFHGCD) -
               INDEXED                      -
               CYLINDERS(n1 n2)             -
               FREESPACE(10 10)             -
               SHAREOPTIONS(2)              -
               RECORDSIZE(4089 32760)       -
               REUSE                        -
               VOLUMES(volid))              -
             DATA                           -
               (NAME(CICSTS54.CICS.applid.DFHGCD.DATA)  -
               CONTROLINTERVALSIZE(32768)    -
               KEYS(52 0))                  -
             INDEX                          -
               (NAME(CICSTS54.CICS.applid.DFHGCD.INDEX) )
/*
//INITGCD  EXEC PGM=DFHRMUTL,REGION=1M
//STEPLIB  DD DSNAME=CICSTS54.CICS.SDFHLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//DFHGCD   DD DSNAME=CICSTS54.CICS.applid.DFHGCD,DISP=OLD
//SYSIN    DD *
SET_AUTO_START=AUTOINIT
/*
```

Figure 12. Example job to define and initialize the global catalog

Space calculations:

Estimate the amount of space needed in your global catalog to keypoint installed resource definitions, table entries, and control blocks. Each global catalog record has a 52-byte key. Each entry is one VSAM record, and the records for each type of table have different keys.

You must regularly review your space usage to ensure that extents are not running too close to the limit for your environment. Use the sizes specified in Table 7 on page 53 to estimate the amount of space needed in your global catalog.

The space requirements for a VSAM KSDS such as DFHGCD can vary for different CICS cold starts. They can vary when no changes have been made to the CICS definitions that are going to be stored on the VSAM KSDS, because VSAM uses 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 calls VSAM to perform sequential writes. VSAM honors 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, the **freespace** value is only honored after a CI split has occurred.

The size of the index portion of the data set can also vary depending on the number of CI and CA splits that have occurred. The size affects the index sequence set.

When you are initializing the global catalog, you can use the **COLD_COPY** parameter; for example, SET_AUTO_START=AUTOCOLD,COLD_COPY. The cold copy creates a reduced

copy of the global catalog data set, which improves the performance of the cold start. The CI splits cease after the first cold start and the data set will not expand into additional extents. Alternatively, you can reorganize or reinitialize the data set from time to time.

Note: In Table 1, the **Number of bytes per entry** column includes the 52-byte key length.

Table 7. Sizes for entries in the global catalog

Installed definition, table entry, or control block	Number of bytes per entry	Size in previous release if different
Installed ATOMSERVICE definition	660 bytes	
Installed BUNDLE definition **1**	300 bytes up to 800 bytes approximately	
Installed CONNECTION definition	528 bytes	440 bytes
Installed CORBASERVER definition	1304 bytes	1216 bytes
Installed DB2CONN definition	1548 bytes	1460 bytes
Installed DB2 ENTRY definition	332 bytes	244 bytes
Installed DB2TRAN definition	160 bytes	62 bytes
Installed DJAR definition	432 bytes	344 bytes
Installed DOCTEMPLATE definition	284 bytes	196 bytes
Installed ENQMODEL definition	152 bytes	64 bytes
Installed EVENTBINDING and CAPTURESPEC definitions **2**	4000 bytes approximately	
Installed extrapartition queue definition	392 bytes	296 bytes
Installed FILE definition	588 bytes	500 bytes
Installed indirect queue definition	180 bytes	92 bytes
Installed intrapartition queues definition	328 bytes	240 bytes
Installed IPCONN definition	402 bytes	312 bytes
Installed JOURNALMODEL definition	168 bytes	80 bytes
Installed JVM Program definition	168 bytes up to 307 bytes	80 bytes up to 219 bytes
Installed JVMSERVER definition	146 bytes	
Installed LIBRARY definition	852 bytes	764 bytes
Installed MQCONN definition	620 bytes	
Installed MQINI definition	212 bytes	
Installed PARTNER definition	148 bytes	124 bytes
Installed PIPELINE definition	1500 bytes	1412 bytes
Installed PROCESSTYPE definition	148 bytes	60 bytes
Installed PROFILE definition	158 bytes	70 bytes
Installed PROGRAM definition	168 bytes	44 bytes
Installed REQUESTMODEL definition	226 bytes	138 bytes
Installed remote queue definition	172 bytes	84 bytes
Installed TCPIPSERVICE definition	924 bytes	580 bytes
Installed model TERMINAL definitions **3**	634 bytes	610 bytes
Installed TRANCLASS definitions	124 bytes	36 bytes
Installed TRANSACTION definitions (without TPNAME)	244 bytes	140 bytes
Installed TRANSACTION definitions (with TPNAME or XTPNAME)	388 bytes	204 bytes
Installed TSMODEL definition	236 bytes	148 bytes
Installed TYPETERM definitions **3**	634 bytes	610 bytes
Installed VSAM file (or data table) definition	312 bytes	288 bytes
Installed URIMAP definition	1316 bytes	1220 bytes

Table 7. Sizes for entries in the global catalog (continued)

Installed definition, table entry, or control block	Number of bytes per entry	Size in previous release if different
Installed WEBSERVICE definition	1040 bytes	936 bytes
BDAM file control table entry (FCT)	284 bytes	146 bytes
BDAM data control blocks	156 bytes	132 bytes
VSAM LSR share control blocks [4]	1224 bytes	1184 bytes
Data set names (JCL or dynamically allocated) [5]	96 bytes	72 bytes
Data set name blocks	194 bytes	170 bytes
Eventprocess status	89 bytes	
File control recovery blocks [6]	149 bytes	125 bytes
Terminal control table entry (TCT)	1552 bytes approximately	1500 bytes approximately
Dump table entry	100 bytes	76 bytes
Interval control element (ICE)	120 bytes	96 bytes
Automatic initiator descriptor (AID)	120 bytes	96 bytes
Deferred work element (DWE) [7]	132 bytes	108 bytes
Installed journal	111 bytes	88 bytes
Recovery manager remote names	158 bytes	134 bytes
Transient data destination record	70 bytes	46 bytes
Transient data destination auxiliary record	58 bytes	34 bytes
Loader program definitions	68 bytes	44 bytes
Session TCTTEs	918 bytes	894 bytes
Log streams	112 bytes	88 bytes
Uri virtual hosts	180 bytes	156 bytes

Notes

1. The size of a bundle catalog record depends on the number of parts that are included in the bundle, the size of the bundle directory, and the length of the scope. The minimum size for a bundle with no scope and small directory is approximately 300 bytes. However, with a scope and large directory, the number of bytes can increase to 800. For each bundle part, the size can also range from 300 to 800 bytes.
2. The size of an event binding catalog record depends on the number of capture specifications in the event binding and the number of filters and capture data items in each capture specification. If required, you might have multiple catalog records for each event binding.
3. The 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 through a group list. For example, if you start CICS with the startup parameter GRPLIST=DFHLIST, the CICS-supplied TYPETERM and model terminal definitions, defined in the groups DFHTERM and DFHTYPE, are recorded in the global catalog. Allocate space in your calculations for all autoinstall resources installed in your CICS region.
4. One for each LSR pool; that is, 8.
5. If you open a VSAM path, you get two of these; for BDAM or VSAM base data sets you get one.
6. You will have these if you use the VSAM RLS SHCDS option NONRLSUPDATEPERMITTED. In this case, for each data set for which you have specified NONRLSUPDATEPERMITTED, you can have an upper limit. This limit is the number of different file names through which you access the

data set multiplied by the number of tasks that update the data set. You will typically have only a few, if any, of these control blocks.

7. The value given is for a DWE chained off an LU6.1 session or an APPC session.

Buffer space sizings:

Use the buffer space parameters to define buffers of a specific size in the VSAM definition for the global catalog data set.

You must ensure that the REGION parameter on your CICS jobs is high enough to cope with the increase in buffer storage used for the global catalog, because this storage comes out of region storage not EDSA.

By default, VSAM uses a buffer space equal to twice the control interval size of the data component and the control interval size of the index. In the sample job, this calculation gives a default of 69632 bytes. A larger minimum buffer size can improve cold start and warm restart times, and might significantly reduce CICS shutdown times.

You can use two parameters to control the buffer size. The **BUFFERSPACE** parameter defines the minimum buffer space size that is allowed. The **BUFSP** parameter defines the maximum buffer size. You can add either parameter to the sample job to set an appropriate buffer size.

For performance reasons, CICS defines a STRNO (number of strings) value of 32. Based on the sample job, the minimum value of BUFSP is calculated as follows:

```
BUFND = (STRNO + 1)
BUFNI = STRNO
BUFSP = 33 * 32768 (BUFND * CI size) + 32 * 1024  (BUFNI * CI size) =
        1114112 bytes
```

Another way to define buffer space for the global catalog data set is to use the **AMP** parameter on the DD statement in the CICS startup job stream, which you can use to override the default or defined value. If the **AMP BUFSP** specifies fewer bytes than the **BUFFERSPACE** parameter of the access method services DEFINE command, the **BUFFERSPACE** number overrides the **BUFSP** number.

Reusing the global catalog to perform a cold start:

If you need to perform a cold start, **do not** delete and redefine the global catalog data set.

If you were to delete and redefine the global catalog, CICS would perform an *initial* start, and all recovery information for remote systems would be lost. When remote systems reconnected, CICS would inform them that it had lost any information that they needed to resynchronize their units of work, and messages would be produced to record the fact, on both the local and the remote systems.

Instead, to specify that the next start should be cold, use the DFHRMUTL utility with the SET_AUTO_START=AUTOCOLD option. This has the following advantages:

- You do not have to reset the START system initialization parameter from AUTO to COLD, and back again.

- Because sufficient information is preserved on the global catalog and the system log, CICS is able to recover information for remote systems from the log, and to reply to remote systems in a way that enables them to resynchronize their units of work.

You can speed up a cold start by using the DFHRMUTL COLD_COPY option to copy only those records that are needed for the cold start to another catalog data set. If the return code set by DFHRMUTL indicates that the copy was successful, a subsequent job-step can copy the new (largely empty) catalog back to the original catalog data set. The performance gain occurs because, at startup, CICS does not have to spend time deleting all the definitional records from the catalog. This technique will also speed up initial starts, for the same reason. Figure 13 is an example of this technique.

Note: Before you use COLD_COPY, you should be certain that you want to perform a cold or initial start. As a safeguard, make a backup copy of the original global catalog before you copy the new catalog output by DFHRMUTL over it. For more information about the use of the global catalog in a cold start of CICS, see "Controlling start and restart" on page 114.

```
//RMUTL    EXEC PGM=DFHRMUTL,REGION=1M
//STEPLIB  DD DSNAME=CICSTS54.CICS.SDFHLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//DFHGCD   DD DSNAME=CICSTS54.CICS.applid.DFHGCD,DISP=OLD
//NEWGCD   DD DSNAME=CICSTS54.CICS.applid.COPY.DFHGCD,DISP=OLD
//SYSIN    DD *
SET_AUTO_START=AUTOCOLD,COLD_COPY
/*
//         IF (RMUTL.RC<16) THEN
//*  Steps to be performed if RMUTL was a success
//COPY     EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//DFHGCD   DD DSNAME=CICSTS54.CICS.applid.DFHGCD,DISP=OLD
//NEWGCD   DD DSNAME=CICSTS54.CICS.applid.COPY.DFHGCD,DISP=OLD
//SYSIN    DD *
   REPRO INFILE(NEWGCD) OUTFILE(DFHGCD) REUSE
/*
//*  End of steps to be performed if RMUTL was a success
//         ENDIF
```

Figure 13. DFHRMUTL example—setting the global catalog for a cold start. The COLD_COPY option is used to improve startup performance. Note that the NEWGCD and DFHGCD data sets must have been defined with the REUSE attribute.

Defining the local catalog

The local catalog is a VSAM key-sequenced data set (KSDS). The local catalog is not shared by any other CICS region.

Before you begin

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.

CICS Transaction Server for z/OS 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. All the domains 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 "Controlling start and restart" on page 114.

About this task

You can define and initialize the CICS local catalog in two ways. You can use the sample job as described below or you can use the CICS-supplied job, DFHDEFDS, to create a local catalog for an active CICS region.

Edit the sample job as follows:

Procedure

1. If you are defining local catalogs for multiple CICS regions, identify the clusters uniquely by making the specific APPLID of each CICS one of the data set qualifiers. For example, you might use the following names for the clusters of active and alternate CICS regions, where DBDCCIC1 and DBDCCIC2 are the specific APPLIDs:

   ```
   DEFINE CLUSTER -
          (NAME( CICSTS54.CICS.DBDCCIC1.DFHLCD)

   DEFINE CLUSTER -
          (NAME( CICSTS54.CICS.DBDCCIC2.DFHLCD)
   ```

2. Space for 200 records is probably adequate for the local catalog. However, specify space for secondary extents as a contingency allowance.
3. The local catalog records are small in comparison with the global catalog. Use the record sizes shown, which, with the number of records specified, ensure enough space for the data set.
4. Define the data definition statement for the local catalog as follows:

   ```
   //DFHLCD  DD  DSN=CICSTS54.CICS.applid.DFHLCD,DISP=OLD
   ```

Example

```
//LOCAT    JOB accounting info,,CLASS=A
//DEFLCD   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
         DEFINE CLUSTER -
                (NAME( CICSTS54.CICS.applid.DFHLCD) -  1
                 INDEXED                        -
                 RECORDS( 200 10 )              -                  2
                 FREESPACE(10 10)               -
                 SHAREOPTIONS( 2 )              -
                 REUSE                          -
                 VOLUMES( volid ))              -
           DATA                                 -
                (NAME( CICSTS54.CICS.applid.DFHLCD.DATA )   -
                 KEYS( 52 0 )                   -
                 RECORDSIZE( 70 2041 )          -                  3
                 CONTROLINTERVALSIZE( 2048 ))  -
           INDEX (NAME( CICSTS54.CICS.applid.DFHLCD.INDEX ) )
/*
//*****************************************************************
//INITLCD EXEC PGM=DFHCCUTL
//*
//*            INITIALIZE THE CICS LOCAL CATALOG
//*
//STEPLIB   DD DSN=CICSTS54.CICS.SDFHLOAD,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//DFHLCD    DD DSN=CICSTS54.CICS.applid.DFHLCD,DISP=SHR
//*
//
```

Figure 14. Sample job to define and initialize the local catalog

Initializing the local catalog:

Before the local catalog can be used to start a CICS region, you must initialize it with data from the domain manager parameter records and three loader domain parameter records.

About this task

The domain manager parameter records contain information relating to the CICS domains and are identified by their domain names. See Table 8 on page 59 for a complete list.

The three loader domain parameter records contain information relating to these resources:

- DFHDMP, the CSD file manager
- DFHEITSP, the RDO language definition table
- DFHPUP, the CSD parameter utility program

Procedure

1. To enable you to initialize the local catalog correctly, with all the records in the correct sequence, run the CICS-supplied utility DFHCCUTL immediately after you have defined the VSAM data set. This utility writes the necessary data to the local catalog from the CICS domains.

2. The loader domain also writes a program definition record for each CICS nucleus module. The number of records varies depending on the level of function that you have included in your CICS region. Allow for at least 75 of these loader-domain records.
3. Use the CICS-supplied utility DFHSMUTL to add records to the local catalog to enable the CICS self-tuning mechanism for storage manager domain subpools.

Example

Table 8. List of domain names

Domain name in catalog	Description
DFHAP	Application domain
DFHAS	Asynchronous services domain
DFHBA	Business application manager
DFHCC	CICS local catalog domain
DFHDD	Directory manager domain
DFHDH	Document handler domain
DFHDM	Domain manager domain
DFHDP	Debugging profiles domain
DFHDS	Dispatcher domain
DFHDU	Dump domain
DFHEJ	Enterprise Java™ domain
DFHEM	Event manager domain
DFHEP	Event processing domain
DFHGC	CICS global catalog domain
DFHIE	ECI over TCP/IP domain
DFHKE	Kernel domain
DFHLD	Loader domain
DFHLG	Log manager domain
DFHLM	Lock manager domain
DFHME	Message domain
DFHML	XML services domain
DFHMN	Monitoring domain
DFHNQ	Enqueue manager domain
DFHOT	Object transaction domain
DFHPA	System initialization parameter domain
DFHPG	Program manager domain
DFHPI	Pipeline manager domain
DFHPT	Partner management domain
DFHRL	Resource life-cycle domain
DFHRM	Recovery manager domain
DFHRS	Region status domain
DFHRX	RRMS domain

Table 8. List of domain names (continued)

Domain name in catalog	Description
DFHRZ	Request streams domain
DFHSH	Scheduler services domain
DFHSJ	JVM domain
DFHSM	Storage manager domain
DFHSO	Sockets domain
DFHST	Statistics domain
DFHTI	Timer domain
DFHTR	Trace domain
DFHTS	Temporary storage domain
DFHUS	User domain
DFHWB	Web domain
DFHW2	Web 2.0 domain
DFHXM	Transaction manager domain
DFHXS	Security domain

Domains that write to the local catalog:

Some domains 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:

- Dispatcher domain
- Dump domain
- Loader domain
- Message domain
- Parameter manager domain
- Storage manager domain
- Transient data

Setting up auxiliary trace data sets

CICS trace entries can be directed to auxiliary trace data sets, which are CICS-owned BSAM data sets. If you want to use auxiliary trace, you must create the data sets before you start CICS; you cannot define them while CICS is running.

About this task

You can choose from a number of destinations for trace data that is handled by the CICS trace domain. Any combination of these destinations can be active at any time:

- The internal trace table, in main storage in the CICS address space.
- The auxiliary trace data sets, defined as BSAM data sets on disk or tape.
- The z/OS generalized trace facility (GTF) data sets.
- The JVM server trace file in z/OS Unix System Services

Auxiliary trace has the following advantages:

- You can collect large amounts of trace data, if you initially define large enough trace data sets. For example, you might want to do this to trace system activity over a long period of time, perhaps to solve an unpredictable storage violation problem.
- You can use the auxiliary switch (**AUXTRSW** system initialization parameter) to specify that auxiliary trace data cannot be overwritten by subsequent trace data. With the internal trace table, when the end of the table is reached subsequent entries overwrite those at the start of the table, and diagnostic information is lost.
- Auxiliary trace can be particularly useful if you are using CICS trace during startup, because of the high volume of trace entries that are written when CICS is initializing.

Auxiliary trace is held in one or two sequential data sets, on either disk or tape. The DD names of the auxiliary trace data sets are defined by CICS as DFHAUXT and DFHBUXT. If you define a single data set only, its DD name must be DFHAUXT.

Trace entries are of variable length, but the physical record length (block size) of the data that is written to the auxiliary trace data sets is fixed at 4096 bytes. As a rough guide, each block contains an average of 40 trace entries, although the actual number of trace entries depends on the processing that is being performed.

Procedure

1. Decide whether to define one or two sequential data sets for auxiliary trace. If you want to specify automatic switching for your auxiliary trace data sets, you must define two data sets. If you specify automatic switching for auxiliary trace, but define only one data set, auxiliary trace is stopped and a system dump is generated.
2. Decide on the location of the auxiliary trace data sets. If you use tape for recording auxiliary trace output, use unlabeled tape. Using standard-labeled tape, whether on a single tape drive or on two tape drives, stops you processing the contents of any of the volumes with the DFHTU710 utility until after the CICS step has been completed. If you have to use standard-labeled tape, make sure all the output produced in the CICS run fits on the one (or two) volumes mounted. You cannot catalog data sets that are on unlabeled tapes.
3. If you are defining auxiliary trace data sets on disk, allocate and catalog the auxiliary trace data sets before you start CICS. Use one of the following methods:
   - Use the sample job shown in “Sample job to allocate auxiliary trace data sets” on page 62 to allocate and catalog the data sets.
   - Use the supplied job DFHDEFDS to create the auxiliary trace data sets. For more information, see Creating the CICS data sets in Installing.
4. If you are using tape for the auxiliary data sets, and you want auxiliary trace to be active from CICS startup, assign tape units and mount the tapes before you start CICS. If you plan to start auxiliary trace by entering a command when CICS is running, ensure the tapes are mounted before you enter the command.
5. Define the auxiliary trace data sets to CICS in the startup job stream.
   a. For auxiliary trace data sets on disk, use the following DD statements:

```
//DFHAUXT  DD  DSN=CICSTS54.CICS.applid.DFHAUXT,DCB=BUFNO=n,DISP=SHR
//DFHBUXT  DD  DSN=CICSTS54.CICS.applid.DFHBUXT,DCB=BUFNO=n,DISP=SHR
```

If you specify BUFNO greater than 1, you can reduce the I/O overhead involved in writing auxiliary trace records. A value between 4 and 10 can greatly reduce the I/O overhead when running with auxiliary trace on. DISP=SHR allows the simultaneous processing of a data set by the DFHTU710 offline utility program after a switch to the other data set has taken place.

b. For auxiliary trace data sets on unlabeled tapes, use the following sample DD statements:

```
//DFHAUXT  DD  DSN=CICSTS54.CICS.applid.DFHAUXT,UNIT=3400,VOL=SER=volid,
//         DISP=(NEW,KEEP),LABEL=(,NL)
//DFHBUXT  DD  DSN=CICSTS54.CICS.applid.DFHBUXT,UNIT=3400,VOL=SER=volid,
//         DISP=(NEW,KEEP),LABEL=(,NL)
```

Sample job to allocate auxiliary trace data sets

If you are defining auxiliary trace data sets on disk, you can use this sample job to allocate and catalog them before you run CICS.

Procedure

1. The DCB subparameters shown in this sample job specify the required DCB attributes for the CICS auxiliary trace data sets. As an alternative to this job, you can specify (NEW,CATLG) on the DD statements in the CICS startup job stream, omit the DCB parameter, and let CICS open the data sets with the same default values.
2. Change the space allocations in this sample job stream to suit your installation's needs.

Example

```
//DEFTRCDS JOB (accounting information),
//             MSGCLASS=A,MSGLEVEL=(1,1),
//             CLASS=A,NOTIFY=userid
//*******************************************************************
//*            Create auxiliary trace data sets
//*******************************************************************
//ALLOCDS  EXEC PGM=IEFBR14
//DFHAUXT  DD  DSN=CICSTS54.CICS.applid.DFHAUXT,UNIT=3380,VOL=SER=volid,
//         DISP=(NEW,CATLG),DCB=(BLKSIZE=4096,RECFM=F,LRECL=4096),  1
//              SPACE=(CYL,(25))                                   2
//DFHBUXT  DD  DSN=CICSTS54.CICS.applid.DFHBUXT,UNIT=3380,VOL=SER=volid,
//         DISP=(NEW,CATLG),DCB=(BLKSIZE=4096,RECFM=F,LRECL=4096),  1
//              SPACE=(CYL,(25))                                   2
//
```

Figure 15. Sample job to define auxiliary trace data sets on disk

Defining dump data sets

CICS uses the dump data sets for recording dumps as a consequence of a failure that is detected when CICS is running, or upon explicit request.

About this task

You must define two types of dump data sets:

1. MVS system dump data sets, for recording system dumps that CICS requests using the MVS SDUMP macro.
2. CICS transaction dump data sets, for recording transaction dumps.

Procedure

1. For the initial installation of CICS, define a dump data set of between 5 and 10 MB. When normal operations begin, you can adjust this to suit your own installation's requirements.
2. Change the SDUMP options using the MVS **CHNGDUMP** command. You must thoroughly review your use of the **CHNGDUMP** command when setting up your CICS region.
   a. Use the MERGE function to ensure that the areas selected by CICS to dump are included in the MVS dump data set output.
   b. Use the ADD option to replace any options specified by CICS when issuing the SDUMP. This can result in partial dumps being taken to the MVS dump data set.
3. Define the CICS transaction dump data sets, as described in "Defining the transaction dump data sets" on page 65.

Results

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.

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.

System dumps

CICS produces a system dump using the MVS SDUMP macro.

The MVS SDUMP dump results from CICS issuing an MVS SDUMP macro. It includes almost the entire CICS address space, that is, the MVS nucleus and other common areas, as well as the CICS private storage areas. The SDUMP dump is written to an MVS dump data set, which you can process using the interactive problem control system (IPCS), either online under TSO, or by submitting a batch job to print it. IPCS is described in the *z/OS MVS IPCS User's Guide*. For information about the IPCS **VERBEXIT** parameters that you use with the CICS IPCS dump exit, see Using IPCS to format and analyze CICS dumps: Overview. For information about the SDUMP macro, and the associated MVS dump data sets, see *z/OS MVS Diagnosis: Tools and Service Aids*.

The SDUMP macros issued by CICS normally contain the `QUIESCE=NO` parameter. They might not if the SDUMP is taken because of an abend in CICS SVC code or when altering MRO control blocks. This parameter allows the MVS system to remain dispatchable while the SDUMP is being taken, thus reducing the impact on the system. However if `QUIESCE=YES` is specified as an MVS system default it will override that specified by CICS. These defaults can be altered by using the **MVS CHNGDUMP** command.

Suppressing system dumps that precede ASRx abends

The MVS system dump data sets can become full with unwanted SDUMPs that precede ASRA, ASRB, and ASRD abends (after message DFHAP0001 or DFHSR0001).

If CICS storage protection is active, 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 a **CEMT SET SYDUMPCODE** command, or an **EXEC CICS SET SYSDUMPCODE** command to suppress system dumps for system dump code SR0001. For example:

```
CEMT SET SYDUMPCODE(SR0001) ADD NOSYSDUMP
```

CICS uses dump code SR0001 if an application program was executing in user-key at the time of the program check or MVS abend. This is possible only if storage protection is active. If the program was executing in CICS-key, dump code AP0001 is used instead.

When storage protection is not active, you can suppress system dumps for system dump code AP0001. However, this suppresses dumps for errors in both application and CICS programs. The XDUREQ global user exit can be used to distinguish between AP0001 situations in application and CICS programs.

For more information about the storage protection facilities available in CICS, see Storage protection.

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 taken for ASRA, ASRB, or ASRD abends. For example, to add an entry to the dump table and ensure that SDUMPs are taken for ASRB abends.specify the following:

```
CEMT SET TRDUMPCODE(ASRB) ADD SYSDUMP
```

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 CICS messages and Finding where a program check occurred.

The CICS transaction dump data sets

You can optionally specify DCB parameters for dump data sets if you want to copy the data sets to tape or disk. When CICS opens the dump data set, it issues an MVS DEVTYPE macro. This macro returns the track size for direct access devices, or 32760 for magnetic tape.

The maximum block size used for a transaction dump is the lesser of the values returned from the DEVTYPE macro and 4096. As this usually results in a block size of 4096 (because devices generally have a track size greater than this), CICS writes multiple blocks per track. After writing each block, MVS returns the amount of space remaining on the current track. If the space remaining is 256 bytes or more, then the size of the next block written is the lesser of the values returned by MVS and 4096.

If the space remaining is less than 256 bytes, the next block is written to the next track.

There are four global user exits that you can use with the 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

Printing the transaction dump data sets

With two data sets, you can print transaction dumps from one data set while CICS is running.

About this task

You can either print transaction dumps explicitly or use automatic switching.

Procedure

- To print transaction dumps explicitly:
  1. Use the command **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.
  2. Print the completed data set using the DFHDU710 dump utility program. For information about the DFHDU710 dump utility program, see Dump utilities (DFHDU680 and DFHPD680).
- To print transaction dumps using automatic switching:
  1. Use the command **CEMT SET DUMP AUTO** to cause automatic switching when the current data set becomes full. This command 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 data set closed.
     DFHDU0304I applid Transaction Dump Data set data set opened.
     DFHDU0305I applid Transaction Dump Data set switched to ddname.
     ```

  2. If you specified DISP=SHR for the dump data set, you can print the completed data set with the DFHDU710 utility program and then reissue the **CEMT SET DUMP AUTO** command. This again switches data sets automatically (once only) when the current data set is full.

Defining the transaction dump data sets

You must define the data sets in the CICS startup job stream with the DD names DFHDMPA and DFHDMPB. If you define a single data set only, its DD name must be DFHDMPA.

About this task

You can either define DFHDMPA and DFHDMPB as temporary data sets for each CICS run, or allocate and catalog the data sets in advance to reuse them repeatedly.

Procedure

1. 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.

2. Allocate the dump data sets. You can do this in one of two ways:
   - Use the sample data definition statements to allocate and catalog dump data sets on disk.
   - Use the CICS-supplied job DFHDEFDS to allocate and catalog the dump data sets.

   If you select to use the sample job, edit the statements as follows:
   a. Change the space allocations in this sample job stream to suit your own installation's needs.
   b. If you are running CICS with XRF, allocate different data sets for the alternate.
   c. If you use tape for recording dump output, use unlabeled tape. Standard-labeled tape, whether on a single tape drive or on two tape drives, stops you processing the contents of any of the volumes with the DFHDU710 utility until after the CICS step has been completed. If you want to use standard-labeled tape, make sure that all the output produced in the CICS run fits on the one or two volumes mounted.

      You cannot catalog dump data sets defined on unlabeled tapes. Your data set definitions must be in the CICS startup job stream each time CICS is run.

```
//DFHDMPA  DD  DSN=CICSTS54.CICS.applid.DFHDMPA,DISP=(NEW,CATLG),
//             UNIT=3380,VOL=SER=volid,SPACE=(CYL,(25))
//DFHDMPB  DD  DSN=CICSTS54.CICS.applid.DFHDMPB,DISP=(NEW,CATLG),
//             UNIT=3380,VOL=SER=volid,SPACE=(CYL,(25))
```

Figure 16. Sample job control statements for defining disk dump data sets

3. Optional: To copy dump data sets to tape or disk, specify DCB parameters on the DD statements when allocating and cataloging the dump data sets, as follows:

   ```
   //          DCB=(RECFM=VB,BLKSIZE=4096,LRECL=4092)
   ```

4. Include the following DD statements in the CICS startup job stream.
   - If you have cataloged the transaction dump data sets, add the following DD statement:

     ```
     //DFHDMPA  DD  DSN=CICSTS54.CICS.applid.DFHDMPA,DISP=SHR
     //DFHDMPB  DD  DSN=CICSTS54.CICS.applid.DFHDMPB,DISP=SHR
     ```

     DISP=SHR enables each data set, if held on disk, to be processed by the DFHDU710 offline utility after the switch to the other data set has taken place.
   - If you have put the transaction dump data sets on unlabeled tapes, add the following DD statement:

     ```
     //DFHDMPA  DD  DSN=CICSTS54.CICS.applid.DFHDMPA,UNIT=3400,VOL=SER=volid1,
     //         DISP=(NEW,KEEP),LABEL=(,NL)
     //DFHDMPB  DD  DSN=CICSTS54.CICS.applid.DFHDMPB,UNIT=3400,VOL=SER=volid2,
     //         DISP=(NEW,KEEP),LABEL=(,NL)
     ```

Results

CICS always attempts to open at least one transaction dump data set during initialization. If you do not include a DD statement for at least one transaction dump data set in your CICS job, initialization continues after the DFHDU0306 message is sent to the console.

When CICS opens the dump data set, it issues an MVS DEVTYPE macro. This returns the track size for direct access devices, or 32760 for magnetic tape. The maximum block size used for a transaction dump is the lesser of the values returned from the DEVTYPE macro and 4096. As this usually results in a block size of 4096 (because devices generally have a track size greater than this), CICS writes multiple blocks per track. After writing each block, MVS returns the amount of space remaining on the current track. If the space remaining is 256 bytes or more, then the size of the next block written is the lesser of the values returned by MVS and 4096.

If the space remaining is less than 256 bytes, the next block is written to the next track.

There are four global user exits that you can use with the 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

Defining user files

This section tells you how to define user files and how to access VSAM data sets, BDAM data sets, data tables, and coupling facility data tables.

About this task

CICS application programs process files, which, to CICS, are logical views of a physical data set or data table. For data tables, the file provides a view of the data table, which resides either in data space storage or in a coupling facility structure. Except in the case of coupling facility data tables, for which an underlying physical data set is optional, a data table is also associated with a source data set from which the table is loaded. For non-data-table files, the file provides a view of the data set.

A file is identified to CICS by a **file name** of up to eight characters, and there can be many files defined to CICS that refer to the same physical data set or data table. This has the following effect, depending on the type of object the file is defining:

- For non data table files, if more than one file refers to the same data set, each file refers to the same physical data.
- For user-maintained data tables, if more than one file refers to the same data set, each file represents a view of a unique data table.
- For CICS-maintained data tables, if more than one file refers to the same data set, only one can be defined as a CMT. The other files access data from the CMT created by the CMT file definition.

- For coupling facility data tables, if more than one file refers to the same data set, each file represents a view of a unique coupling facility data table in a CFDT pool (unless each file specifies the same tablename and poolname, in which case each they provide a separate view of the same table.

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 BDAM data. Any data sets referred to by CICS files must be created and cataloged, so that they are known to MVS 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-maintained or user-maintained data tables to improve the performance and function of CICS regions using files that refer to VSAM data sets. Data tables offer a method of constructing, maintaining, and gaining rapid access to data records contained in tables held in data space storage, above 16MB. Each data table is associated with a VSAM KSDS, known as its **source data set**. For more information about data tables, see "Defining data sets with multiple extents and volumes" on page 25.

You can use coupling facility data tables to share data across a sysplex, using the CICS file control API, subject to some restrictions, such as a 16 byte key length.

You can use RLS access mode to share VSAM data sets between CICS application-owning regions throughout a sysplex. See "VSAM record-level sharing (RLS)" on page 70 for further information.

Each of the above methods is discussed under the following topics:

- "VSAM data sets"
- "BDAM data sets" on page 73
- "Defining data sets to CICS" on page 74
- "Opening VSAM or BDAM files" on page 76
- "Closing VSAM or BDAM files" on page 77
- "CICS data tables" on page 78
- "Coupling facility data tables" on page 169.

VSAM data sets

You create a VSAM data set by running the Access Methods Services (AMS) utility program IDCAMS in a batch job, or by using the TSO DEFINE command in a TSO session.

About this task

The DEFINE command specifies to VSAM and MVS 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 AMS 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 69.

When you create a data set, you can define a data set name of up to 44 characters. If you choose not to define a name, VSAM assigns the name for you. This name, known as the data set name (or DSNAME), uniquely identifies the data set to your MVS system.

You can define VSAM data sets accessed by user files under CICS file control as eligible to be backed up while CICS is currently updating these data sets. For more information about backing up VSAM files open for update, see "Defining backup while open (BWO) for VSAM files" on page 28.

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 through 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 through 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:

There are two ways you can load data into an empty VSAM data set.

An empty data set can be loaded using either of the following methods:

- Running the AMS utility program, IDCAMS
- Writing records to the data set using CICS transactions

Although VSAM imposes some restrictions during initial data set load processing, when the data set is said to be in **load mode**, these do not affect CICS transactions. For files opened in non-RLS mode, CICS file control "hides" load mode processing from your application programs. For files opened in RLS mode against an empty data set, load mode processing is hidden from CICS by VSAM, and all VSAM requests are allowed.

Using IDCAMS:

If you have a large amount of data to load into a new data set, run the AMS 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.

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 AMS, 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 by 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 that 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.

Reuse of data sets:

If you define a data set with the AMS REUSE attribute, it can also be emptied of data during a CICS run.

This allows it to be used as a work file. When the status of a file referencing the data set is CLOSED and DISABLED (or UNENABLED), you can use the SET EMPTY command, either from an application program using the EXEC CICS command-level interface, or from a master terminal using the master terminal CEMT command. This command sets an indicator in the installed file definition 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 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

VSAM record-level sharing (RLS):

Record-level sharing (RLS) is an access mode for VSAM data sets supported by DFSMS 1.3 and later releases. RLS enables VSAM data to be shared, with full update capability, between many applications running in many CICS regions.

With RLS, CICS regions that share VSAM data sets can reside in one or more MVS images within an MVS parallel sysplex. This concept, in a parallel sysplex with

VSAM RLS supporting a CICSplex, is illustrated in Figure 17.

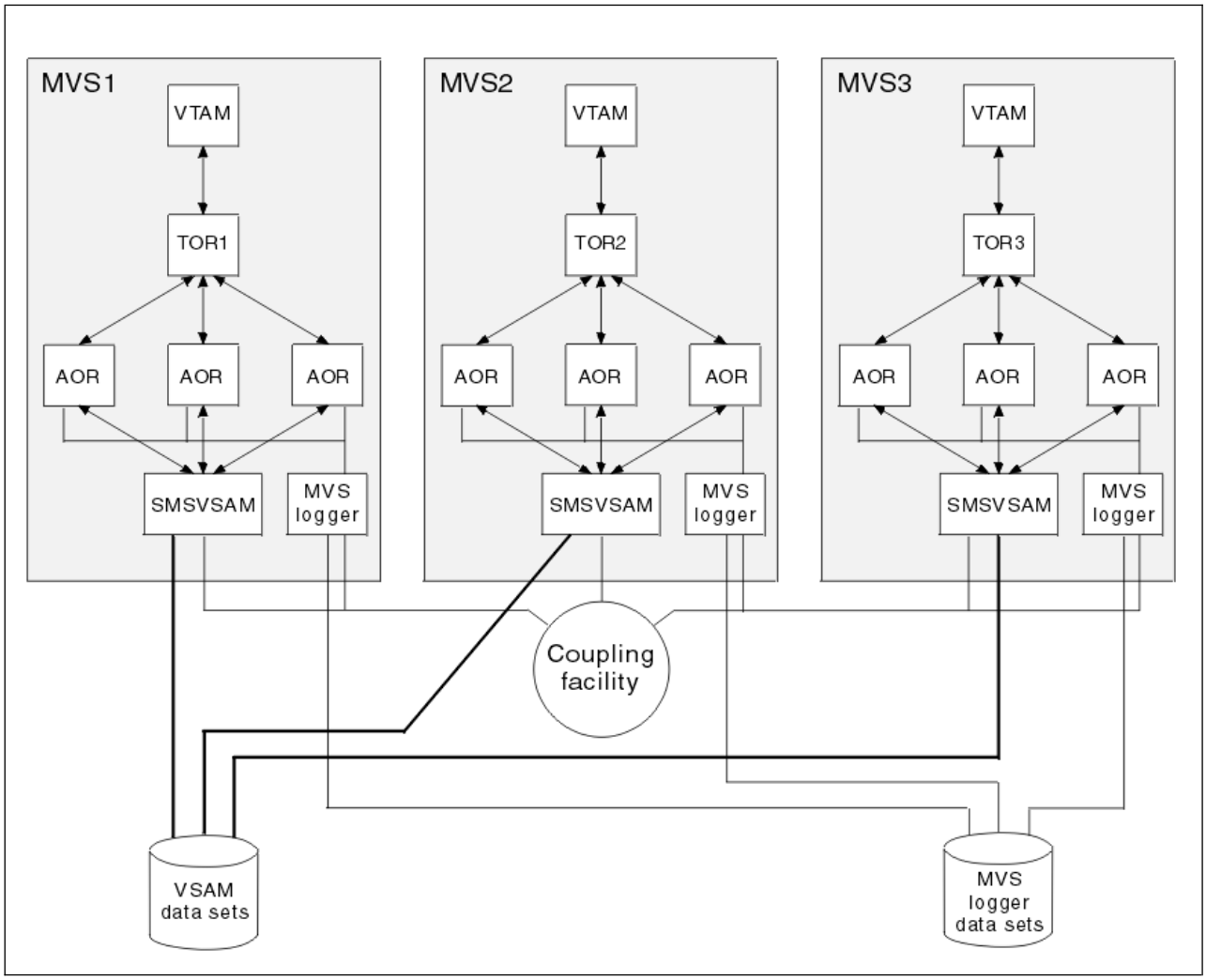


Figure 17. Diagram illustrating a Parallel Sysplex with RLS. This view of RLS shows multiple CICS regions using VSAM RLS, through the services of a SMSVSAM server in each MVS image.

Without RLS support (RLS=NO system initialization parameter), more than one CICS region cannot open the same VSAM data set concurrently using a non-RLS mode (such as LSR or NSR). These access modes mean that to share VSAM data between CICS regions, you must either:

- Use shared data tables,

  or
- Allocate the VSAM data sets to one CICS region, a file-owning region (FOR), and function ship file requests from the applications to the FOR using either MRO, APPC, or IPIC connections.

With RLS support, multiple CICS regions can open the same data set concurrently. To use RLS:

- You need a level of DFSMS that supports RLS, and RLS=YES specified as a CICS system initialization parameter
- The CICS regions must all run in the same parallel sysplex
- There must be one SMSVSAM server started in each MVS image

- Specify RLSACCESS(YES) in the CICS file resource definition to provide full update capability for data sets accessed by multiple CICS regions.

You can specify RLS access for all files supported by CICS file control, except for the following:

- Key range data sets are not supported.
- VSAM clusters defined with the IMBED attribute are not supported. However, you can remove the IMBED attribute from the cluster definition without loss of function. Use the access method services REPRO function to move the data into a new cluster defined without the IMBED attribute. You can then use RLS access mode for files that reference the new cluster. (IMBED is a performance option that is generally unnecessary with modern caching disk controllers.)
- Opening individual components of a VSAM cluster (which is not supported by CICS for any mode of access).
- Temporary data sets are not supported.
- Key-sequence data sets (KSDS) in relative byte address (RBA) mode (OPTCDE=ADR) are not supported. Application programs that specify the RBA keyword on file control API commands for a KSDS opened RLS mode receive an INVREQ with RESP2 51 exception condition.
- Direct open of alternate index data is not supported in RLS access mode. However, path access to data is supported.
- VSAM catalogs and VVDS data sets are not supported.

Although you can specify RLS access for entry-sequenced data sets (ESDS), it is not recommended, because it can have a negative effect on the performance and availability of the data set when you are adding records. For more information, see VSAM and file control: improving performance.

For details of all the steps necessary to set up support for VSAM RLS, see Setting up VSAM RLS support.

Mixed-mode operation for VSAM data sets:

Generally, you choose which data sets need to be shared and updated in RLS mode by multiple CICS regions. When you have made this choice, you are recommended always to update these data sets in RLS mode.

However, with RLS support, data sets can be shared in mixed access mode, between CICS regions and batch jobs. Mixed access mode means that a data set is open in RLS mode and a non-RLS mode concurrently by different users.

Although data sets can be open in different modes at different times, all the data sets within a VSAM sphere normally should be opened in the same mode. (A sphere is the collection of all the components—the base, index, any alternate indexes and alternate index paths—associated with a given VSAM base data set.) However, VSAM does permit mixed-mode operations on a sphere by different applications, subject to some CICS restrictions. In the following discussion about mixed-mode operation, references to a data set refer to any component of the sphere.

SMSVSAM operation of mixed mode:

SMSVSAM permits a data set to be opened in different modes concurrently, by different applications in a sysplex.

There are some sharing rules and limitations:

- A data set that is open in RLS mode to a number of CICS regions can also be opened in non-RLS mode for **read-only** operations
- Read-integrity is not guaranteed for the non-RLS application
- A data set to be opened concurrently in RLS and non-RLS mode must be defined with cross-region SHAREOPTIONS(2)

CICS restrictions:

You can open a file in RLS mode or non-RLS mode in a CICS region when the referenced data set is already open in a different mode by another user (CICS region or batch job).

However, in addition to the above VSAM rules, a data set cannot be open in different modes concurrently within the same CICS region. This ensures that CICS maintains a consistent view of data within the CICS region.

The CICS restrictions operate as follows:

- If a data set is opened in RLS mode in a CICS region, it cannot be opened, through another file, in non-RLS mode in the same CICS region.

  A non-RLS mode file open request fails with message DFHFC0512 if CICS already has the data set open in RLS mode.
- If a data set is opened in non-RLS mode in a CICS region, it cannot be opened, through another file, in RLS mode in the same CICS region.

  An RLS mode file open request fails with message DFHFC0511 if CICS already has the data set open in non-RLS mode.
- If a CICS region has unresolved recovery work for a data set it cannot be opened, through another file, in non-RLS mode in the same CICS region.

  A non-RLS mode file open request fails with message DFHFC0513 if CICS has outstanding recovery work for the data set.

BDAM data sets

CICS supports access to keyed and nonkeyed BDAM data sets. To construct and format such data sets, you use BDAM.

About this task

A BDAM data set must contain data before it is used in a CICS run. You load the data set using a batch program that writes the records sequentially. An example of this is Figure 18.

Notes:

```
//BDAM EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN    DD DUMMY
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD DSN=CICSTS54.bdam.user.file.init,DISP=SHR        1
//SYSUT2   DD DSN=CICSTS54.bdam.user.file,DISP=(,CATLG),       2
//         SPACE=(TRK,(1,1)),UNIT=3380,VOL=SER=volid,
//         DCB=(RECFM=F,LRECL=80,BLKSIZE=80,DSORG=DA)          3
```

Figure 18. Sample JCL to create and load a BDAM data set

1. The input data set (called SYSUT1 in this example) should be physically sequential and have attributes that are compatible with the DCB for the output data set (called SYSUT2 in this example; see note 3). In particular:

- If RECFM=F is specified for the output data set, then the input data set must have RECFM=F or RECFM=FB specified, and the value of LRECL should be the same for both data sets.
- If RECFM=V or RECFM=U is specified for the output data set, then the value of LRECL for the input data set must not be greater than that specified on the output data set.

2. When you create a data set, you define a data set name (DSNAME) of up to 44 characters. This data set name uniquely identifies the data set to your MVS system.
3. The DCB parameter for the output data set should specify the following:
   - DSORG=DA. This identifies the data set as a BDAM data set.
   - BLKSIZE. This should have the same value as specified for BLKSIZE in the associated file control table (FCT) entry.
   - RECFM. This can take the values F (fixed), V (variable), or U (undefined), and correspond to the first subparameter of the RECFORM operand in the associated FCT entry.

These options are specified on the DFHFCT TYPE=FILE definition. gives information about defining files using DFHFCT TYPE=FILE options. A data set created by this example, and loaded with data such as that shown in Figure 19, would have the following attributes specified in its FCT entry:

- BLKSIZE=80
- LRECL=40
- RECFORM=(FIXED BLOCKED)
- KEYLEN=8

```
RECORD 1 DATA FOR RECORD 1     RECORD 2 DATA FOR RECORD 2
RECORD 3 DATA FOR RECORD 3     RECORD 4 DATA FOR RECORD 4

RECORD98 DATA FOR RECORD 98    RECORD99 DATA FOR RECORD 99
1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
```

Figure 19. Sample data for loading a BDAM data set

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. The definition can be installed either from the CSD or from a file control table (FCT).

You can define VSAM files **only** through the CSD, and BDAM files **only** through the FCT. Definitions other than for BDAM in the FCT will not be installed, but they may remain in the FCT.

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 in one of the following ways:

- Using JCL in the startup job stream
- Using the DSNAME and DISP parameters of the FILE resource definitions
- Using dynamic allocation with CEMT
- Using dynamic allocation with an application program

The number of VSAM files that can be allocated to a CICS address space is about 10000. However, VSAM maintains a table entry for each file opened, and table space limits the number of files opened to about 8189.

Note: CICS will only deallocate files which were initially allocated by CICS. If a file was allocated by a third-party application, using a CEMT CLOSE command will close the file, but not deallocate it.

Using JCL

You can define the data set in a DD statement in the JCL of the CICS startup job. The DD name must be the same as the file name that refers to the data set.

For example, the following DD statements would correspond to file definitions for the file names VSAM1A and BDAMFILE:

```
//VSAM1A    DD   DSN=CICSTS54.CICS.vsam.user.file,DISP=OLD
//BDAMFILE  DD   DSN=CICSTS54.CICS.bdam.user.file,DISP=SHR
```

If you define a data set to CICS in this way it is allocated to CICS by MVS at CICS startup time, and it normally remains allocated until the end of the CICS run. Also, the physical data set is associated with the installed file definition throughout the CICS run.

If you use JCL to define a user data set to the CICS system, the DD statement must not include the FREE=CLOSE operand.

If you use the RLS=CR or RLS=NRI option on your DD statement, it will be ignored. The access mode for the file (RLS or non-RLS) and any read integrity options must be specified in the file definition.

Using the DSNAME and DISP file resource definition parameters:

You can define a data set to CICS by specifying the **DSNAME** and **DISP** parameters when you define the file.

You can specify these parameters either using RDO for files, or by using DFHFCT macros (BDAM files only). If you want to use DSNAME and DISP from the file definition, do **not** provide a DD statement for the data set in the startup job stream, because the attributes in the DD statement will override those in the CICS resource definition.

If you use DSNAME and DISP on the file definition, CICS allocates the data set dynamically at the time the first file referencing that data set is opened; that is, immediately before the file is opened. At this stage, CICS associates the file name with the data set.

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, it is automatically deallocated by CICS when the file is closed.

For information about using the **DSNAME** and **DISP** parameters, see System recovery table (SRT) and File control table (FCT).

Dynamic allocation using CEMT:

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) SHARE|OLD
```

When you use this command, CICS allocates the data set as part of OPEN processing as described above. The data set is automatically deallocated 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.

This method of defining the data set to CICS allows a file definition to be associated with different data sets at different times. Alternatively, you can close the file and deallocate the data set and then reallocate and open the same file with a different DISP setting. For example, you could do this to enable the physical data set to be shared with a batch program, which reads the data set.

For information about the CEMT SET command, see CEMT SET commands.

Other forms of dynamic allocation:

You are recommended to use only those methods of dynamic allocation that are part of CICS file control, and are described in the previous sections.

Do **not** use the CICS dynamic allocation transaction, ADYN, which invokes the sample CICS utility program, DFH99, for dynamic allocation of VSAM and BDAM **user files**. Use of the ADYN transaction may conflict with the dynamic allocation methods used within CICS file control, and can give unpredictable results.

Restrict the use of the ADYN transaction to those data sets not managed by CICS file control, such as auxiliary trace and CICS transaction dump data sets.

Opening VSAM or BDAM 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.

About this task

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. If you defined the file to be opened in RLS access mode, VSAM allocates control blocks and buffers in the SMSVSAM address space and associated data set. RLS mode also uses a CF cache structure, to which the data set is bound when the first file that references it is opened.

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 RDO OPENTIME(STARTUP) attribute (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 BDAM files

You can close files with a CLOSE command, with or without the FORCE option.

About this task

Closing files normally:

You can close a file explicitly using the **SET FILE(filename) CLOSED** command.

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 BDAM 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 requests:

```
READ UPDATE ---- REWRITE
STARTBROWSE ---- READNEXT ... ---- ENDBROWSE
```

A transaction is also a user of a file if it completes a recoverable change to the file but has not yet reached a sync point 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. If a transaction has made recoverable changes to a file and then suffered a failure during syncpoint, the unit of work is shunted, and the file can be closed at this point.

Closing files using the FORCE option:

You can close a file using the FORCE option on the **SET FILE(filename)** command.

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.

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.

CICS data tables

You can define a data table by using the **CEDA DEFINE FILE** command.

About this task

When a table is opened, CICS builds it by extracting data from the tables corresponding source VSAM data set and loading it into an MVS data space owned by the CICS data tables server region, and constructing an index in CICS virtual storage above the 16 MB line. The commands used to access these tables are the file control commands of the CICS application programming interface (API).

For programming information about the file control commands of the application programming interface, see the CICS command summary. CICS supports two types of data table:

- **CICS-maintained data tables** that CICS keeps in synchronization with their source data sets.
- **User-maintained data tables** that are 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 Global user exit programs. For further information about CICS data tables, see Shared data tables overview.

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 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 BDAM files" on page 76.

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 Shared data tables overview 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 space 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 BDAM files" on page 77.

Defining the CDBM GROUP command data set

The CDBM GROUP command data set, DFHDBFK, is a VSAM key-sequenced data set (KSDS). The CDBM transaction uses this data set to provide a repository for stored groups of DBCTL commands, .

About this task

Procedure

1. Create the DFHDBFK data set by running an IDCAMS job, an example of which is shown in Figure 20 on page 80.

```
//DBFKJOB  JOB  'accounting information',name,MSGCLASS=A
//*
//DBFKDEF  EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD SYSOUT=*
//AMSDUMP  DD SYSOUT=*
//SYSIN    DD *
  DELETE CICSTS54.CICS.DFHDBFK
  SET MAXCC=0
  DEFINE CLUSTER (                          -
                NAME( CICSTS54.CICS.DFHDBFK )  -
                INDEXED                     -
                RECORDS(100 20)             -
                KEYS(22,0)                  -
                RECORDSIZE(1428 1428)       -
                )                           -
  INDEX         (                           -
                NAME( CICSTS54.CICS.DFHDBFK.INDEX )     -
                CONTROLINTERVALSIZE(512)    -
                )                           -
  DATA          (                           -
                NAME( CICSTS54.CICS.DFHDBFK.DATA )      -
                CONTROLINTERVALSIZE(2048)   -
                )
/*
//*     The next two job steps are optional.
//*
//DBFKINID EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DELETE  CICSTS54.CICS.DBFKINIT
/*
//DBFKINIF EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT2   DD   DSN=CICSTS54.CICS.DBFKINIT,DISP=(NEW,CATLG),
// UNIT=dbfkunit,VOL=SER=dbfkvol,SPACE=(TRK,(1,1)),
// DCB=(RECFM=FB,LRECL=40,BLKSIZE=6160)
//*  Place the definitions you want to load after SYSUT1. For example:
//SYSUT1   DD *
SAMPLE      DIS       DB DI21PART
SAMPLE      STA       DB DI21PART
SAMPLE      STO       DB DI21PART
/*
//SYSIN    DD *
 GENERATE MAXFLDS=1
 RECORD FIELD=(40)
/*
//DBFKLOAD EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD SYSOUT=*
//AMSDUMP  DD SYSOUT=*
//SYS01    DD DSN=CICSTS54.CICS.DBFKINIT,DISP=SHR
//DFHDBFK  DD DSN=CICSTS54.CICS.DFHDBKF,DISP=SHR
//SYSIN    DD *
  REPRO INFILE (SYS01)                    -
        OUTFILE (DFHDBFK)
/*
//
```

where *dbfkvol* is the volume on which the DFHDBFK data set is to be created and *dbfkunit* is the unit type for that volume.

Figure 20. Sample job to define and initialize the DFHDBFK data set

2. Use this job to load IMS™ commands or use the maintenance function within the CDBM transaction.

Job control statements for CICS execution

About this task

If you define the DFHDBFK data set using the sample JCL shown in Figure 20 on page 80, the data definition statement for the CICS execution is as follows:

```
//DFHDBFK  DD DSN=CICSTS54.CICS.DFHDBFK,DISP=SHR
```

Alternatively, if you want to use dynamic file allocation, add the fully-qualified data set name to the DFHDBFK file resource definition.

Record layout in the CDBM GROUP command file

Each record in the DFHDBFK file may be up to 1428 characters long, as follows:

About this task

field	length	content	description
1	12	Group	a 12-character field containing your chosen name for this group. The acceptable characters are A-Z 0-9 $ @ and #. Leading or embedded blanks are not allowed, but trailing blanks are acceptable.
2	10	IMS Command	a 10-character field containing any of the IMS command verbs that are valid for CDBM (see for details). Leading or embedded blanks are not allowed, but trailing blanks are acceptable. **Note:** The validity of the IMS command verb is not checked by CDBM. Invalid values will be reported by IMS when the command is attempted.
3	1406	IMS Command parameters	Up to 1406 characters of parameters appropriate to the chosen IMS command verb. (This will often consist of lists of databases.) **Note:** Wildcard characters may not be used in the parameters stored in the CDBM Group command file. This is unlike the other functions of the CDBM transaction which permit the use of wildcard characters to describe multiple similarly named databases.

Defining the CMAC messages data set

The CMAC messages data set is a VSAM key-sequenced data set (KSDS) called DFHCMACD. The CMAC transaction uses DFHCMACD to provide online descriptions of the CICS messages and codes.

About this task

To use the CICS messages facility in your CICS region, you must create your own CSD group lists to include the DFHCMAC group for the CICS messages facility and any other groups of resources that your CICS region requires. Specify your new group lists on the **GRPLIST** system initialization parameter when you start up your CICS region.

Specify the DFHCMAC group of resources for the CICS messages facility only in those CICS regions that use the facility; for example, on terminal-owning regions, but perhaps not on data-owning regions.

Procedure

1. Create the DFHCMACD data set and load it with the CICS-supplied messages and codes data. You can create the data set in one of the following ways:
   - Run the DFHCMACI job. For more information about the DFHCMACI job, see Creating the CICS data sets in Installing.
   - Use the supplied sample job, as described in "Job control statements to define and load the messages data set."
2. If you use the supplied sample job to define the messages data set, add the following data definition statement to your CICS startup JCL:

   ```
   //DFHCMACD  DD DSN=CICSTS54.CICS.DFHCMACD,DISP=SHR
   ```
3. Create a CICS resource definition for the CMAC file. The supplied definition for the CMAC file and other resources required by the CICS messages facility are in the CSD group DFHCMAC. The CICS IVPs have a DD statement for the CMAC file, but for dynamic allocation you must copy the supplied resource definition for the CMAC file and add the DSNAME option.
4. Create your own CSD group list to include the supplied group list DFHLIST, the DFHCMAC group for the CICS messages facility, and any other groups of resources that your CICS region needs. Specify this group list in the **GRPLIST** system initialization parameter when you start your CICS region.

Results

The CICS messages are available from the CMAC transaction.

Job control statements to define and load the messages data set

Before its first use, the DFHCMACD data set should be defined and loaded as a VSAM key sequenced data set (KSDS). The following sample job shows you how to do this.

About this task

Note that `cmacvol` is the volume on which the DFHCMACD data set is to be created.

```
//CMACJOB  JOB  'accounting information',name,MSGCLASS=A
//CMACDEF  EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD SYSOUT=*
//AMSDUMP  DD SYSOUT=*
//SYSIN    DD *
  DELETE CICSTS54.CICS.DFHCMACD
  SET MAXCC=0
  DEFINE CLUSTER (                              -
                NAME( CICSTS54.CICS.DFHCMACD )      -
                CYL(2,1)                        -
                KEYS( 9 0 )                     -
                INDEXED                         -
                VOLUME ( cmacvol)               -
                RECORDSIZE( 8192 30646 )        -
                FREESPACE( 5 5 )                -
                SHAREOPTIONS( 2 )               -
                )                               -
  INDEX         (                               -
                NAME( CICSTS54.CICS.DFHCMACD.INDEX )    -
                )                               -
  DATA          (                               -
                NAME( CICSTS54.CICS.DFHCMACD.DATA )     -
                )
/*
//CMACLOAD EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD SYSOUT=*
//AMSDUMP  DD SYSOUT=*
//SYS01    DD DSN=CICSTS54.CICS.SDFHMSGS(DFHCMACD),DISP=SHR
//DFHCMACD DD DSN=CICSTS54.CICS.DFHCMACD,DISP=SHR
//SYSIN    DD *
  REPRO INFILE (SYS01)                      -
        OUTFILE (DFHCMACD)
/*
//
```

Figure 21. Sample job to define and initialize the CMAC data set

Defining the WS-AT data set

The data set you require to enable CICS Web Services Atomic Transaction (WS-AT) support is the WS-AT directory data set, DFHPIDIR. The WS-AT directory data set, DFHPIDIR, is a file containing a mapping between contexts and tasks.

The WS-AT directory and object store data set, DFHPIDIR, is a CICS file-control-managed data set that requires a resource definition in the CSD. Because the data set is shared across CICS regions that together provide a WS-AT capable web service provider, you can define it to CICS file control as one of the following formats:

- An ordinary VSAM data set, to be opened in either LSR or NSR mode; that is, with the LSRPOOLNUM attribute specified with a pool number, or as NONE. In this case, the data set has to be owned by one CICS region (a file-owning region) to which the other regions can function ship file requests.

  A definition for the data set is supplied in the CICS CSD group DFHPIVS, with the LSRPOOLNUM default value of 1. Make a copy of this CSD group, rename it, and edit the file definitions to add details such as the data set name for dynamic allocation, a specific LSRPOOLNUM to match an explicit LSRPOOL resource definition, or the remote system attributes.

- A VSAM data set to be opened in RLS mode. A definition for the data set is supplied in the CICS CSD group DFHPIVR with RLSACCESS(YES) specified. Make a copy of this CSD group, rename it, and edit the file definitions to add attributes such as the data set name for dynamic allocation.
- A coupling facility data table. A definition for the data set is supplied in the CICS CSD group DFHPICF. Make a copy of this CSD group, rename it, and edit the file definitions to modify the CFDT details, such as the pool name and the table name.

Figure 22 shows the statements used to define DFHPIDIR.

```
 DEFINE CLUSTER(NAME(@dsindex@.CICS@regname@.DFHPIDIR)-
  INDEXED-
  LOG(UNDO)-                                  1
  CYL(2 1)-
  VOLUME(@dsvol@)-                            2
  RECORDSIZE( 1017 1017 )-                    3
  KEYS( 16 0 )-
  FREESPACE ( 10 10 )-
  SHAREOPTIONS( 2 3 ))-
DATA  (NAME(@dsindex@.CICS@regname@.DFHPIDIR.DATA) -
  CONTROLINTERVALSIZE(1024)) -
INDEX (NAME(@dsindex@.CICS@regname@.DFHPIDIR.INDEX))
```

Figure 22. Example JCL to define the WS-AT directory data set

1. Define the backout recovery attribute in the ICF catalog, so that the data set defined by this job can be used in either RLS or non-RLS mode. For data sets used in RLS mode, you define the recovery attributes in the ICF catalog, and those attributes override any that are specified in the CICS file resource definition. You must define the PI directory data set as recoverable.

2. Specify your own value for the VOLUME parameter, or remove it completely if you are using SMS-managed storage.

3. The default record size is 1 KB, which can be changed.

Note: You do not change any of the definition values shown in Figure 22. DFHPIDIR contains only one record, its control record. However, ensure that you specify runtime settings (such as BUFND, BUFNI, and STRNO subparameters on the AMP parameter on the DD statement, or the equivalent in the CICS file resource definition) to support the maximum number of requests that might be active at any one time.

Setting up the debugging profiles data sets

Application programmers who use certain debugging tools with CICS create debugging profiles which are stored in a VSAM key sequenced data set with an alternative index.

To set up the debugging profiles data sets:

1. Use the IDCAMS utility to create and initialize the VSAM data sets.
2. Create file definitions for the data sets. The data sets can be shared by more than one CICS region, and can be defined as:
   VSAM RLS files
   Define VSAM RLS files when you want to share profiles between several CICS regions in the same sysplex

VSAM non-RLS files
Define VSAM non-RLS files when you do not need to share profiles between regions in the same sysplex

Remote files
Define remote files when you want to use profiles that are stored in a region that is connected using MRO or ISC.

Creating the debugging profiles data sets

You can create the debugging profiles data sets in one of two ways. You can either use an IDCAMS utility to create and initialize the data sets or you can run the CICS-supplied job, DFHDEFDS, to create the data sets for a CICS region

Use the IDCAMS utility to create and initialize the following VSAM data sets:

DFHDPFMB
The debugging profiles base data set.

DFHDPFMP
The debugging profiles path data set.

DFHDPFMX
The debugging profiles alternate index data set.

Use the JCL in Figure 23 on page 86.

```
//DPFM  JOB  'accounting information',name,MSGCLASS=A
//DEFINE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN  DD  *
    DELETE CICSTS54.CICS.DFHDPFMB

    DEFINE CLUSTER (RECORDS(1000)-
     NAME (CICSTS54.CICS.DFHDPFMB) -
     SHAREOPTIONS(2 3) -
     LOG(NONE) -
     VOLUME (&DSVOL)  -
     IXD)             -
    DATA -
     (RECSZ(2560,2560) -
     CONTROLINTERVALSIZE(3072) -
     NAME (CICSTS54.CICS.DFHDPFMB.DATA) -
     KEYS(17 1) -
     FREESPACE(10 10) -
     BUFFERSPACE (8192)) -
   INDEX -
     (NAME(CICSTS54.CICS.DFHDPFMB.INDX))
//INITDP EXEC PGM=IDCAMS,REGION=512K
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
     REPRO INFILE ( SYS01 ) -
           OUTDATASET(CICSTS54.CICS.DFHDPFMB)
//SYS01     DD *
 DDUMMY   RECORD                      !! DO NOT ALTER !!
 EEXAMPLE RECORD   REMOVE THIS LINE IF SAMPLES NOT REQUIRED
/*
//DEFALT   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN  DD  *
    DEFINE ALTERNATEINDEX -
     ( NAME(CICSTS54.CICS.DFHDPFMX ) -
     RECORDS(1000) -
     VOLUME(&DSVOL) -
     KEYS(12 20) -
     RELATE(CICSTS54.CICS.DFHDPFMB) -
     RECORDSIZE(200 200) -
     SHAREOPTIONS(2 3) -
     UPGRADE ) -
    DATA -
     ( NAME(CICSTS54.CICS.DFHDPFMX.DATA) ) -
    INDEX -
     ( NAME(CICSTS54.CICS.DFHDPFMX.INDEX) )
    DEFINE PATH -
     ( NAME(CICSTS54.CICS.DFHDPFMP) -
     PATHENTRY(CICSTS54.CICS.DFHDPFMX) )
/*
//BLDDP EXEC PGM=IDCAMS
//BDSET1 DD DSN=CICSTS54.CICS.DFHDPFMB,DISP=SHR
//ADSET1 DD DSN=CICSTS54.CICS.DFHDPFMX,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
     BLDINDEX -
            INFILE(BDSET1) -
            OUTFILE(ADSET1)
/*
//*
```

Figure 23. Sample JCL to create the debugging profiles data sets

The sample JCL creates data sets which contain example debugging profiles. To create empty data sets, remove the following line:

```
 EEXAMPLE RECORD   REMOVE THIS LINE IF SAMPLES NOT REQUIRED
```

Defining the debugging profiles data sets as VSAM RLS files

Define VSAM RLS files when you want to share profiles between several CICS regions in the same sysplex.

CICS provides the following sample definitions:

```
*---------------------------------------------------------------*
*    Define base file for Debugging Profiles (RLS)              *
*---------------------------------------------------------------*
DEFINE FILE(DFHDPFMB) GROUP(DFHDPVR)
DESCRIPTION(Debugging Profiles base file - VSAM RLS)
        RLSACCESS(YES)            TABLE(NO)
        LSRPOOLNUM(1)             DSNSHARING(ALLREQS)
        STRINGS(5)                STATUS(ENABLED)
        OPENTIME(FIRSTREF)        DISPOSITION(SHARE)
        DATABUFFERS(3)            INDEXBUFFERS(2)
        RECORDFORMAT(V)           READINTEG(REPEATABLE)
        ADD(YES)                  BROWSE(NO)
        DELETE(YES)               READ(YES)
        UPDATE(YES)               JOURNAL(NO)
        JNLREAD(NONE)             JNLSYNCREAD(NO)
        JNLUPDATE(NO)             JNLADD(NONE)
        JNLSYNCWRITE(YES)         RECOVERY(NONE)
        FWDRECOVLOG(NO)           BACKUPTYPE(STATIC)
*---------------------------------------------------------------*
*    Define path file for Debugging Profiles (RLS)              *
*---------------------------------------------------------------*
DEFINE FILE(DFHDPFMP) GROUP(DFHDPVR)
DESCRIPTION(Debugging Profiles path file - VSAM RLS)
        RLSACCESS(YES)            TABLE(NO)
        LSRPOOLNUM(1)             DSNSHARING(ALLREQS)
        STRINGS(10)               STATUS(ENABLED)
        OPENTIME(FIRSTREF)        DISPOSITION(SHARE)
        DATABUFFERS(11)           INDEXBUFFERS(10)
        RECORDFORMAT(V)           READINTEG(REPEATABLE)
        ADD(YES)                  BROWSE(NO)
        DELETE(YES)               READ(YES)
        UPDATE(YES)               JOURNAL(NO)
        JNLREAD(NONE)             JNLSYNCREAD(NO)
        JNLUPDATE(NO)             JNLADD(NONE)
        JNLSYNCWRITE(YES)         RECOVERY(NONE)
        FWDRECOVLOG(NO)           BACKUPTYPE(STATIC)
```

Figure 24. Resource definitions for debugging profiles data sets defined as VSAM RLS files

1. Copy the sample FILE definitions for DFHDPFMB and DFHDPFMP resource to another group.
2. Add the DSNAME attribute:
   - For DFHDPFMB, specify the name of the debugging profiles base data set. For example:

     ```
     DSNAME ==> CICSTS54.CICS.DFHDPFMB
     ```

   - For DFHDPFMP, specify the name of the debugging profiles base data set. For example:

     ```
     DSNAME ==> CICSTS54.CICS.DFHDPFMP
     ```

   Alternatively, you can omit the DSNAME attribute, and include a DD card in the CICS startup JCL. For example:

   ```
   //DFHDPFMB DD DSN=CICSTS54.CICS.DFHDPFMB,DISP=SHR
   //DFHDPFMP DD DSN=CICSTS54.CICS.DFHDPFMP,DISP=SHR
   ```

3. Install the FILE definitions.

Defining the debugging profiles data sets as VSAM non-RLS files

Define VSAM non-RLS files when you do not need to share profiles between regions in the same sysplex.

CICS provides the following sample definitions:

```
*---------------------------------------------------------------*
*    Define base file for Debugging Profiles (non-RLS)          *
*---------------------------------------------------------------*
 DEFINE FILE(DFHDPFMB) GROUP(DFHDPVSL)
 DESCRIPTION(Debugging Profiles Base File)
         RLSACCESS(NO)             LSRPOOLNUM(1)
         READINTEG(UNCOMMITTED)    DSNSHARING(ALLREQS)
         STRINGS(10)               STATUS(ENABLED)
         OPENTIME(FIRSTREF)        DISPOSITION(SHARE)
         DATABUFFERS(11)           INDEXBUFFERS(10)
         TABLE(NO)                 RECORDFORMAT(V)
         ADD(YES)                  BROWSE(YES)
         DELETE(YES)               READ(YES)
         UPDATE(YES)               JOURNAL(NO)
         JNLREAD(NONE)             JNLSYNCREAD(NO)
         JNLUPDATE(NO)             JNLADD(NONE)
         JNLSYNCWRITE(NO)          RECOVERY(NONE)
         FWDRECOVLOG(NO)           BACKUPTYPE(STATIC)
*---------------------------------------------------------------*
*    Define path file for Debugging Profiles (non-RLS)          *
*---------------------------------------------------------------*
 DEFINE FILE(DFHDPFMP) GROUP(DFHDPVSL)
 DESCRIPTION(Debugging Profiles Path File)
         RLSACCESS(NO)             LSRPOOLNUM(1)
         READINTEG(UNCOMMITTED)    DSNSHARING(ALLREQS)
         STRINGS(10)               STATUS(ENABLED)
         OPENTIME(FIRSTREF)        DISPOSITION(SHARE)
         DATABUFFERS(11)           INDEXBUFFERS(10)
         TABLE(NO)                 RECORDFORMAT(V)
         ADD(YES)                  BROWSE(YES)
         DELETE(YES)               READ(YES)
         UPDATE(YES)               JOURNAL(NO)
         JNLREAD(NONE)             JNLSYNCREAD(NO)
         JNLUPDATE(NO)             JNLADD(NONE)
         JNLSYNCWRITE(NO)          RECOVERY(NONE)
         FWDRECOVLOG(NO)           BACKUPTYPE(STATIC)
```

Figure 25. Resource definitions for debugging profiles data sets defined as VSAM non-RLS files

1. Copy the sample FILE definitions for DFHDPFMB and DFHDPFMP resource to another group.
2. Add the DSNAME attribute:
   - For DFHDPFMB, specify the name of the debugging profiles base data set. For example:

     ```
     DSNAME ==> CICSTS54.CICS.DFHDPFMB
     ```

   - For DFHDPFMP, specify the name of the debugging profiles base data set. For example:

     ```
     DSNAME ==> CICSTS54.CICS.DFHDPFMP
     ```

   Alternatively, you can omit the DSNAME attribute, and include a DD card in the CICS startup JCL. For example:

   ```
   //DFHDPFMB DD DSN=CICSTS54.CICS.DFHDPFMB,DISP=SHR
   //DFHDPFMP DD DSN=CICSTS54.CICS.DFHDPFMP,DISP=SHR
   ```

3. Install the FILE definitions.

Defining the debugging profiles data sets as remote files

Define remote files when you want to use profiles that are stored in a region that is connected using MRO or ISC.

CICS provides the following sample definitions:

```
*---------------------------------------------------------------------*
*    Define base file for Debugging Profiles (non-RLS remote)       *
*---------------------------------------------------------------------*
DEFINE FILE(DFHDPFMB) GROUP(DFHDPVSR)
DESCRIPTION(Debugging Profile Base File - VSAM Remote)
        REMOTESYSTEM(CICA)        REMOTENAME(DFHDPFMB)
*---------------------------------------------------------------------*
*    Define path file for Debugging Profiles (non-RLS remote)       *
*---------------------------------------------------------------------*
DEFINE FILE(DFHDPFMP) GROUP(DFHDPVSR)
DESCRIPTION(Debugging Profile Path File - VSAM Remote)
        REMOTESYSTEM(CICA)        REMOTENAME(DFHDPFMP)
```

Figure 26. Resource definitions for debugging profiles data sets defined as remote files

1. Copy the sample FILE definitions for DFHDPFMB and DFHDPFMP resource to another group.
2. Add the DSNAME attribute:
   - For DFHDPFMB, specify the name of the debugging profiles base data set. For example:

     ```
     DSNAME ==> CICSTS54.CICS.DFHDPFMB
     ```

   - For DFHDPFMP, specify the name of the debugging profiles path data set. For example:

     ```
     DSNAME ==> CICSTS54.CICS.DFHDPFMP
     ```

   Alternatively, you can omit the DSNAME attribute, and include a DD card in the CICS startup JCL. For example:

   ```
   //DFHDPFMB DD DSN=CICSTS54.CICS.DFHDPFMB,DISP=SHR
   //DFHDPFMP DD DSN=CICSTS54.CICS.DFHDPFMP,DISP=SHR
   ```

3. Install the FILE definitions.

If you define remote files, you will need to install corresponding file definitions in the remote system.

Specifying CICS system initialization parameters

You can use the CICS system initialization parameters to modify CICS system attributes when you start your CICS regions. Different methods are available to define the parameters to CICS. Each system initialization parameter is described, including the syntax and a detailed description.

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 your own versions of CICS control tables (for example, DFHMCTxx)
3. Special information used to control the initialization process

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 can specify which SIT you want, and other system initialization parameters (with a few exceptions), in any of three ways:

1. In the PARM parameter of the EXEC PGM=DFHSIP statement
2. In the SYSIN data set defined in the startup job stream
3. Through the system operator's console

You can also use these methods of input to the system initialization process to override most of the system initialization parameters assembled in the SIT.

Some system initialization parameters cannot be coded in the SIT; other parameters cannot be specified in any other way. Finally, some system initialization parameters cannot be specified through the system operator's console. For a summary of the parameters and how they can be specified, see System initialization parameter descriptions and summary.

When you upgrade to a new CICS release

When you upgrade to a new CICS release, 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 z/OS, Version 5 Release 4 macro libraries.

If you have system initialization parameters defined in CICS startup procedures, you must modify these also.

To avoid generating specific system initialization tables for each CICS region, a simple solution is to let CICS load the default, unsuffixed table (DFHSIT) at startup, and supply the system initialization parameters for each region in a SYSIN data set. For more information about the source of the default system initialization table, see The default system initialization table.

System initialization parameters to set up a CICS region

The CICS system initialization parameters that you must configure, or change from their default values, to set up a CICS region are listed. The system initialization parameters that you might consider whether to configure are also listed.

For parameters that are not listed in the following sections, it is unlikely that you need to change them from their defaults unless there are specific requirements for your environment.

For a complete list of system initialization parameters and their defaults, see System initialization parameter descriptions and summary.

System initialization parameters that you must configure

You must configure the following system initialization parameters:

APPLID
The **APPLID** parameter specifies the z/OS Communications Server application identifier for this CICS region. See APPLID system initialization parameter.

CICSSVC
The **CICSSVC** parameter specifies the number that you have assigned to the CICS type 3 SVC. See CICSSVC system initialization parameter.

GRPLIST
The **GRPLIST** system initialization parameter specifies the names (each 1 - 8 characters) of up to four lists of resource definition groups on the CICS system definition (CSD) file. See GRPLIST system initialization parameter.

ISC
The **ISC** system initialization parameter specifies whether the CICS programs required for multiregion operation (MRO) and intersystem communication over SNA are to be included. See ISC system initialization parameter.

MXT
The **MXT** system initialization parameter specifies the maximum number, in the range 10 through 2000, of user tasks that can exist in a CICS system at the same time. The MXT value does not include CICS system tasks. See MXT system initialization parameter.

PLTPI
The **PLTPI** system initialization parameter specifies a program list table, which contains a list of programs to be run in the final stages of system initialization. See PLTPI system initialization parameter.

PLTSD
The **PLTSD** system initialization parameter specifies a program list table that contains a list of programs to be run during system termination. See PLTSD system initialization parameter.

SIT
The **SIT** system initialization parameter specifies the suffix, if any, of the system initialization table that you want CICS to load at the start of initialization. See SIT system initialization parameter.

START
The **START** system initialization parameter specifies the type of start for the system initialization program. See START system initialization parameter.

SUFFIX
The **SUFFIX** system initialization parameter specifies the last two characters of the name of this system initialization table. See SUFFIX system initialization parameter.

SYSIDNT
The **SYSIDNT** system initialization parameter specifies a 1- to 4-character name that is known only to your CICS region. See SYSIDNT system initialization parameter.

TCPIP
The **TCPIP** system initialization parameter specifies whether CICS TCP/IP services are to be activated at CICS startup. See TCPIP system initialization parameter.

USSHOME
The **USSHOME** system initialization parameter specifies the name and path of the root directory for CICS Transaction Server files on z/OS UNIX. See USSHOME system initialization parameter.

System initialization parameters that you might need to configure

You must consider whether you need to configure the following system initialization parameters:

AICONS
The **AICONS** parameter specifies whether you want autoinstall support for consoles. See AICONS system initialization parameter.

AIRDELAY
The **AIRDELAY** parameter specifies the delay period that elapses after an emergency restart before autoinstalled terminal and APPC connection entries that are not in session are deleted. See AIRDELAY system initialization parameter.

CLINTCP
The **CLINTCP** parameter specifies the default client code page to be used by the DFHCNV data conversion table, but only if the **CLINTCP** parameter in the DFHCNV macro is set to SYSDEF. See CLINTCP system initialization.

CMDPROT
The **CMDPROT** parameter specifies whether to allow or inhibit CICS validation of start addresses of storage referenced as output parameters on EXEC CICS commands. See CMDPROT system initialization parameter.

CONFDATA
The **CONFDATA** parameter specifies whether CICS is to suppress user data that might otherwise appear in CICS trace entries or in dumps. See CONFDATA system initialization parameter.

CONFTXT
The **CONFTXT** system initialization parameter specifies whether CICS is to prevent z/OS Communications Server from tracing user data. See CONFTXT system initialization parameter.

CPSMCONN
The **CPSMCONN** parameter specifies whether you want CICS to invoke the specified CICSPlex® SM component during initialization of the region. See CPSMCONN system initialization parameter.

CRLPROFILE
The **CRLPROFILE** parameter specifies the name of the profile that is used to authorize CICS to access the certification revocation lists (CRLs) that are stored in an LDAP server. See CRLPROFILE system initialization parameter.

CSDDISP
The **CSDDISP** parameter specifies the disposition of the data set to be allocated to the CSD. See CSDDISP system initialization parameter.

CSDDSN
The **CSDDSN** parameter specifies the 1-44 character JCL data set name (DSNAME) to be used for the CSD. See CSDDSN system initialization parameter.

CSDFRLOG
The **CSDFRLOG** parameter specifies a number that corresponds to the journal name that CICS uses to identify the forward recovery log stream for the CSD. See CSDFRLOG system initialization parameter.

CSDLSRNO
The **CSDLSRNO** system initialization parameter specifies whether the CSD is to be associated with a local shared resource (LSR) pool. See CSDLSRNO system initialization parameter.

CSDRECOV
The **CSDRECOV**system initialization parameter specifies whether the CSD is a recoverable file. See CSDRECOV system initialization parameter.

DATFORM
The **DATFORM** system initialization parameter specifies the external date display standard that you want to use for CICS date displays. See DATFORM system initialization parameter.

DBCTLCON
The **DBCTLCON** system initialization parameter specifies whether you want CICS to start the DBCTL connection automatically during initialization. See DBCTLCON system initialization parameter.

DB2CONN
The **DB2CONN** system initialization parameter specifies whether you want CICS to start the DB2 connection automatically during initialization. See DB2CONN system initialization parameter.

DFLTUSER
The **DFLTUSER** system initialization parameter specifies the RACF userid of the default user; that is, the user whose security attributes are used to protect CICS resources in the absence of other, more specific, user identification. See DFLTUSER system initialization parameter.

DOCCODEPAGE
The **DOCCODEPAGE** system initialization parameter specifies the default host code page to be used by the document domain. See DOCCODEPAGE system initialization parameter.

DSALIM
The **DSALIM** system initialization parameter specifies the upper limit of the total amount of storage within which CICS can allocate the individual dynamic storage areas (DSAs) that reside in 24-bit storage (below 16 MB, also known as below the line). See DSALIM system initialization parameter.

DSRTPGM
The **DSRTPGM** system initialization parameter specifies the name of a distributed routing program. The distributed routing program must be specified in the routing region and each target CICS region. See DSRTPGM system initialization parameter.

DTRPGM
The **DTRPGM** system initialization parameter specifies the name of a dynamic routing program. See DTRPGM system initialization parameter.

DUMPSW
The **DUMPSW** system initialization parameter specifies whether you want CICS to switch automatically to the next dump data set when the first is full. See DUMPSW system initialization parameter.

EDSALIM
The **EDSALIM** system initialization parameter specifies the upper limit of the total amount of storage within which CICS can allocate the individual extended dynamic storage areas (EDSAs) that reside in 31-bit (above-the-line) storage; that is, above 16 MB but below 2 GB. See EDSALIM system initialization parameter.

FCQRONLY
The **FCQRONLY** system initialization parameter specifies whether you want CICS to force all file control requests to run under the CICS QR TCB. This parameter applies to file control requests that access VSAM RLS files and local VSAM LSR files. Requests for all other file types always run on the QR TCB. See FCQRONLY system initialization parameter.

GMTEXT
The **GMTEXT** system initialization parameter specifies whether the default logon message text (WELCOME TO CICS) or your own message text is to be displayed on the screen. See GMTEXT system initialization parameter.

GMTRAN
The **GMTRAN** system initialization parameter specifies the name of a transaction ID. See GMTRAN system initialization parameter.

GRNAME
The **GRNAME** system initialization parameter specifies the z/OS Communications Server generic resource name, as 1 through 8 characters, under which a group of CICS terminal-owning regions in a CICSplex register to z/OS Communications Server. See GRNAME system initialization parameter.

ICV
The **ICV** system initialization parameter specifies the region exit time interval in milliseconds. See ICV system initialization parameter.

ICVR
The **ICVR** system initialization parameter specifies the default runaway task time interval in milliseconds as a decimal number. See ICVR system initialization parameter.

INITPARM
The **INITPARM** system initialization parameter specifies that parameters are to be passed to application programs that use the **ASSIGN INITPARM** command. See INITPARM system initialization parameter.

IRCSTRT
The **IRCSTRT** system initialization parameter specifies whether IRC is to be started up at system initialization. See IRCSTRT system initialization parameter.

JVMPROFILEDIR
The **JVMPROFILEDIR** system initialization parameter specifies the name (up to 240 characters long) of a z/OS UNIX directory that contains the JVM profiles for CICS. CICS searches this directory for the profiles it needs to configure JVMs. See JVMPROFILEDIR system initialization parameter.

KEYRING
The **KEYRING** system initialization parameter specifies the fully qualified name of the key ring, within the external security manager's database, that contains the keys and X.509 certificates used by CICS support for the secure sockets layer (SSL) and for web services security. See KEYRING system initialization parameter.

LGNMSG
The **LGNMSG** system initialization parameter specifies whether z/OS Communications Server logon data is to be made available to an application program. See LGNMSG system initialization parameter.

LOCALCCSID
The **LOCALCCSID** system initialization parameter specifies the default CCSID for the local region. See LOCALCCSID system initialization parameter.

LPA
The **LPA** system initialization parameter specifies whether CICS and user modules can be used from the link pack areas. See LPA system initialization parameter.

MN The **MN** system initialization parameter specifies whether monitoring is to be switched on or off at initialization. See MN system initialization parameter.

MNPER
The **MNPER** system initialization parameter specifies whether the monitoring performance class is to be made active during CICS initialization. See MNPER system initialization parameter.

MQCONN
The **MQCONN** system initialization parameter specifies whether you want CICS to start a connection to WebSphere® MQ automatically during initialization. See MQCONN system initialization parameter.

MSGCASE
The **MSGCASE** system initialization parameter specifies how you want the message domains to display mixed case messages. See MSGCASE system initialization parameter.

NATLANG
The **NATLANG** system initialization parameter specifies the single-character code for the language to be supported in this CICS run. See NATLANG system initialization parameter.

PGCHAIN
The PGCHAIN system initialization parameter specifies the character string that is identified by terminal control as a BMS terminal page-chaining command. See PGCHAIN system initialization parameter .

PGCOPY
The **PGCOPY** system initialization parameter specifies the character string that is identified by terminal control as a BMS command to copy output from one terminal to another. See PGCOPY system initialization parameter.

PGPURGE
The **PGPURGE** system initialization parameter specifies the character string that is identified by terminal control as a BMS terminal page-purge command. See PGPURGE system initialization parameter.

PGRET
The **PGRET** system initialization parameter specifies the character string that is recognized by terminal control as a BMS terminal page-retrieval command. See PGRET system initialization parameter.

PLTPISEC
The **PLTPISEC** system initialization parameter specifies whether you want CICS to perform command security or resource security checking for PLT programs during CICS initialization. See PLTPISEC system initialization parameter.

PLTPIUSR
The **PLTPIUSR** system initialization parameter specifies the user ID that CICS uses for security checking for PLT programs that run during CICS initialization. See PLTPIUSR system initialization parameter.

PSBCHK
The **PSBCHK** system initialization parameter specifies whether CICS is to perform PSB authorization checks for remote terminal users who use transaction routing to initiate a transaction in this CICS region to access an attached IMS system. See PSBCHK system initialization parameter.

RLS
The **RLS** system initialization parameter specifies whether CICS is to support VSAM record-level sharing (RLS). See RLS system initialization parameter.

RRMS
The **RRMS** system initialization parameter specifies whether CICS is to register as a resource manager with recoverable resource management services (RRMS). See RRMS system initialization parameter.

SDTRAN
The **SDTRAN** system initialization parameter specifies the name of the shutdown transaction to be started at the beginning of normal and immediate shutdown. See SDTRAN system initialization parameter.

SECPRFX
The **SECPRFX** system initialization parameter specifies whether CICS prefixes the resource names in any authorization requests to the external security manager. See SECPRFX system initialization parameter.

SPOOL
The **SPOOL** system initialization parameter specifies whether the system spooling interface is required. See SPOOL system initialization parameter.

SRBSVC
The **SRBSVC** system initialization parameter specifies the number that you have assigned to the CICS type 6 SVC. See SRBSVC system initialization parameter.

SRVERCP
The **SRVERCP** system initialization parameter specifies the default server code page to be used by the DFHCNV data conversion table but only if the SRVERCP parameter in the DFHCNV macro is set to SYSDEF. See SRVERCP system initialization parameter.

STARTER
The **STARTER** system initialization parameter specifies whether the generation of starter system modules (with $ and # suffixes) is permitted, and various MNOTES are suppressed. See STARTER system initialization parameter.

STATRCD
The **STATRCD** system initialization parameter specifies the interval statistics recording status at CICS initialization. See STATRCD system initialization parameter.

STGPROT
The **STGPROT** system initialization parameter specifies whether you want storage protection to operate in the CICS region. See STGPROT system initialization parameter.

TBEXITS
The **TBEXITS** system initialization parameter specifies the names of your

backout exit programs for use during emergency restart backout processing. See TBEXITS system initialization parameter.

TCT
The **TCT** system initialization parameter specifies which terminal control table, if any, is to be loaded. See TCT system initialization parameter.

TD The **TD** system initialization parameter specifies the number of VSAM buffers and strings to be used for intrapartition transient data (TD). See TD system initialization parameter.

TRANISO
The **TRANISO** system initialization parameter specifies, together with the **STGPROT** system initialization parameter, whether you want transaction isolation in the CICS region. See TRANISO system initialization parameter.

TS The **TS** system initialization parameter specifies whether you want to perform a cold start for temporary storage, as well as the number of VSAM buffers and strings to be used for auxiliary temporary storage. See TS system initialization parameter.

WLMHEALTH
The **WLMHEALTH** system initialization parameter specifies the time interval and the health adjustment value to be used by CICS on z/OS Workload Manager Health API (IWM4HLTH) calls, which CICS makes to inform z/OS WLM about the health state of a CICS region. See WLMHEALTH.

XAPPC
The **XAPPC** system initialization parameter specifies whether RACF session security can be used when establishing APPC sessions. See XAPPC system initialization parameter.

XDB2
The **XDB2** system initialization parameter specifies whether you want CICS to perform DB2ENTRY security checking. See XDB2 system initialization parameter.

Specifying DFHSIT macro parameters

You can specify most system initialization parameters in a DFHSIT macro.

Procedure

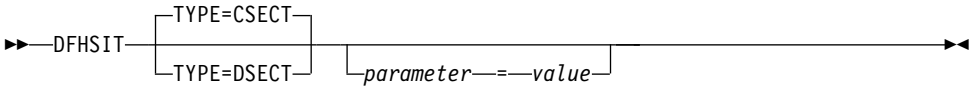
1. Code the following macro parameter first:

   TYPE={CSECT|DSECT}
   This parameter specifies the type of SIT to be generated.
   CSECT
   A regular control section that is normally used.
   DSECT
   A dummy control section.

   For example:

   ```
   DFHSIT TYPE=CSECT,      *
   ```

   The asterisk (*) can be any character in column 72 to indicate the macro continues on the next line.
2. Use the following syntax to add system initialization parameters to the macro:



where *parameter* is the system initialization parameter that you want to code, and *value* is the appropriate parameter value. You should code the parameters and keywords in uppercase, except for parameters where case is important. For example, any parameter that specifies the name of a z/OS UNIX directory is case sensitive. The system initialization parameters are described in detail in System initialization parameter descriptions and summary. For example:

```
ADI=30, *
AIBRIDGE=AUTO, *
AIEXIT=DFHZATDX, *
AILDELAY=0, *
```

3. Terminate your macro parameters with the following statement:

```
END DFHSITBA
```

Example

For information on the sample system initialization tables provided, see Defining resources in CICS control tables.

Defining CICS resource table and module keywords

About this task

The following table shows the system initialization keywords for those CICS resources that:

- Have a suffix option, or
- Result in a dummy program or table if you code resource=NO, or
- Can perform individual cold starts

Table 9. Summary of resources with a suffix, a dummy load module, or a COLD option

DFHSIT keyword 1	Default 2	Suffix 3	Dummy 4	COLD start 5
BMS	FULL	-	-	COLD
CLT	-	xx	-	-
DIP	NO	-	program	-
FCT	YES	xx	-	-
ICP	-	-	-	COLD
MCT	NO	xx	6	-
PDIR	NO	xx	-	-
PLTPI	NO	xx	-	-
PLTSD	NO	xx	-	-
RST	NO	xx	-	-
SRT	YES	xx	-	-
TCT	YES	xx	table	-
TS	-	-	-	COLD
TST	NO	xx	-	-
XLT	NO	xx	-	-

Note:

1. When the DFHSIT keyword is specified with more than one value, these values must be enclosed within parentheses: for example, BMS=(FULL,COLD).
2. The **Default** column indicates the default value for the keyword in the DFHSIT macro.

   For keywords with the suffix option, if you code YES in the SIT, an unsuffixed version of the table or program is loaded. For example, TCT=YES results in a table called DFHTCT 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:

   ```
   FCT=, or FCT=YES
   ```

   blanks are appended to DFHFCT, and these unsuffixed names are used during initialization.

   The result of specifying ***keyword=,*** as a system initialization parameter through PARM, SYSIN, 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.
3. The **Suffix** column indicates whether you can code a suffix. (xx indicates that a suffix can be coded.)

   A suffix can be any 1 or 2 characters, but you must not use DY, and 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.
4. 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 107.
5. The **COLD start** column indicates whether the resource can be forced to start COLD. (COLD indicates that a cold start can be performed for the resource individually.) TST and cold start ensure that a cold start is performed for temporary storage or the whole system if you make any change to the TST.

   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 ALL.

   For more information about CICS table and program selection, see "Selecting versions of CICS programs and tables" on page 107.
6. 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.

The default system initialization table

This default SIT is in CICSTS54.CICS.SDFHAUTH, and its source, named DFHSIT$$, is in CICSTS54.CICS.SDFHSAMP.

The parameters and values in the macro source statements used to assemble the default system initialization table are listed in Table 10 on page 100.

Table 10. DFHSIT, the pregenerated default system initialization table

Parameter	Default value	Description
ADI	30	XRF(B) - Alternate delay interval
AIBRIDGE	AUTO	Bridge Autoinstall URM
AICONS	NO	No autoinstall for MVS CONSOLES
AIEXIT	DFHZATDX	Autoinstall user program name
AILDELAY	0	Delete delay period for AI terminals
AIQMAX	100	Maximum number of terminals queued for AI
AIRDELAY	700	Restart delay period for AI terminals
AKPFREQ	4000	Activity keypoint frequency
APPLID	DBDCCICS	z/OS Communications Server APPL identifier
AUTCONN	0	Autoconnect delay
AUTODST	NO	Language Environment® automatic storage tuning
AUTORESETTIME	IMMEDIATE	Time-of-day synchronization
AUXTR	OFF	Auxiliary trace option
AUXTRSW	NO	Auxiliary trace autoswitch facility
BMS	FULL, UNALIGN, DDS	Basic Mapping Support options
BRMAXKEEPTIME	86400	Bridge Max Keeptime
CICSSVC	216	The CICS SVC number
CILOCK	NO	Do not keep CI lock after read update
CLSDSTP	NOTIFY	Notification for ISSUE PASS command
CLT		The command list table option or suffix
CMDPROT	YES	Exec storage command checking
CMDSEC	ASIS	API command security checking
CONFDATA	SHOW	Show confidential data in dump and trace
CONFTXT	NO	Do not prevent z/OS Communications Server tracing user data
CPSMCONN	NO	Do not connect to CPSM
CRLPROFILE		Name of profile that allows CICS to access certificate revocation lists
CSDACC	READWRITE	CSD access
CSDBKUP	STATIC	Backup type of CSD (STATIC or DYNAMIC)
CSDBUFND		Number of data buffers for the CSD
CSDBUFNI		Number of index buffers for the CSD
CSDDISP		CSD disposition for dynamic allocation
CSDDSN		CSD data set name for dynamic allocation
CSDFRLOG	NO	Journal ID for CSD forward recovery
CSDINTEG	UNCOMMITTED	Read integrity = uncommitted
CSDJID	NO	Journal ID for CSD auto journaling

Table 10. DFHSIT, the pregenerated default system initialization table (continued)

Parameter	Default value	Description
CSDLSRNO	1	The VSAM LSR pool number for the CSD
CSDRECOV	NONE	CSD recoverable file option
CSDRLS	NO	Use traditional VSAM
CSDSTRNO	6	CSD number of strings
CWAKEY	USER	CWA storage key
DAE	NO	SDUMPS will not be suppressed by DAE
DATFORM	MMDDYY	CSA date format
DB2CONN	NO	Do not connect to DB2 at CICS startup
DBCTLCON	NO	Do not connect to DBCTL at CICS start
DEBUGTOOL	NO	No Debugging Tool access
DFLTUSER	CICSUSER	Default user
DIP	NO	Batch data interchange program
DISMACP	YES	Disable macro programs
DOCCODEPAGE	037	Default host code page
DSALIM	5M	Upper limit of DSA below 16 MB line
DSHIPIDL	020000	Delete shipped idle time
DSHIPINT	120000	Delete shipped interval
DSRTPGM	NONE	Distributed routing program
DTRPGM	DFHDYP	Dynamic routing program
DTRTRAN	CRTX	Default dynamic transaction routing transid
DUMP	YES	Dump option
DUMPDS	AUTO	CICS dump data set opening option
DUMPSW	NO	Dump data set autoswitch option
DURETRY	30	SDUMP total retry time (in seconds)
EDSALIM	800M	Upper limit of DSA in 31-bit storage
EODI	E0	End-of-data indicator for sequential devices
ESMEXITS	NOINSTLN	External security manager exits
FCT	NO	File control table option or suffix
FCQRONLY	YES	Threadsafe FC runs on QR TCB
FEPI	NO	Front-End Programming Interface
FLDSEP	' ' (four blanks)	End-of-field separator characters
FLDSTRT	' ' (one blank)	Field start character for built-in function
FORCEQR	NO	Do not force QR for threadsafe programs
FSSTAFF	NO	Function-shipped START affinity option
FTIMEOUT	30	File timeout 30 seconds
GMTEXT	'WELCOME TO CICS'	Good-morning message text
GMTRAN	CSGM	Initial transaction
GNTRAN	NO	Signoff transaction
GRNAME		Generic resource name for CICS TORs
GRPLIST	DFHLIST	List name of CSD groups for startup

Table 10. DFHSIT, the pregenerated default system initialization table (continued)

Parameter	Default value	Description
GTFTR	OFF	GTF trace option
HPO	NO	z/OS Communications Server High Performance Option (HPO)
ICP		Interval control program start option
ICV	1000	Region exit interval (milliseconds)
ICVR	2000	Runaway task interval (milliseconds)
ICVTSD	0	Terminal scan delay interval
INITPARM		Initialization parameters for programs
INTTR	ON	CICS internal trace option
IRCSTRT	NO	Interregion communication start
ISC	NO	Intersystem communication option
JESDI	30	JES delay interval for XRF alternate
JVMCCSIZE	24M	Shared Class Cache size
JVMCCSTART	AUTO	Start Shared Class Cache when needed
JVMPROFILEDIR	/usr/lpp/cicsts /cicsts54/JVMProfiles	JVM profile directory
KEYRING		Key ring to be used by SSL support
LGDFINT	5	Log defer interval in Log Manager
LGNMSG	NO	Extract z/OS Communications Server logon data
LLACOPY	YES	Use MVS LLACOPY support
LPA	NO	Use-LPA option for CICS/user modules
MAXSOCKETS	65535	Maximum number of IP sockets
MAXSSLTCBS	32	Limit on number of SSL TCBs
MCT	NO	Monitoring control table option or suffix
MINTLSLEVEL	TLS10	Level of encryption for TLS
MN	OFF	CICS monitoring option
MNCONV	NO	Monitoring converse recording option
MNEXC	OFF	Monitoring exception class option
MNFREQ	0	Monitoring frequency period
MNIDN	OFF	Monitoring identity class option
MNPER	OFF	Monitoring performance class option
MNRES	OFF	Monitoring resource class option
MNSYNC	NO	Monitoring syncpoint recording option
MNTIME	GMT	Monitoring timestamp (GMT or LOCAL)
MQCONN	NO	Do not connect to MQ at startup
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

Table 10. DFHSIT, the pregenerated default system initialization table (continued)

Parameter	Default value	Description
MXT	250	Maximum number of tasks in CICS
NATLANG	E	List of national languages
NCPLDFT	DFHNC001	Named counter default pool name
NONRLSRECOV	VSAMCAT	Select location of recovery options for non-RLS files
NQRNL	NO	RNL processing by z/OS global resource serialization
OPERTIM	120	Write to operator timeout (seconds)
OPNDLIM	10	OPNDST/CLSDST request limit
PARMERR	INTERACT	System initialization parameter errors option
PDI	30	Primary delay interval - XRF active
PDIR	NO	DL/I PSB directory option or suffix
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 or suffix
PLTPISEC	NONE	No PLT security checks on PI programs
PLTPIUSR		PLT PI userid = CICS region userid
PLTSD	NO	Program list table SD option or suffix
PRGDLAY	0	BMS purge delay interval
PRINT	NO	Print key option
PRTYAGE	1000	Dispatcher priority aging value
PSBCHK	NO	PSB resource checking required
PSDINT	0	Persistent session delay interval
PSTYPE	SNPS	z/OS Communications Server single node persistent Sessions
PVDELAY	30	Timeout value for LUIT table
QUIESTIM	240	Timeout value for quiesce requests
RAMAX	256	Maximum I/O area for RECEIVE ANY
RAPOOL	50	Maximum RECEIVE ANY request parm. lists
RENTPGM	PROTECT	Reentrant program write protection
RESP	FME	Logical unit response type
RESSEC	ASIS	Resource security check
RLS	NO	RLS option
RLSTOLSR	NO	RLS files in LSRPOOL build calculation

Table 10. DFHSIT, the pregenerated default system initialization table (continued)

Parameter	Default value	Description
RMTRAN	CSGM	XRF alternate recovery transaction
RRMS	NO	Recoverable resource management services
RST	NO	Recovery service table (XRF-DBCTL)
RSTSIGNOFF	NOFORCE	XRF - Re-sign on after takeover
RSTSIGNTIME	500	XRF - sign off timeout value
RUWAPOOL	NO	Allocating storage pool for Language Environment
SDTRAN	CESD	Shutdown transaction
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
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
SNSCOPE	NONE	Multiple CICS sessions per userid
SOTUNING	YES	Whether performance tuning for HTTP connections occurs to protect CICS from unconstrained resource demand.

Table 10. DFHSIT, the pregenerated default system initialization table (continued)

Parameter	Default value	Description
SPCTR	(1,2)	Levels of special tracing required
SPOOL	NO	System spooling interface option
SRT	1$	System recovery table option or suffix
SRBSVC	215	HPO Type 6 SVC number
SSLCACHE	CICS	SSL session ID caching
SSLDELAY	600	SSL timeout value
START	AUTO	CICS system initialization option
STARTER	YES	Starter ($ and #) suffixes option **Note:** The default is NO but the parameter must be set to YES here to enable the SIT to assemble correctly.
STATEOD	0	Statistics end-of-day time
STATINT	010000	Statistics interval time
STATRCD	OFF	Statistics recording status
STGPROT	YES	Storage protection facility
STGRCVY	NO	Storage recovery option
STNTR	1	Level of standard tracing required
SUBTSKS	0	Number of concurrent mode TCBs
SUFFIX	$$	Suffix of this SIT
SYDUMAX	999	Number of SYSDUMPS to be taken
SYSIDNT	CICS	Local system identifier
SYSTR	ON	Master system trace flag
TAKEOVR	MANUAL	XRF alternate takeover option
TBEXITS		Backout exit programs
TCP	YES	Terminal control program option or suffix
TCPIP	YES	TCP/IP support
TCSACTN	NONE	TC Shutdown action
TCSWAIT	4	TC Shutdown wait
TCT	NO	Terminal control table option or suffix
TCTUAKEY	USER	TCT user area storage key
TCTUALOC	ANY	TCT user area ANY
TD	(3,3)	Transient data buffers and strings
TDINTRA	NOEMPTY	Initial state of transient data queues
TRANISO	NO	Transaction Isolation
TRAP	OFF	F.E. global trap exit option
TRDUMAX	999	Number of TRANDUMPS to be taken
TRTABSZ	12288	Internal trace table size in 1K bytes
TRTRANSZ	1024	Transaction dump trace table size
TRTRANTY	TRAN	Transaction dump trace option
TS	(3,3)	Temporary storage buffers and strings

Table 10. DFHSIT, the pregenerated default system initialization table (continued)

Parameter	Default value	Description
TSMAINLIMIT	64M	Upper limit of storage for TS main queues
TST	NO	Temporary storage table option or suffix
UOWNETQL		Qualifier for NETUOWID
USERTR	ON	Master user trace flag
USRDELAY	30	Timeout value for user directory entries
USSCONFIG	/var/cicsts/dfhconfig	The name and path of the root directory for CICS configuration files on z/OS UNIX
USSHOME	/usr/lpp/cicsts/cicsts54	The name and path of the root directory for CICS files on z/OS UNIX
VTAM	YES	z/OS Communications Server access method option
VTPREFIX	\	Client virtual terminal prefix
WEBDELAY	(5,60)	Web timer values
WRKAREA	512	Common work area (CWA) size in bytes
XAPPC	NO	RACF class APPCLU required
XCFGROUP	DFHIR000	XCF group to use for MRO communications
XCMD	YES	SPI use default name for RACF check
XDB2	NO	Security check for DB2ENTRY resources
XDCT	YES	Security check for transient data queues
XFCT	YES	Security check for files
XHFS	YES	Security check for z/OS UNIX files
XJCT	YES	Security check for journals
XLT	NO	Transaction list table option or suffix
XPCT	YES	Security check for started transactions
XPPT	YES	Security check for programs
XPSB	YES	Security check for DL/I PSBs
XRES	YES	For resources subject to XRES security, checks use the default name for the RACF check. For a list of resources subject to XRES security checks, see Resource and command check cross-reference.
XRF	NO	Extended recovery feature (XRF) option
XTRAN	YES	Security check for transaction-attach
XTST	YES	Security check for temporary storage queues
XUSER	YES	Surrogate user checking to be done

Assembling the SIT

When you have coded the DFHSIT macro, you must assemble the SIT.

About this task

Procedure

1. Assemble and link-edit the table into an APF-authorized library, such as CICSTS54.CICS.SDFHAUTH.
2. Include this library in the STEPLIB concatenation of the CICS startup job stream.

Results

If you code a system initialization parameter in your SIT source, and the parameter's keyword is not defined in the CICS-supplied version of the DFHSIT macro, you get an IEV017 warning message from the assembly, as follows:

```
IEV017  ** WARNING **  UNDEFINED KEYWORD PARAM. DEFAULT TO POSITIONAL, INCLUDING KW  --  OPENC/aaaaaaa
```

What to do next

For information about assembling and link-editing CICS control tables, and an explanation of the syntax notation used to describe CICS macros, see Defining resources in CICS control tables.

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.

About this task

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 basic mapping support (BMS) suite, however, you can select from different versions, by explicitly selecting the level of function needed.

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 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 system initialization parameter BMS, 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

There are three ways to exclude programs that are not required.

You can exclude programs by specifying:
1. programname=NO
2. tablename=NO
3. function=NO

Specifying programname=NO:

If you code system initialization parameter *programname*=NO (for example, DIP=NO), you exclude the named management program at CICS system initialization.

You can exclude the following programs by coding *programname*=NO:

- 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 programs control table:

Not all of the CICS programs have a programname parameter in the SIT.

Ansystem initialization parameter alternative method is to code NO on the for the associated table. 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 system recovery table (SRT) can be used in this way, and the associated system recovery program (SRP) will be excluded.

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 your terminal resource definitions are in the CSD.

When you specify TCT=NO, CICS loads a dummy TCT named DFHTCTDY. A pregeneratedCICSTS54.CICS dummy table of this name is supplied in .SDFHLOAD, and the sourceCICSTS54.CICS statements of DFHTCTDY are supplied in .SDFHSAMP. If you specify TCT=NO, a generated table of this name must be available in a library of the DFHRPL concatenation when you start CICS.

The dummy TCT provides **only** the CICS and z/OS Communications Server control blocks that you need if you are using z/OS Communications Server terminals and using the CSD for storing terminal definitions. You define your z/OS Communications Server terminals using the RDO transaction, CEDA, or the DEFINE command of the CSD batch utility program, DFHCSDUP.

Specifying function=NO:

If you code *function*=NO as a system initialization parameter, you exclude the management program associated with the named function at CICS system initialization.

You can exclude functions such as intersystem communication (ISC), the 3270 print-request facility, and the system spooling interface in this way.

Processing system initialization parameters

This section describes the CICS system initialization process.

About this task

It covers:

1. A brief introduction to the process of how you supply system initialization parameters, and the role of the CICS parameter manager domain in this process
2. An explanation of how CICS uses the special system initialization keywords
3. A description of the start and restart classes, and how they are controlled

Supplying system initialization parameters to CICS

The CICS parameter manager domain loads a system initialization table (SIT) at the start of the initialization process.

About this task

You specify the SIT that defines the CICS characteristics appropriate to your needs by coding the suffix of the DFHSITxx load module (where xx is the suffix) on the SIT= system initialization parameter. If you fail to specify the suffix of a SIT, then CICS tries to load an unsuffixed module.

You can modify many of the system initialization parameters dynamically at the beginning of CICS initialization by providing system initialization parameters in the startup job stream, or through the system console. There are also some system initialization parameters that you cannot code in the SIT, and can only supply at startup time. You specify system initialization parameters at startup time in any of three ways:

1. In the PARM parameter of the EXEC PGM=DFHSIP statement
2. In the SYSIN data set defined in the startup job stream
3. Through the system operator's console

You can use just one of these methods, or two, or all three. However, CICS processes these three sources of input in strict sequence, as follows:

1. The PARM parameter
2. The SYSIN data set (but only if SYSIN is coded in the PARM parameter; see SYSIN)
3. The console (but only if CONSOLE is coded in either the PARM parameter or in the SYSIN data set; see CONSOLE(CN))

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 MCT=1$ in the PARM parameter, MCT=2$ in the SYSIN data set, and finally enter MCT=3$ through the console, CICS loads DFHMCT3$.

Using system initialization control keywords

You can use the SYSIN, CONSOLE, and END control keywords at startup to control how CICS is to read system initialization parameters.

About this task

The purpose of these special keywords, and where you can code them, are described as follows:

SYSIN (SI)
This keyword tells CICS to read initialization parameters from the SYSIN data set.

Where to code SYSIN: You can code SYSIN (or SI) only in the PARM parameter of the EXEC PGM=DFHSIP statement. The keyword can appear once only and must be at the end of the PARM parameter. CICS does not read SYSIN until it has finished scanning all of the PARM parameter, or until it reaches a .END before the end of the PARM parameter.

Examples:

```
//stepname   EXEC  PGM=DFHSIP,PARM='SIT=6$,SYSIN,.END'
//stepname   EXEC  PGM=DFHSIP,PARM='SIT=6$,DLI=YES,SYSIN,.END'
```

CONSOLE (CN)

This keyword tells CICS to read initialization parameters from the console. CICS prompts you with message DFHPA1104 when it is ready to read parameters from the console.

Where to code CONSOLE: You can code CONSOLE (or CN) in the PARM parameter of the EXEC PGM=DFHSIP statement or in the SYSIN data set. This keyword can appear either at the end of the PARM parameter or in the SYSIN data set, but code it in one place only.

If you code CONSOLE (or CN) in the PARM parameter, and PARM also contains the SYSIN keyword, CICS does not begin reading parameters from the console until it has finished reading and processing the SYSIN data set. Similarly, wherever you place the CONSOLE keyword in the SYSIN data set, CICS does not begin reading parameters from the console until it has finished reading and processing the SYSIN data set.

Examples:

```
//stepname   EXEC  PGM=DFHSIP,PARM='SIT6$,CONSOLE,.END'
//stepname   EXEC  PGM=DFHSIP,PARM='CONSOLE,SYSIN,.END'
//stepname   EXEC  PGM=DFHSIP,PARM='SIT=6$,CN,SI,.END'
```

If both SYSIN (or SI) and CONSOLE (or CN) appear as keywords of the PARM parameter, the order in which they are coded is irrelevant as long as no other keywords, other than .END, follow them.

.END

The meaning of this keyword varies:

PARM The use of the .END keyword is optional in the PARM parameter. 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 SYSIN) .END denotes the end of the PARM parameter only.

   For example:

   ```
   //stepname   EXEC  PGM=DFHSIP,PARM='SIT6$,SI,CN,.END'
   ```

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:

   ```
   //stepname   EXEC  PGM=DFHSIP,PARM='SIT=6$,.END'
   ```

If .END is not the last entry in the PARM parameter, CICS truncates the PARM parameter and the parameters following the .END keyword are lost.

SYSIN

The use of the .END keyword is optional in the SYSIN data set. If you

omit it, CICS assumes it to be at the end of SYSIN. If you code .END in the SYSIN data set 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 SYSIN data set, .END denotes the end of the SYSIN data set only.
- If you do not code the CONSOLE control keyword in the PARM parameter or in the SYSIN data set, .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 SYSIN data set, or not at the end of a SYSIN 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 SYSIN data set, 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 SYSIN data set:

```
//SYSIN   DD  *
* CICS system initialization parameters
SIT=6$,START=COLD,
PDIR=1$,             ( SUFFIX of PSB directory

.END
/*
```

CONSOLE

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 SYSIN 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 PGM=DFHSIP statement, CICS assumes that there are no SIT overrides or other initialization parameters and tries to load an unsuffixed module named DFHSIT.

As a general rule, the **PARM** parameter must at least specify the suffix of your system initialization table, using the SIT keyword. 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 SYSIN 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 also a SYSIN keyword, CICS does not try to load the SIT until it has also finished scanning the SYSIN data set. In this case, loading the SIT is deferred because there might be other **SIT** parameters coded in the SYSIN 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 the PARM parameter

The following rules apply when coding the **PARM** parameter:

- The maximum number of characters you can code is 100, excluding the opening and closing delimiters, which can be either apostrophes or parentheses.
- All CICS system initialization parameters must be separated by a comma, and the separating commas are included in the 100 character limit. Because of this limiting factor, you might prefer to limit the use of the **PARM** parameter to specify the SYSIN control keyword only.

The rules for coding the **PARM** parameter on an EXEC job control statement are described fully in *z/OS MVS JCL Reference*.

Processing the SYSIN data set

CICS scans the SYSIN 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 SYSIN, it tries to load that SIT, overriding any that was specified in the PARM parameter. If CICS does not find a SIT= parameter in SYSIN, it tries to load any SIT specified in the PARM parameter.

However, if after scanning the PARM parameter and the SYSIN 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 SYSIN data set, CICS prompts you with the following message to enter the SIT suffix as the first parameter through the console:

   ```
   rr DFHPA1921  DBDCCICS  PLEASE SPECIFY THE REQUIRED SIT SUFFIX, OR
                           SPECIFY 'NONE'(UNSUFFIXED).
   ```

2. If you did not specify CONSOLE, CICS tries automatically 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 SYSIN data set until after the SIT has been loaded.

Rules for coding CICS system initialization parameters in the SYSIN data set:

When you code CICS system initialization parameters in the SYSIN data set, observe the following rules.

- You must use a comma to separate parameters that are on the same line.
- You can optionally use a comma at the end of a SYSIN record.
- You can use an asterisk in column 1 to code comments, or to remove an initialization parameter temporarily from a specific execution of CICS.
- You can also add comments after the parameters on a SYSIN line, but they must be preceded by at least one blank character.
- Everything that appears in positions 1 through 80 is treated by CICS as input data.
- You can continue, on another line in SYSIN, parameters that have multiple operands if you make the split immediately after a comma. CICS concatenates the operands, omitting the intervening blanks.
- Generally, you cannot split an individual operand between lines. However, for 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 as follows:

  ```
  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). The apostrophes that delimit the text are mandatory.
- Be careful when you code parameters that use apostrophes, parentheses, or commas as delimiters, because if you do not include the correct delimiters, unpredictable results can occur.

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 SYSIN 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 DFHPA1104 message.

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.

CICS prompts you to enter corrections to any errors it finds in the **PARM** parameter or the SYSIN 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 SYSIN data set, it is the APPLID coded in the SIT that CICS displays in messages DFHPA1912 and DFHPA1915.

Controlling 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.

You can provide system initialization parameters either in the SIT or at run time by one of the three methods described in "Using system initialization control keywords" on page 109.

The role of the CICS catalogs

CICS uses its catalogs to save information between CICS shutdown and the next restart.

The global catalog:

CICS uses the global catalog to save all resource definitions that were installed when CICS shut down.

These are:

- BMS maps sets and partition sets
- Connections and sessions
- Files
- LIBRARY concatenations
- Programs
- Terminals and typeterms
- Transactions and transaction profiles
- Transient data queues

The resource definitions that CICS saves at its shutdown might have been installed during a cold start from a list of groups specified by a group list system initialization parameter, or dynamically using RDO when the CICS region is running.

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. If a subsequent error occurs during the restore, appropriate messages are displayed. The general rule is that you cannot alter installed resource definitions during a restart except by coding START=COLD or START=INITIAL. For details of the results of the possible combinations of CICS restart-type and global catalog state, see "Specifying the START system initialization parameter" on page 115.

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. In other cases the domain information saved in the catalog is always used in a restart. Alternatively, you can enforce system defaults by performing a cold start.

Note: If you have to reinitialize the 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:

1. Initialize the local catalog with an initial set of domain records.
2. 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 Local catalog storage program (DFHSMUTL).
3. Delete and reinitialize the global catalog.

For more information about initializing the local catalog, see "Defining the local catalog" on page 56. 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.

Specifying 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 which of four possible types of start to perform by looking for two records which may or may not be present in the global catalog:

- The recovery manager control record
- The recovery manager autostart override record

Depending on whether either or both of these records exist, and their contents, CICS decides which type of start to perform:

Initial start

CICS performs an initial start in each of the following cases:

- There is a recovery manager autostart override record that specifies AUTOINIT in the global catalog.
- The control record specifies an initial start. (This can happen if a previous initial start failed.)

Note: Because certain domains store information in the local catalog, if you set CICS to perform an initial start, you should reinitialize the local catalog before you bring up CICS. For more information on domains that write to the local catalog, see Domains that write to the local catalog.

Cold start

CICS performs a cold start in the following cases:

- The recovery manager control record specifies a cold start. (This can happen if a previous cold start did not complete.)

- There is both a recovery manager control record (which specifies anything other than an initial start) *and* an autostart override record that specifies AUTOCOLD.

Log records for local resources are purged and resource definitions rebuilt from the CSD or CICS control tables. Units of work on other systems are resynchronized with this system, as described under START=COLD.

Warm start

If the recovery manager control record indicates that the previous run of CICS terminated normally with a successful warm keypoint, CICS performs a warm restart—*unless* the autostart override record specifies AUTOINIT or AUTOCOLD, in which case an initial or cold start is performed.

For the warm restart to be successful, the local catalog must contain the information saved by the CICS domains during the previous execution.

A warm start restores CICS to the state it was in at the previous shutdown.

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 is the system initialization parameter 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.

Emergency start

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.

START=AUTO should be the normal mode of operation, with the choice of start being made by CICS automatically. Use the recovery manager utility program, DFHRMUTL, to set overrides.

START=INITIAL

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.

Note: The global catalog and system log are initialized, and all information in them is lost. Because recovery information for remote systems is not preserved, damage may be done to distributed units of work

You should rarely need to specify START=INITIAL; if you want to reinstall definitions of local resources from the CSD, use START=COLD instead.

Examples of times when an initial start is necessary are:

- When bringing up a new CICS system for the first time.
- After a serious software failure, when the system log has been corrupted.
- If the global catalog is cleared or reinitialized.

- When you want to run CICS with a dummy system log. (If the system log is defined as a dummy, it is ignored.)

If it is necessary to perform an initial start of CICS, you can do so in two ways:

- By specifying START=INITIAL.
- By using the recovery manager utility program, DFHRMUTL, to set the autostart override record to AUTOINIT, and specifying START=AUTO.

Note: An initial start does not automatically reinitialize the local catalog.

START=COLD

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.

Recovery information relating to remote systems or to RMI-connected resource managers is preserved. The CICS log is scanned during startup, and any information regarding unit of work obligations to remote systems, or to non-CICS resource managers (such as DB2) connected through the RMI, is preserved. (That is, any decisions about the outcome of local UOWs, needed to allow remote systems or RMI resource managers to resynchronize their resources, are preserved.)

Note that, on a cold start, the following are *not* preserved:

- Updates to *local* resources that were not fully committed or backed out during the previous execution of CICS. In particular, although remote systems may resynchronize their units of work successfully, local resources updated in those distributed units of work are not locked and need not be in either a committed or a backed-out state.
- Recovery information for remote systems connected by LU6.1 links, or for earlier releases of CICS systems connected by MRO.
- Any program LIBRARY definitions that had been dynamically defined. Only the static DFHRPL concatenation will remain, together with any LIBRARY definitions in the grouplist specified at startup or installed via BAS at startup.

A start initiated by START=COLD is not entirely without reference to the previous run of a CICS system using the same global catalog. If you want to perform a fully cold start of CICS, without reference to any previous execution, code START=INITIAL. If you want to reinstall definitions of local resources from the CSD, use START=COLD.

There may be times when it is necessary to perform a cold start of CICS, irrespective of the type of system termination that CICS recorded in the global catalog. You can do this in two ways:

- By specifying START=COLD.
- By using DFHRMUTL to set the autostart override record to AUTOCOLD, and specifying START=AUTO.

Note: A cold start does not automatically reinitialize the local catalog.

Table 11 on page 118 shows how the effect of the START parameter depends on the state of the CICS global catalog and system log.

Note: If the system log is defined as a dummy, it is ignored.

Table 11. Effect of the START= parameter in conjunction with the global catalog and system log

START parm.	State of global catalog	State of system log	Result at restart
Any.	Not defined to VSAM.	Any.	JCL error.
INITIAL	Defined.	Any.	CICS performs an initial start. The global catalog and system log are initialized.
COLD	Defined but contains no recovery manager control record.	Any.	After prompting for confirmation, CICS performs an initial start. The global catalog and system log are initialized.
COLD	Contains recovery manager records.	Not defined or dummy or empty.	Message DFHRM0401 is issued. Startup fails.
COLD	Contains recovery manager records.	Contains records from previous run.	CICS performs a cold start. Recovery records in the system log that relate to changes to local resources are deleted.
AUTO	Defined but contains no recovery manager control record and no AUTOINIT autostart override.	Any.	Message DFHRM0137 is issued. Startup fails.
AUTO	Contains a recovery manager AUTOINIT autostart override.	Any.	CICS performs an initial start, without prompting. The global catalog and system log are initialized.
AUTO	Contains a recovery manager control record that does *not* indicate an initial start, and an AUTOCOLD autostart override.	Contains records from previous run.	CICS performs a cold start. Recovery records in the system log that relate to changes to local resources are deleted.
AUTO	Contains recovery manager records, but no AUTOINIT override.	Not defined or dummy or empty.	Message DFHRM0401 is issued. Startup fails.
AUTO	Contains a recovery manager control record indicating an initial start.	Any.	CICS performs an initial start. The global catalog and system log are initialized.
AUTO	Contains a recovery manager control record indicating a cold start, and no autostart override.	Contains records from previous run.	CICS performs a cold start. Recovery records in the system log that relate to changes to local resources are deleted.
AUTO	Contains a recovery manager control record indicating an emergency start, and no autostart override.	Contains records from previous run.	CICS performs an emergency start.
AUTO	Contains a recovery manager control record indicating a warm start, and no autostart override.	Contains records from previous run.	CICS performs a warm start.

Note:

1. It is important to keep the CICS global and local catalogs in step. If CICS tries to perform a warm or emergency start and finds that the local catalog has been initialized, startup fails. Therefore, only initialize the local catalog at the same time as the global catalog.
2. It is recommended that you always run the DFHRMUTL and DFHCCUTL utilities in the same job. Run DFHRMUTL first and check its return code before running DFHCCUTL. If you do this, the global and local catalogs should never get out of step.

Table 12 shows the effect of different types of CICS startup on the CICS trace, monitoring, statistics, and dump domains.

Table 12. Effect of the type of startup on CICS domains

Domain	State of the CICS catalogs	Warm or emergency start	Initial or cold start
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.
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. Other dump attributes are 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 z/OS Communications Server session

In an SNA network, the session between CICS and the z/OS Communications Server is started automatically if the Communications Server is started before CICS.

If the z/OS Communications Server is not active when you start CICS, you receive the following messages:

```
F vtamname,USERVAR,ID=generic-applid,VALUE=specific-applid
+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, you can start the CICS-z/OS Communications Server session manually when the Communications Server eventually starts, by using the **CEMT SET VTAM OPEN** command from a supported MVS console or a non-Communications Server terminal.

If the z/OS Communications Server is active, but CICS still cannot open the SNA ACB because Communications Server does not recognize the CICS APPLID, you receive the following messages:

```
F vtamname,USERVAR,ID=generic-applid,VALUE=specific-applid
+DFHSI1592I 'applid' CICS applid not (yet) active to VTAM.
+DFHSI1572  'applid' Unable to OPEN VTAM ACB - RC=00000008, ACB CODE=5A.
```

This might 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 CICS messages.

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, SYSIN, 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—initial, cold, or warm. (For this purpose, the parameter manager domain treats an emergency restart as a warm start.)

Startup is initial or cold
: If startup is initial or 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 by PARM, SYSIN, 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, SYSIN, 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 in NEWSIT system initialization parameter.

Note: The trace domain is an exception to the above rules in that it will always start cold. 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.

End of CICS startup

Whichever type of startup is performed, when the DFHSI1517 message is displayed on the operating system console, CICS is ready to process terminal requests.

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 .

Even though DFHSI1517 signals the end of system initialization, the region might not be fully ready to receive work. This is because in some cases initialization continues after this message. For example, some bundle-defined resources are being installed asynchronously; so resources such as JVMSERVERs required for some applications have not completed initialization, even though the TCP/IP listener is open and accepting work. This could cause transactions to abend. For web services, a pipeline scan might not have completed, so use of the web service before the scan completes will fail.

You can allow CICS to have a warm-up process by setting the **WLMHEALTH** system initialization parameter. If the z/OS Workload Manager health service is active in the region, when the message DFHSI1517 is returned, the warm-up process is started. Then CICS adjusts the region's z/OS WLM health indicator to control flow of work into the region until the region is fully capable to process work. For more information about the CICS warm-up process, see CICS warm-up and cool-down by use of z/OS Workload Manager health service.

Retrieving information about system initialization parameters

You can use the CICSPlex SM API to discover information about CICS system initialization parameters and system initialization parameter overrides. System initialization parameter retrieval is supported by the CICSPlex SM command-level interface, the CICS management client interface (CMCI), and the Web User Interface (WUI).

When you retrieve parameters you have the following options:

- You can retrieve the current values of the system initialization parameters including any override values.
- You can retrieve the original system initialization parameter values as specified at system startup.
- You can retrieve the values from a single override source.

In common with many other CICSPlex SM operations, you can control which CICS regions the retrieval operates on by specifying context and scope.

System initialization parameter retrieval is implemented using the CICSPlex SM resource SYSPARM. The SYSPARM resource has two required parameters, **PARMSRCE** and **PARMTYPE**, associated with the GET operation. You use these parameters to specify which parameters to retrieve according to their source.

You can implement system initialization parameter discovery in the following ways:

- In the CICS Explorer® **SM Operations views** > **Regions view**. Right-click a region and select **Show SIT Parameters**.
- In an API program using the **EXEC CPSM GET** command operating on the SYSPARM object.
- Using the **CMCI GET** method operating on the CICSSystemParameter external resource.
- Using the WUI operations view based on the SYSPARM resource table linked from the CICS region view set.

When you use **EXEC CPSM GET** command or CMCI, you can define a parameter expression using **PARMSRCE** and **PARMTYPE** to specify which parameters to retrieve. In the WUI you can use **PARMSRCE** and **PARMTYPE** as filters to control the records displayed. Both of these parameters are mandatory. For **PARMTYPE**, you must specify a value of SIT. For **PARMSRCE**, specify one of the following options:

COMBINED
: Combination of the original system initialization parameter definitions and any applied parameter overrides

CONSOLE
: Override parameters as specified at startup on the system console

JCL
: Override parameters provided through a JCL EXEC PGM statement

SYSIN
: Override parameters from the startup job stream defined in the SYSIN data set

TABLE
: The original system initialization table values extracted from the DFHSIT*xx* load module

The parameters and override values are returned in the order of the KEYWORD attribute of the SYSPARM resource table irrespective of their order in the source, for example the SYSIN data set.

System parameter values can exceed 255 bytes. In these cases the parameter value is split into multiple records each containing a maximum of 255 bytes of parameter value. Each record has a unique segment number, which can be used to order the records correctly, a total segment number which contains the number of segments for this parameter value, and a total value length field, which contains the overall length of the segment lengths added together.

You must ensure that the system initialization parameters to be retrieved were valid for the CICS region at CICS startup. The retrieval operation can behave inconsistently if invalid parameter values have been corrected from the console at startup. Some parameters show the corrected values while others display their original values. Additionally for some corrected values, the system parameter source location might be returned as the console instead of the original source location.

The following CMCI request retrieves the values of the control tables suffixes set to for region `REGION` in the CICSplex `MYPLEX`.

```
/CICSSystemManagement/CICSSystemParameter/<MYPLEX>/<REGION>?PARAMETER=PARMSRCE(COMBINED)
%20PARMTYPE(SIT)&CRITERIA=KEYWORD%3D++T%20AND%20NOT%20KEYWORD%3DMXT
```

Specifying feature toggles

New features are continually delivered to in-service CICS releases to enhance CICS capabilities. Usually, these features are disabled by default. To enable and use these features in your CICS region, you must set up one or more feature toggle configuration files. In the feature toggle configuration file, you can define feature toggles to switch features on or off. For some features, you can also specify feature configuration options.

Before you begin

A list of features that you can choose to install and configure in your CICS region is available in Toggle-enabled features, support by release. Additional features might also be made available and described in APARs. What new features are available for enablement depends on the APARs that have been applied to your CICS system.

About this task

There are two types of configuration files where you can define feature toggles and their options: a group-level file for setting up a collection of regions, and a region-level file. Both files are optional. If neither is present, default values are used. See the topics that describe individual toggle-enabled features for default values.

The group-level file is typically used to specify which features are enabled in a group of regions. This file can also be used to prevent one or more toggle settings or configuration options from being set by region-level files.

The region-level file is typically used to try out a feature in a test region before it is put into production.

The configuration files are read at CICS startup, immediately after system initialization reads the SIT. The group-level file is read first, followed by the region-level file. Options read in the region-level file will override those set in the group-level file except those that are specified with the parameter **finalize**.

A feature configuration file specifies feature toggles and feature configuration options. A feature toggle takes the following form:

```
[finalise] com.ibm.cics.component.feature = {true|false}
```

A feature can have one or more configuration options. You can specify a feature configuration option in the form:

```
[finalise] com.ibm.cics.component.feature.config_property = value
```

The value can be a number, string, or boolean.

Tip:

- You can use a line-continuation character \ at the point you want the line to break. If a \ character is required in either a name or value, double the \ character to escape line continuation.
- You can use a hash character # at the start of a line to specify a comment line.

The optional parameter **finalize** can be used in the group-level file to prevent a region-level file from changing the toggle setting or option. For example, if you have given developers access to their region-level file, you can prevent them from using certain features.

Procedure

- To set up a region-level configuration file, follow the procedure in "Setting up a region-level configuration file" on page 125.
- To set up group-level configuration files, follow the procedure in "Setting up group-level configuration files" on page 125.

Example

This is an example feature toggle configuration file:

```
# Feature Toggle File
com.ibm.cics.bms.ids=true 1
com.ibm.cics.bms.ids.vtamignore=false 2
com.ibm.cics.bms.ids.action=ABEND 3
```

1 switches on the BMS 3270 Intrusion Detection Service.

2 and **3** are configuration options for the feature.

What to do next

After the CICS region is started, you can see the feature toggles and their values in message DFHPA1956I in the CICS log. The message indicates what settings have been read in and stored from the feature toggle configuration file, but it does not indicate whether and how the feature toggles are being used.

Restriction: Feature toggles are not displayed by CPSM.

Setting up a region-level configuration file

A region-level file can be used to set up feature toggles for a single region.

Procedure

1. Create a file with the name `featuretoggle.properties` in the USSCONFIG directory of the CICS region where you want to enable features. By default, the path is `/var/cicsts/dfhconfig`.

   Note: The CICS installation process creates an empty `featuretoggle.properties` file in `/pathprefix/usr/lpp/cicsts/ussdira/dfhconfig`, which you can copy to the required USSCONFIG directory.

   Tip: If you are not sure which path the CICS region is using, you can see the path in message DFHPA1951I in the system message log.
2. In the file, for each feature that you want to enable, specify the feature's toggle setting and, if necessary, configuration options, as described in "Specifying feature toggles" on page 123.

Results

You have now created a feature toggle region-level configuration file, `featuretoggle.properties`, for the CICS region.

Setting up group-level configuration files

A group-level file can be used to set up feature toggles for a collection of regions. This file can also be used to prevent region-level files from setting feature toggles and configuration options.

Procedure

1. Create a feature toggle configuration file in a directory that is shared by the CICS regions that will use this file.

   Note: Ensure that all the CICS regions that will use this configuration file have read access to the file.
2. In the file, for each feature that you want to enable, specify the feature's toggle setting and, if necessary, configuration options, as described in "Specifying feature toggles" on page 123.

   Note: To prevent region-level files from setting certain feature toggles and configuration options, you can specify the parameter **`finalize`** in the toggle setting and configuration options, as in the following example:

   ```
   [finalise] com.ibm.cics.bms.ids=true
   [finalise] com.ibm.cics.bms.ids.vtamignore=false
   ```
3. In the USSCONFIG directory of each CICS region that will use the configuration file created in Step 1, create a file with the name `groupfeaturetoggle.properties`, which is a symbolic link to the configuration file. By default, the path is `/var/cicsts/dfhconfig`.

   Note: The CICS installation process creates an empty `groupfeaturetoggle.properties` file in `/pathprefix/usr/lpp/cicsts/ussdira/dfhconfig`, which you can copy to the required USSCONFIG directory.

Tip: If you are not sure which path the CICS region is using, you can see the path in message DFHPA1951I in the system message log.
In the following example, CICS regions AOR1, AOR2, and AOR3 are using the following configuration file:

`/etc/cicsts/aorgroup/featuretoggle.properties`

To do so, you must specify a group-level configuration file, `groupfeaturetoggle.properties`, in the USSCONFIG directory of each region. `groupfeaturetoggle.properties` is a symbolic link to `featuretoggle.properties`, as shown in Figure 27.

```
AOR1     /var/cicsts/dfhconfig/AOR1/groupfeaturetoggle.properties
AOR2     /var/cicsts/dfhconfig/AOR2/groupfeaturetoggle.properties
AOR3     /var/cicsts/dfhconfig/AOR3/groupfeaturetoggle.properties
```

Figure 27. Symbolic links

To specify a symbolic link for AOR1 as shown in the preceding figure, enter the following command:

```
ln -s /etc/cicsts/aorgroup/featuretoggle.properties
      /var/cicsts/dfhconfig/AOR1/groupfeaturetoggle.properties
```

CICS startup

Depending on your system environment, you can start the CICS job from a procedure by using the **START** command, or you can submit the CICS startup job stream through the internal reader.

About this task

For an example of a batch job that you can submit through the internal reader, see "A sample CICS startup job" on page 129. "A sample CICS startup procedure" on page 139 gives an example of a cataloged procedure suitable for starting CICS as a started task.

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
- 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.

Data sets may also be opened for backout even after a successful shutdown, if they had suffered backout failures before the CICS shutdown, because failed backouts are retried at emergency restart.

- Opening BSAM 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. For example, backout can occur if you start CICS with START=AUTO and CICS detects that the previous shutdown was immediate or uncontrolled. With SDTRAN, an immediate shutdown does not always leave in-flight units of work to be backed out. Also, even if there were no in-flight UOWs, it is possible (although rare) that there were backout-failed UOWs for which backout will be retried.

For background information about backout, and recovery and restart, see Troubleshooting for recovery processing.

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.

The PLTPI resource managers start in the following sequence when specified in the system initialization table:

1. CPSMCONN=CMAS
2. CPSMCONN=LMAS
3. CPSMCONN=WUI
4. DBCTLCON=YES
5. DB2CON=YES
6. MQCON=YES

This sequence only applies to the above system initialization parameters and not for PLTPI programs in general.

For more information about the user ID under which PLT programs run, see PLT programs.

For programming information about writing PLT programs, see Writing initialization and shutdown programs.

If you are running CICS with DB2, you can specify the DB2 subsystem ID to be used at PLT startup by the **INITPARM** system initialization parameter, as follows:

```
INITPARM=(DFHD2INI='yyyy')
```

where yyyy is the 4-character DB2 subsystem ID. The value must conform to MVS JCL rules about special characters. You cannot use the ID of a data sharing group of DB2 subsystems on the **INITPARM** system initialization parameter — you must specify the ID of a single DB2 subsystem. If you want to use the **INITPARM** system initialization parameter to specify a DB2 subsystem, leave blank both the DB2GROUPID and the DB2ID in the installed DB2CONN definition. An ID specified in either of these attributes of the DB2CONN definition overrides an ID specified on the **INITPARM** system initialization parameter.

Setting address space storage limits for a CICS region

When you submit your CICS job, set the z/OS parameters **REGION** and **MEMLIMIT** to specify the amount of virtual storage for the address space in which CICS runs.

About this task

You cannot alter the **REGION** and **MEMLIMIT** values for the CICS region while CICS is running. You can specify new values only when you start the CICS region.

The **REGION** parameter specifies your request for an amount of 24-bit and 31-bit storage, that is, storage below the bar, for the CICS address space. Up to 2047 MB of storage can be requested, but you must leave enough storage below the bar for the system region in 24-bit storage, and items in the high private area such as the local system queue area (LSQA). You can specify the **REGION** parameter in different ways to request a specific amount of storage, or all the available 24-bit or 31-bit private storage. The resulting region size below the bar can be unpredictable.

The **MEMLIMIT** parameter specifies the limit of 64-bit (above-the-bar) storage for the CICS region. A CICS region needs a **MEMLIMIT** value of at least 10 GB. The default value in z/OS for **MEMLIMIT** is 2 GB.

Procedure

1. Calculate the amount of 24-bit (below-the-line) storage and 31-bit (above-the-line) storage to request on the **REGION** parameter. See Estimating and setting REGION.
2. Specify a suitable **REGION** parameter for the CICS region. You can set **REGION** in the following ways:
   - You can specify the **REGION** parameter in the JOB statement in the CICS JCL. In this situation, each step of the job runs in the requested amount of space.
   - You can specify the **REGION** parameter in the EXEC statement (program execution line) for CICS. In this situation, each step runs in its own amount of space. Use the EXEC statement instead of the JOB statement if different steps need greatly different amounts of space. For example, use the EXEC statement if you are using extra job steps to print auxiliary trace data sets after CICS has shut down (as in the DFHIVPOL installation verification procedure).
   - The z/OS installation exit IEFUSI can limit the **REGION** value that you specify. For information about IEFUSI, see IEFUSI - Step Initiation Exit in z/OS MVS Installation Exits.

   For further details about specifying the **REGION** parameter, and information about the amount of storage that z/OS allocates in response to your request, see REGION Parameter in the z/OS MVS JCL Reference.
3. Calculate the amount of 64-bit (above-the-bar) storage that you need to specify with the **MEMLIMIT** parameter. This amount depends on the facilities that you plan to use. See Estimating, checking, and setting MEMLIMIT in Improving performance.
4. Identify the **MEMLIMIT** value that applies to your CICS region, and change it if necessary. You can set **MEMLIMIT** in the following ways:
   - A **MEMLIMIT** value can be specified in the JOB statement in the CICS JCL, or in the EXEC statement (program execution line) for CICS.
   - If there is no **MEMLIMIT** value specific to the CICS region, the **MEMLIMIT** value that is set in the z/OS SMFPRM*xx* PARMLIB member, or the system default, applies.

- The z/OS installation exit IEFUSI can override any other **MEMLIMIT** values.

The following example shows a **MEMLIMIT** value set in the program execution line:

```
//CICS EXEC PGM=DFHSIP,PARM='SI',REGION=0M,MEMLIMIT=10G
```

For further details about changing a **MEMLIMIT** value in z/OS, or determining the value that applies to the CICS region that you are setting up, see Limiting the use of memory objects in the z/OS MVS Programming: Extended Addressability Guide.

Using the sample startup job stream

You can use the sample job control statements to initialize a CICS region.

About this task

The sample startup job stream is based on the system initialization parameters contained in the CICS-supplied sample table, DFHSIT6$.

For more information about the DD statements in this job stream that are needed by CICS and IMS, see the appropriate section in “Defining data sets” on page 23.

JCL similar to that in “A sample CICS startup job” is supplied as a sample startup procedure, DFHSTART, in the CICSTS54.CICS.SDFHINST library. You can use the DFHSTART procedure to start the CICS regions, or as a basis for your own startup procedures. For information about the DFHSTART procedure, see CICS startup procedure, DFHSTART in Installing.

A sample CICS startup job

You can use the sample CICS startup job to help you determine and adjust values needed for CICS startup parameters.

The EXEC statement contains the **REGION** parameter to define the size of CICS region. In this example, the value is set to 240, requesting MVS to allocate to the job all 16 MB of private storage below the 16 MB line and an extended region size of 240 MB.

```
/*
//*  1    The JOB statement
//CICSRUN  JOB accounting info,name,CLASS=A,
//            MSGCLASS=A,MSGLEVEL=(1,1),NOTIFY=userid
//*
//*  2    The JOBPARM statement
/*JOBPARM SYSAFF=sysid
/*
//*************************************************************
//******************  EXECUTE CICS  ***********************
//*************************************************************
//*
//*  3    The EXEC CICS=DFHSIP statement
//CICS    EXEC PGM=DFHSIP,REGION=240M,
//*  4    System initialization parameters specified on PARM parameter
//             PARM=('SIT=6$',
//             'DSALIM=6M,EDSALIM=120M',
//             'RENTPGM=PROTECT,STGPROT=YES',
//             'START=AUTO,SI')
//*
//*  5    System initialization parameters specified on the SYSIN data set
//SYSIN    DD *
GRPLIST=(DFHLIST,userlist1,userlist2),
LPA=YES,
APPLID=CICSHTH1,
DFLTUSER=CICSUSER,   The default user ID
MXT=30,              Maximum number of user tasks is 30
INITPARM=(DFHDBCON='01',DFHD2INI=('MYDB')),
                     Pass DFSPZP01 suffix to DBCTL connect program
                     Connect to DB2 subsystem MYDB
ISC=YES,             Include intersystem communication program
IRCSTRT=YES,         Start interregion communication
.END
/*
//*  6    The STEPLIB library
//STEPLIB  DD  DSN=CICSTS54.CPSM.SEYUAUTH,DISP=SHR
//         DD  DSN=CICSTS54.CICS.SDFHAUTH,DISP=SHR
//         DD  DSN=CICSTS54.CICS.SDFJAUTH,DISP=SHR
//         DD  DSN=CEE.SCEERUN2,DISP=SHR
//         DD  DSN=CEE.SCEERUN,DISP=SHR
//*
//*  7    The CICS library (DFHRPL) concatenation
//DFHRPL   DD DSN=CICSTS54.CICS.SDFHLOAD,DISP=SHR
//*        Your application library
//         DD  DSN=your.prog.library,DISP=SHR
//*        Your CICS control tables library
//         DD  DSN=your.table.library,DISP=SHR
//*        The Language Environment runtime data sets
//         DD  DSN=CEE.SCEECICS,DISP=SHR
//         DD  DSN=CEE.SCEERUN2,DISP=SHR
//         DD  DSN=CEE.SCEERUN,DISP=SHR
//*        The Debug Tool runtime library
//         DD  DSN=EQA.SEQAMOD,DISP=SHR
//*  7a    The DB2 load library
//         DD  DSN=SYS2.DB2.SDSNLOAD,DISP=SHR
//*  8    Auxiliary temporary storage data sets
//DFHTEMP  DD DSN=CICSTS54.CICS.CNTL.CICSHTH1.DFHTEMP,DISP=SHR
//*
//*  9    Intrapartition data sets
//DFHINTRA DD DSN=CICSTS54.CICS.CNTL.CICSHTH1.DFHINTRA,DISP=SHR
//*
//*  10   The auxiliary trace data sets
//DFHAUXT  DD DSN=CICSTS54.CICS.CICSHTH1.DFHAUXT,DISP=SHR
//DFHBUXT  DD DSN=CICSTS54.CICS.CICSHTH1.DFHBUXT,DISP=SHR
//*
```

Figure 28. CICS startup job stream

```
//* 11    Extrapartition data sets
//DFHCXRF  DD SYSOUT=*
//LOGUSR   DD SYSOUT=*,DCB=(DSORG=PS,RECFM=V,BLKSIZE=136)
//MSGUSR   DD SYSOUT=*,DCB=(DSORG=PS,RECFM=V,BLKSIZE=140)
//COUT     DD SYSOUT=*,DCB=(DSORG=PS,RECFM=V,BLKSIZE=137)
//CEEMSG   DD SYSOUT=A
//CEEOUT   DD SYSOUT=A
//*
//* 12    The CICS local catalog data set
//DFHLCD   DD DSN=CICSTS54.CICS.CICSHTH1.DFHLCD,DISP=SHR
//* 13    The CICS global catalog data set
//DFHGCD   DD DSN=CICSTS54.CICS.CICSHTH1.DFHGCD,DISP=SHR,
//            AMP=('BUFND=33,BUFNI=32,BUFSP=1114112')
//*
//* 14    The transient data destination data set (CXRF)
//DFHCXRF  DD SYSOUT=A
//*
//* 15    The CICS transaction dump data sets
//DFHDMPA  DD DSN=CICSTS54.CICS.CICSHTH1.DFHDMPA,DISP=SHR
//DFHDMPB  DD DSN=CICSTS54.CICS.CICSHTH1.DFHDMPB,DISP=SHR
//*
//* 16    MVS system dump data sets
//SYSABEND DD SYSOUT=*
//*
//* 16a    Default stdout file for C-language system services used by CICS
//SYSPRINT DD SYSOUT=*
//* 17    The CICS system definition data set
//DFHCSD   DD DSN=CICSTS54.CICS.DFHCSD,DISP=SHR
//*
//* 18    The CICS BTS local request queue data set
//DFHLRQ   DD DSN=CICSTS54.CICS.DFHLRQ,DISP=SHR
//*

//* 19    The CMAC data file
//DFHCMACD DD DSN=CICSTS54.CICS.DFHCMACD,DISP=SHR
//*
//* 20    The CDBM group command file
//DFHDBFK  DD DSN=CICSTS54.CICS.DFHDBFK,DISP=SHR
//*
//* 21    FILEA & other permanently allocated data sets
//* (The FILEA DD statement below overrides the CSD definition in
//*  group DFHMROFD)
//FILEA    DD DISP=SHR,
// DSN=CICSTS54.CICS.CICSHTH1.FILEA
//*
//APPLICA  DD  DSN=CICSTS54.CICS.CICSHTH1.APPLICA,DISP=SHR
//APPLICB  DD  DSN=CICSTS54.CICS.CICSHTH1.APPLICB,DISP=SHR
//*
//*
//* 22
//PRINTER  DD  SYSOUT=A,DCB=BLKSIZE=125,OUTLIM=0
//* User transactions input from sequential terminal
//CARDIN DD *
CESF GOODNIGHT\
/*
```

1. The JOB statement specifies the accounting information that you want to use for this run of CICS. For example:

   ```
   //CICSRUN  JOB 24116475,userid,MSGCLASS=A,MSGLEVEL=(1,1),
   //         CLASS=A,NOTIFY=userid
   ```

2. The JES JOBPARM statement is included to identify the MVS image on which this CICS (the active CICS region) is to run.

3. DFHSIP is the program that starts CICS initialization. CICS does not support more than one EXEC PGM=DFHSIP job step in the same MVS job.

   To determine how much of the allocated private storage CICS gives to dynamic storage areas and leaves for demands on operating system storage, set values for the **DSALIM** and **EDSALIM** system initialization parameters. After obtaining the amount of space required for the DSAs from the total defined by the **REGION** parameter, the remaining storage is available to meet demands for operating system storage.

   In this sample job stream, these system initialization parameters are specified in the **PARM** parameter.

   For more details about the **REGION** parameter and CICS storage, see "Setting address space storage limits for a CICS region" on page 128.

   If you are running CICS with RACF support, see the RACF security overview for information about parameters related to RACF.

4. 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** strings, and excludes the opening and closing parentheses delimiting the **PARM** parameter. Internal parentheses enclosing system initialization operands are included. If 100 characters are not sufficient for the system initialization parameters that you want to provide at startup, indicate continuation by ending the **PARM** field with the SYSIN or CONSOLE control keywords, or SI or CN for short. If you specify SYSIN, system initialization parameters are read from the SYSIN data set; if you specify CONSOLE, CICS prompts you to enter parameters through the console. However, if all of your runtime system initialization parameters are in the **PARM** parameter, you can end the PARM field without any control keywords, or by the END control keyword.

   In this example, CICS uses the values in the DFHSIT6$ system initialization table, modified by the system initialization parameters supplied in the PARM field and the SYSIN data set. For this example, the following system initialization parameters are provided in the **PARM** parameter:

   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. Specify RENTPGM=NOPROTECT for development CICS regions and RENTPGM=PROTECT for production CICS regions. For more information about the **RENTPGM** parameter, see RENTPGM system initialization parameter.

   STGPROT
   : Storage protection is obtained for this run of CICS by specifying YES for the **STGPROT** system initialization parameter, or allowing it to default to YES. See STGPROT system initialization parameter for details about the **STGPROT** parameter.

   START
   : START=AUTO is normally the type of start that you select for a production CICS system, so that CICS can determine the type of start to be performed.

   SI The PARM statement ends with SI (short for SYSIN), to tell CICS to continue reading overrides from the SYSIN data set.

5.

You can include the SYSIN data set inline as part of the job stream. System initialization parameters entered in the SYSIN data set replace any that were entered in the **PARM** parameter for the same keyword. If you include the same parameter more than once, the *last* value read is the value used for initialization except for INITPARM. If you specify the INITPARM keyword and its parameters more than once, each one is accepted by CICS, for example:

```
* The following INITPARM parameters are for DBCTL and a user program
INITPARM=(DFHDBCON='XX,DBCON2',userprog='a,b,c')
* The following INITPARM parameter is for DB2
INITPARM=(DFHD2INI= 'DBA2')
```

Unless you explicitly code the system initialization control keyword CONSOLE, CICS stops reading system initialization parameters when it reaches the end of SYSIN or the END control keyword.

In the sample job, CONSOLE is not coded in either PARM or SYSIN. The END control keyword is the last entry in SYSIN, so CICS does not prompt through the console for further system initialization parameters. After reading the SYSIN data set, CICS loads the specified SIT, applies any system initialization parameters supplied in the PARM field and the SYSIN data set, and begins the initialization process.

The SYSIN data set in this example includes several system initialization parameters:

GRPLIST
: The group list defined in DFHSIT6$ is DFHLIST, the list that is generated when you initialize the CSD using the **DFHCSDUP INITIALIZE** command. DFHLIST contains only the standard 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 either include the resource definitions in one of your own group lists or specify the DFHLIST explicitly. Your own group lists (userlist1 and userlist2 in the example) must contain all the resource definitions generated by your installation for your CICS applications. In addition, your group lists must contain any definitions required for IBM licensed programs that you are using, such as COBOL or DB2.

LPA
: The SIT specifies that modules are not to be used from the link pack area; LPA=YES in SYSIN specifies that modules are to be used from the LPA in this run.

APPLID
: The APPLID for this CICS region is CICSHTH1. If you want this CICS region to use z/OS Communications Server for terminal access or ISC communication, define this APPLID to z/OS Communications Server.

  CICSID is the generic APPLID of this CICS region, and CICSHTH1 is the specific APPLID of the active CICS region.

  The specific APPLID can be useful for naming those data sets that are unique; for example, dump data sets.

CICSSVC
: 245 is the CICS type 3 SVC number installed in the LPA, and defined to MVS in an SVCPARM statement. For more information, see CICSSVC system initialization parameter.

  For guidance information about installing the CICS SVC in the LPA, and defining it to MVS, see Installing the CICS SVCs in Installing.

DFLTUSER
CICSUSER is the default user ID specified to RACF. During startup, CICS tries to sign on the default user ID. If it cannot be signed on (for example, if not defined), CICS issues a message and stops CICS initialization. After the valid default user ID 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 user ID is defined to RACF 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 30 for this run. For information about the tasks that are included in the **MXT** parameter, see MXT system initialization parameter.

INITPARM
Passes parameters to programs. In this example, the DBCTL suffix (01) of DFSPZPxx is passed to DFHDBCON, and CICS is told to connect to the DB2 subsystem MYDB. You cannot use the ID of a data sharing group of DB2 subsystems on the **INITPARM** system initialization parameter; you must specify the ID of a single DB2 subsystem. If you want to use the **INITPARM** system initialization parameter to specify a DB2 subsystem, leave blank both the DB2GROUPID and the DB2ID in the installed DB2CONN definition. An ID specified in either of these attributes of the DB2CONN definition overrides an ID specified on the **INITPARM** system initialization parameter.

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.

6. STEPLIB is the DDNAME of the library containing the modules loaded by the operating system. DFHSIP, which is loaded from STEPLIB, must receive control in an authorized state, so each partitioned data set (library) concatenated in STEPLIB must be individually APF-authorized. In this sample job stream, the CICS authorized library is CICSTS54.CICS.SDFHAUTH. CICSTS54.CICS.SDFJAUTH is also included, which is required for Java support, and must also be APF-authorized. If you do not require Java support, do not include this library. CICSTS54.CPSM.SEYUAUTH is also included in the concatentation so that CICS will load the CPSM message file at initialization even if the region is not running as a Managed Address Space.

   Placing user-written modules into an APF-authorized library might violate your security and integrity rules. The CICSTS54.CICS.SDFHAUTH library (and the CICSTS54.CICS.SDFJAUTH library, if used) must not be included in the MVS link list, so that you can protect the libraries and ensure that only approved users can run the DFHSIP module. When you define your STEPLIB DD statement, remember that all other libraries concatenated with the CICSTS54.CICS.SDFHAUTH library or the CICSTS54.CICS.SDFJAUTH library must also be APF-authorized; for example, IMS.RESLIB. APF-authorization is required because, if any of the libraries in a STEPLIB concatenation are not authorized, MVS regards all of them as unauthorized. If there is a non-authorized library in the STEPLIB concatenation, CICS fails to start, and issues message DFHKE0101 to indicate that the DFHSIP module is not in an APF-authorized library. For information about authorizing access to CICS data sets, see RACF data set profiles.

The pregenerated DFHSIP module, which has been link-edited with the authorized attribute (SETCODE AC(1)), is supplied in CICSTS54.CICS.SDFHAUTH.

You must put the Language Environment runtime libraries, CEE.SCEERUN and CEE.SCEERUN2, in the STEPLIB concatenation, or the MVS link list, to run COBOL, PL/I, C and C++, and Java programs that run in a JVM, under Language Environment. Like SDFHAUTH, SCEERUN and SCEERUN2 must be an APF-authorized library.

The CICS DB2 attachment facility must load the DB2 program request handler, DSNAPRH. You can either place the DB2 library DSNxxx.SDSNLOAD library in the MVS link list or add it to the STEPLIB concatenation of the CICS job.

7. DFHRPL is the DD name of the library that contains modules loaded by CICS. Protect individually the partitioned data sets constituting this library to prevent unapproved or accidental modification of their contents. The DFHRPL concatenation must include the library containing your CICS application programs, shown in this example as `your.prog.library`, and your CICS control tables, shown in this example as `your.table.library`.

   In this sample job stream, the supplied library is CICSTS54.CICS.SDFHLOAD, which must be included in the CICS DFHRPL library concatenation. The CICSTS54.CICS.SDFHLOAD library contains only programs that run in problem state and must not be authorized.

   Language Environment runtime library requirements

   Use the services of Language Environment, which provides a common runtime environment for IBM implementations of assembler and those high-level languages (HLLs) supported by CICS, namely COBOL, PL/I, C, and C++.

   To use the Language Environment runtime libraries to support all your program languages, add the Language Environment runtime libraries SCEERUN2 and SCEERUN to the DFHRPL DD concatenation. If you are running COBOL programs, also add Language Environment runtime library SCEECICS to DFHRPL. The SCEECICS library must be concatenated before the SCEERUN library. Remove any libraries that contain runtime routines from earlier versions of COBOL, PL/I, and C/C++ from the DFHRPL DD concatenation.

   Debug Tool library

   If you are using Debug Tool to debug CICS applications, include the Debug Tool library SEQAMOD in DFHRPL. For information about using Debug Tool with CICS, see Using Debug Tool with CICS applications.

   7a **DB2 requirements**

   Generally, you do not have to include any DB2 libraries in the DFHRPL DD statement. If you do require DB2 libraries in the DFHRPL concatenation for an application, they must be placed after the CICS libraries; for example, add SDSNLOAD in the DFHRPL to support those applications that issue dynamic calls to the DB2 message handling module, DSNTIAR, or the later DSNTIA1, both of which are shipped in SDSNLOAD. DSNTIA1 is loaded by applications programs that include the DB2 application stub DSNTIAC, which issues an **`EXEC CICS LOAD`** command for program DSNTIAC.

8. 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 "Setting up temporary storage data sets" on page 31.

If you are using temporary storage data sharing, ensure that you start the temporary storage server before it is required by the CICS regions.

For more information about the temporary storage server and temporary storage data sharing, see "Setting up and running a temporary storage server" on page 152

9. The transient data intrapartition data set is used for queuing messages and data in the CICS region. For information about how to define these data sets, and about space calculations, see "Defining the intrapartition data set" on page 34.
10. 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 DD name must be DFHAUXT.

    For details of how to define these data sets, see "Setting up auxiliary trace data sets" on page 60.

    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 allocate and catalog the auxiliary trace data sets on disk as shown in Figure 15 on page 62, you can define them to CICS in the startup job stream using the following DD statements:

    ```
    //DFHAUXT  DD  DSN=CICSTS54.CICS.applid.DFHAUXT,DCB=BUFNO=n,DISP=SHR
    //DFHBUXT  DD  DSN=CICSTS54.CICS.applid.DFHBUXT,DCB=BUFNO=n,DISP=SHR
    ```

    If you specify BUFNO greater than 1, you can reduce I/O involved in writing auxiliary trace records. A value in the range 4 - 10 can greatly reduce I/O when running with auxiliary trace on.
11. LOGA and CSSL are examples of extrapartition transient data queues.
    - LOGA defines a user data set used by the CICS sample programs.
    - CSSL defines the data set used by a number of CICS services.
    - CCSO is used as an output queue only when you are running C/370™ application programs.
    - CESO is used as an error queue only when you are running application programs under Language Environment.

      Sample definitions of the queues used by CICS are supplied in group DFHDCTG. DFHDCTG is unlocked, so you can alter the definitions before installation.
12. 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. For details of how to create and initialize a CICS local catalog, see "Setting up the catalog data sets" on page 50.
13. There is only one global catalog.

    For details of how to create and initialize a CICS global catalog, see "Setting up the catalog data sets" on page 50.

    This sample job illustrates the use of the **AMP** parameter on the DD statement. Specifying this parameter, with its buffer sub-parameters, can help to improve restart and shutdown time. This example uses DEFINE CLUSTER statements that are shown in Figure 12 on page 52 and the associated notes given under 4

in "Defining the global catalog" on page 50. The values given are the minimum values suitable for these parameters and must not be reduced.

14. 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.
15. CICS records transaction dumps on a sequential data set or a pair of sequential data sets, tape, or disk. The data sets must be defined with the DD names DFHDMPA and DFHDMPB, but if you define only one data set, its DD name must be DFHDMPA. CICS always attempts to open at least one transaction dump data set during initialization.

    For details about how to define CICS transaction dump data sets and how they are used, see "Defining dump data sets" on page 62.

    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 (DBDCCIC1) as a second-level qualifier.
16. Use a SYSABEND DD statement to direct MVS to produce a dump.

    Produces a dump of user and system areas; this dump contains all the areas dumped in a SYSUDUMP plus the local system queue area (LSQA), including subpools 229 and 230, and the I/O system (IOS) control blocks for the failing task. The dump is formatted, so that it can be printed directly. MVS produces the requested dump if either of the following is true:

    - CICS ends abnormally.
    - CICS starts to end abnormally, but system recovery procedures enable CICS to end normally.

    The dump contents are as described only when you use the IBM-supplied defaults for the dumps. The contents of these dumps can be set during MVS system initialization and can be changed for an individual dump in the ABEND macro instruction, in a CHNGDUMP command, and by a SLIP command. For details, see the z/OS MVS Initialization and Tuning Guide in IBM Knowledge Center.

    Dumps are optional; use a dump DD statement only when you want to produce a dump.

    For information about how defining MVS system dump data sets, and about printing dumps from them, see the z/OS MVS JCL Reference in IBM Knowledge Center. For information about how to interpret dumps, see z/OS MVS Diagnosis: Tools and Service Aids.

    16a Default stdout file for C-language system services used by CICS

    The SYSPRINT statement is opened as the `stdout` file by C-language system service routines that are used by CICS, such as system SSL and JVM servers. If it is omitted, multiple `sysout` files might be dynamically allocated by the operating system instead.
17. The system definition file (CSD) is required by CICS to hold some resource definitions.

    You might want to provide job control DD statements for the CSD. If you do, the CSD data set is allocated at the time of CICS job step initiation, and *remains allocated for the duration of the CICS job step*.

    You might prefer to use dynamic allocation of the CSD. For dynamic allocation, do not specify a DD statement for the CSD. Specify the data set name (DSNAME) and the data set disposition (DISP) either in a **SET FILE** command or in the System Initialization (as parameters CSDDSN and CSDDISP). CICS uses the DSNAME and DISP to allocate the file as part of OPEN processing.

For information about creating and initializing the CSD, see Chapter 1, "Setting up shared data sets, CSD and SYSIN," on page 1.

18. DFHLRQ is a file-control-managed VSAM key-sequenced data set (KSDS), used by CICS business transaction services (BTS). The supplied FILE resource definition for DFHLRQ is in CSD group DFHCBTS, which is automatically included in DFHLIST group list when you initialize or upgrade your CSD. The FILE definition specifies that it is to be opened on first reference, which occurs at the end of CICS initialization when it is opened by BTS. CICS issues warning messages DFHFC0951 and DFHSH0109 if the data set is not found. Although CICS continues running without this data set, define the DFHLRQ data set, even if you are not using BTS. For information about DFHLRQ, see Overview of BTS.
19. DFHCMACD is a VSAM key-sequenced data set (KSDS) that is used by the CMAC transaction to provide online descriptions of CICS messages and codes. Before its first use, it must be defined and loaded as a KSDS data set. "Defining the CMAC messages data set" on page 81 describes DFHCMACD in greater detail.
20. DFHDBFK is a VSAM key-sequenced data set (KSDS) that is used by the CDBM transaction to store Group commands. Before its first use, it must be defined as a KSDS data set. The DFHDBFK DD statement is required only if you intend to use the command storage functions of the CDBM transaction. Defining the CDBM GROUP command data set describes DFHDBFK in greater detail.
21. You might want to provide job control DD statements for those user files that are defined in the CSD (if you are using RDO) or in a file control table (for BDAM files only). 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 this startup job stream as an example of direct allocation by job control statement.

    Alternatively, you might prefer to take advantage of the CICS dynamic allocation of files. For dynamic allocation, do not specify a DD statement for the CSD. CICS then uses the full data set name as specified in the **DSNAME** parameter of the FILE resource definition (up to 44 characters), together with the **DISP** parameter, 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 file opening, see "Defining user files" on page 67. For information about the attributes that you can define on FILE resources, see FILE resources.

    The card reader/line printer (CRLP) simulated terminals shown in the sample job stream are defined in the sample TCT (not used in this startup job). See the copy member DFH$TCTS, in CICSTS54.CICS.SDFHSAMP, for the source statements that you need for such devices.
22. 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. So the terminal is still in RECEIVE status in a subsequent warm start, and CICS does not then read the input file.

    Note the end-of-data character (the "\" symbol) at the end of each line of the sample.

A sample CICS startup procedure

As an alternative to submitting a batch job to the MVS internal reader, you can use the MVS **START** command to start CICS as a started task. Using this method, your startup job stream must be coded according to the rules for coding procedures, and the procedure must be installed in an MVS procedure library.

About this task

If you intend to start CICS with the **START** command you must either:

- Give the MVS started task procedure a name different from the subsystem name in IEFSSNaa (default 'CICS'), or
- Issue the start command with the parameter SUB=JES2 or SUB=JES3 as appropriate.

You can use the following form of the MVS **START** command to start a job from the console:

```
S|START procname[.identifier][,SUB=subsystemname][,keyword=option
 [,keyword=option] . . .]
```

where:

procname
The name of the cataloged procedure that defines the job to be started.

identifier
The name you choose to identify the task.

SUB=subystemname
The name of the subsystem that is to select the job for processing. If you omit this parameter, the primary job entry subsystem is used.

keyword=option
Any appropriate keyword to override the corresponding parameter in the procedure. You can use this parameter to override symbolic parameters defined in the cataloged procedure.

For guidance information about the complete syntax of the START command, and all the keywords and options you can use, see *z/OS MVS System Commands*.

To start CICS, you only need to code **procname.identifier,keyword(s)=option**.

For example:

```
START DFHSTART.CICSA,SIP=T,REGNAME1=IDA,REGNAM2=IDA
```

In this example of the MVS **START** command:

- DFHSTART is the name of the CICS-supplied cataloged startup procedure.
- CICS is being started with a task ID of CICSA.
- SIP is the suffix of the DFH$SIPx member in the SYSIN data set, CICSTS54.CICS.SYSIN, to be used by this CICS region.
- REGNAM1 and REGNAM2 are qualifiers added to the CICS system data sets specified in the procedure (for example, CICSTS54.CICS.CICSHTH1.DFHTEMP) to identify uniquely the data sets for this CICS region. REGNAM1 is set to the same value as REGNAM2 for an MRO region.

For information about the DFHSTART procedure, see the CICS startup procedure, DFHSTART in Installing.

If you are running CICS with RACF, you must associate the cataloged procedure name with a suitably authorized RACF user through the RACF table, ICHRIN03. For information about this association, see Authorizing CICS procedures to run under RACF .

Preparing CICS for using debugging tools

Before application programmers can use certain debugging tools with CICS, you will need to perform some system definition tasks. This section describes those tasks.

Preparing your CICS region for debugging

Before application programmers can use certain debugging tools with CICS, you must configure your CICS region accordingly.

About this task

You can prepare your CICS region for debugging application programs using the following tools:

- Debug Tool, for compiled language application programs (programs written in COBOL, PL/I, C, C++), and for Language Environment-enabled Assembler subroutines).
- Remote debugging tools (for compiled language application programs, Language Environment-enabled Assembler Subroutines, and Java programs). Note that for compiled language programs, and Assembler subroutines, Debug Tool is used as the debugging server.

This topic does not apply to other debugging tools, such as the CICS Execution Diagnostic Facility (CEDF).

Since you will need to restart the region, it is advisable to plan which regions will be used for debugging applications.

- It is unlikely that application programs will be debugged in your production regions, especially if the applications are well established and known to be reliable; it is more likely that debugging will take place in regions which are used to develop and test new applications.

To prepare your CICS region for debugging:

Procedure

1. If you plan to use the region for debugging compiled language programs, include the Debug Tool library `SEQAMOD` in the `DFHRPL` concatenation in your CICS startup JCL. For more information, see “Using the sample startup job stream” on page 129.
2. Create the *debugging profile data sets*. For more information, see “Setting up the debugging profiles data sets” on page 84.
3. Include the resource definition for the debugging profile file in a resource definition list that is named in the GRPLIST system initialization parameter.
4. Optionally, specify the following value for the DEBUGTOOL system initialization parameter:

   ```
   DEBUGTOOL=YES
   ```

   If you do not specify `DEBUGTOOL=YES`, you can enable the region for debugging when it is running:

- To enable the region for debugging from a program, use the **EXEC CICS SET SYSTEM DEBUGTOOL** command
- To enable the region for debugging from the master terminal transaction, use the **CEMT SET SYSTEM DEBUGTOOL** command

Enabling the region for debugging when it is running is recommended for regions which are not normally used for debugging. When debugging is complete, you can disable the region for debugging, using the same commands.

5. Define and install Debug Tool's resource definitions. They are located in member EQACCSD in Debug Tool's SEQASAMP data set. For more information, see the *Debug Tool for z/OS and OS/390 User's Guide*.

Starting CICS regions

This section describes how to start CICS regions. It assumes that you have already carried out any customization of CICS, generated any additional support required, and defined all the necessary CICS system definitions.

About this task

You can start CICS in one of two ways:

- Use the MVS **START** command to start CICS as a started task.
- Submit a CICS batch job to the MVS internal reader.

In both methods you determine how CICS starts, and the facilities and resources that it can use. You do this by specifying values for system initialization parameters that are used by the CICS startup procedure. You would normally specify the system initialization parameters before starting CICS. However, after you have started the initialization of CICS, you can override the system initialization parameters specified before startup; for example, to enable a specific facility for that run of CICS.

Results

When system initialization is completed, the message DFHSI1517 is returned, indicating that control is given to CICS and that CICS is ready to process terminal requests.

Tip:

System warm-up by use of z/OS Workload Manager health service

Consider allowing CICS to have a warm-up process after the end of system initialization.

Even though DFHSI1517 signals the end of system initialization, the region might not be fully ready to receive work. This is because in some cases initialization continues after this message. For example, some bundle-defined resources are being installed asynchronously; so resources such as JVMSERVERs required for some applications have not completed initialization, even though the TCP/IP listener is open and accepting work. This could cause transactions to abend. For web services, a pipeline scan might not have completed, so use of the web service before the scan completes will fail.

You can allow CICS to have a warm-up process by setting the **WLMHEALTH** system initialization parameter. If the z/OS Workload Manager health service is active in the region, when the message DFHSI1517 is returned,

the warm-up process is started. Then CICS intermittently adjusts the region's z/OS WLM health indicator to control flow of work into the region until the region is fully capable to process work. For more information about the CICS warm-up process, see CICS warm-up and cool-down by use of z/OS Workload Manager health service.

Specifying system initialization parameters before startup

You usually specify the system initialization parameters that CICS is to use in the following ways, before starting CICS.

Procedure

- In the system initialization table, loaded from a library in the STEPLIB concatenation of the CICS startup procedure
- In the **PARM** parameter of the EXEC PGM=DFHSIP statement of the CICS startup procedure
- In the SYSIN data set defined in the startup procedure, but only if SYSIN is coded in the **PARM** parameter.

Results

The system initialization parameters are processed in the order outlined above, with later system initialization parameter values overriding those specified earlier.

Example

For example, if your CICS startup procedure specifies:

```
//INITCICS  EXEC PGM=DFHSIP,REGION=&REG,
//        PARM=('SYSIDNT=HTH1,SIT=6$,SYSIN,CN')
//*
//SYSIN    DD DISP=SHR,DSN=&libpfx..CICSH###.SYSIN(CICS&CLONE)
```

CICS uses system initialization parameters from the following sources, with later system initialization parameters overriding earlier ones:

1. The system initialization table, DFHSIT6$, from the STEPLIB concatenation.
2. The member CICSH### of the CICSTS54.CICS.CICSH###.SYSIN data set.
3. The system console.

What to do next

In particular, you can specify a new value for the **START** system initialization parameter.

The START system initialization parameter

The **START** system initialization parameter has the following values.

START=AUTO

If you specify this value for the **START** system initialization parameter, CICS determines whether to perform an initial, cold, warm, or emergency start by inspecting two records in the global catalog:

- The recovery manager control record
- The recovery manager autostart override record.

START=AUTO should be the normal mode of operation, with the choice of start being made by CICS automatically.

START=INITIAL
The new run of CICS has no reference to any previous run. The global catalog and system log are initialized, and all information in them is lost.

START=COLD
The new run of CICS has limited reference to the previous run, and uses the same global catalog and system log. In particular, resynchronization information needed by remote systems to resynchronize their units of work is preserved.

START=STANDBY
CICS starts up as an XRF alternate CICS region, by initializing only to the point at which it can monitor the active CICS region. Depending on how the active CICS region was shut down, the alternate CICS region completes either a warm or emergency restart, if it needs to take over, as follows:

- If the active CICS region was shut down with a successfully completed **CEMT PERFORM SHUTDOWN TAKEOVER** command, the alternate CICS region performs a **warm** start.
- If the active CICS region was shut down abnormally, the alternate CICS region performs an **emergency** restart.

Note: You must also specify the XRF=YES system initialization parameter.

Starting CICS as a batch job

Procedure

1. Create a job to start CICS. In your job you can either:
   - Include the CICS startup procedure inline.
   - Invoke a cataloged startup procedure.

   This latter method has the advantage that several CICS startup jobs (for example, for different CICS regions) can use the same procedure, tailoring the procedure through startup parameters. For example, Figure 29 shows a CICS startup job that invokes the cataloged procedure, CICSTASK, to perform a cold start with the startup parameters SYSIDNT=HTH1 and CLONE=HT##1. By altering the SYSIDNT and CLONE parameters, the same job could be used to start other CICS regions with the same procedure.

```
//CIDCTOR  JOB (accounting information),userid,MSGCLASS=A,MSGLEVEL=(1,1),
//         CLASS=C,NOTIFY=userid
//*********************************************************************
//* THIS JOB CAN BE USED TO START UP A CICS REGION
//*********************************************************************
//*
// CICS710 EXEC CICSTASK,
// START=COLD,
// SYSIDNT='HTH1',            SYSID OF CICS REGION
// CLONE='HT##'               CLONE CICS REGION TYPE
//*
```

Figure 29. Job to start a CICS TOR, HTH1

In this example of the MVS **START** command:

- CICSTASK is the name of a cataloged CICS startup procedure, tailored from the CICS-supplied sample startup procedure.
- SYSIDNT is the qualifier used to identify CICS system data sets that are unique to each CICS region.

- CLONE is the qualifier of the member in the SYSIN data set, CICSTS54.CICS.SYSIN, that has system initialization parameters unique to each *type of* CICS region.

2. Submit the job through the MVS internal reader. This starts CICS as a batch job.

Starting CICS as a started task

About this task

To start CICS as a started task:

Procedure

1. Create a procedure and install it in the MVS procedure library. Ensure that your startup job stream is coded according to the rules for coding procedures. You can use the CICS-supplied sample startup procedure DFHSTART and tailor it to suit your needs. For more information about the CICS-supplied startup procedure, see CICS startup procedure, DFHSTART in Installing .
2. To start CICS, you only need to code **procname.identifier,keyword(s)=option** in the procedure
3. You must do either of the following:
   - Give the MVS started task procedure a name different from the subsystem name in IEFSSNaa. The default is 'CICS'.
   - Issue the **START** command with the parameter **SUB** using the value JES2 or JES3 as appropriate.
4. Optional: If you are running CICS with RACF, associate the cataloged procedure name with a suitably authorized RACF user through the RACF table, ICHRIN03. For details about this association, see Authorizing CICS procedures to run under RACF .
5. To start CICS, use the MVS **START** command. For example, to start CICS from the MVS console:

   ```
   S|START procname[.identifier][,SUB=subsystemname][,keyword=option
   [,keyword=option] . . .]
   ```

   where

   procname
   The name of the cataloged procedure that defines the CICS job to be started.

   identifier
   The name you choose to identify the CICS task.

   subystemname
   The name of the subsystem that is to select the job for processing. If you omit this parameter, the primary job entry subsystem is used.

   option
   Any appropriate keyword to override the corresponding parameter in the procedure. You can use this parameter to override symbolic parameters defined in the cataloged procedure.

Example

For example, you could use the following start command to start the CICS tasks listed in Figure 30 on page 145:

```
START CICS710
```

```
//CICS710 PROC
//*
//DUMMY EXEC PGM=IEFBR14
//*
// START CICSTASK.CICSHTH1,SYSIDNT='HTH1',CLONE='HT##'
//*      START=COLD
// START CICSTASK.CICSHAH1,SYSIDNT='HAH1',CLONE='HA##'
//*      START=COLD
// START CICSTASK.CICSHAH2,SYSIDNT='HAH2',CLONE='HA##'
//*      START=COLD
// START CICSTASK.CICSHRH1,SYSIDNT='HRH1',CLONE='HR##'
//*      START=COLD
//*
//* END OF CICS START PROCEDURE
```

Figure 30. Procedure to start a CICS TOR, two AORs, and an ROR

In this example of the MVS **START** command:

- CICSTASK is the name of a cataloged CICS startup procedure, tailored from the CICS-supplied sample startup procedure.
- The following CICS regions are started:
  - Terminal-owning region, CICSHTH1
  - Application-owning region, CICSHAH1
  - Application-owning region, CICSHAH2
  - Resource-owning region, CICSHRH2.
- SYSIDNT is the qualifier used to identify CICS system data sets that are unique to each CICS region.
- CLONE is the qualifier of the member in the SYSIN data set, CICSTS54.CICS.SYSIN, that has system initialization parameters unique to each type of CICS region.

For information about the complete syntax of the START command, and all the keywords and options you can use, see the *z/OS MVS System Commands*.

Overriding system initialization parameters during startup

After you have started the initialization of CICS, you might want to override system initialization parameters specified in the SIT, PARM parameter, and SYSIN data set of the CICS startup procedure. You can do this by specifying new values for system initialization parameters at the system console.

Before you begin

You can specify system initialization parameters at the system console only if the CONSOLE keyword was specified in either the PARM parameter or in the SYSIN data set. If you specify the CONSOLE (or CN) keyword in the PARM statement of the EXEC PGM=DFHSIP statement or SYSIN data set of your CICS startup procedure, CICS prompts you to enter system initialization parameters at the system console.

About this task

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 SYSIN data set.

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'.
```

Procedure

1. Specify a SIT system initialization parameter only as the first parameter when prompted by message DFHPA1921. At this point CICS tries to load the specified SIT. If you try to specify a SIT system initialization parameter after CICS has loaded the SIT, it is rejected as an error.
2. 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 .
3. When you have changed the appropriate initialization parameter, terminate the console input by entering the END control keyword.

Entering corrections to parameters at the console

About this task

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 finds in the PARM parameter or the SYSIN 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 SYSIN data set, it is the APPLID coded in the SIT that CICS displays in messages DFHPA1912 and DFHPA1915.

System console messages for CICS startup

The following example shows the message sequences that are displayed when starting a CICS region, CICSHT61, using a typical startup procedure.

The CICS supplied default system initialization table, DFHSIT, is used (the SIT system initialization parameter is not specified) and system initialization parameters specific to the CICSHT61 region are supplied from a permanent SYSIN data set. You can see these options in the example messages, because the SYSIN data is displayed on the console.

The notes after the example identify specific messages of interest.

```
           J E S 2  J O B  L O G  --  S Y S T E M  M V 2 6  --  N O D E  W I N M V S 2 C
         STC07394
         ↓
09.40.21 .  IEF695I START CICSTASK WITH JOBNAME CICSHT61 IS ASSIGNED TO USER CICSHT##, GROUP TSOUSER
09.40.21 .  $HASP373 CICSHT61 STARTED
09.40.21 .  IEF403I CICSHT61 - STARTED - TIME=09.40.21
09.40.23 .  DFHPA1101  CICSHT61 DFHSIT   IS BEING LOADED.
09.40.23 .  DFHPA1108  CICSHT61 DFHSIT   HAS BEEN LOADED. (GENERATED AT: MM/DD= 10/28 HH:MM= 17:52).
09.40.23 .  DFHPA1100  CICSHT61 OVERRIDE PARAMETERS FROM JCL EXEC STATEMENT:
09.40.23 .  DFHPA1927  CICSHT61 START=AUTO,APPLID=CICSHT61,SYSIDNT=HT61,SYSIN
09.40.23 .  DFHPA1102  CICSHT61 OVERRIDE PARAMETERS FROM SYSIN:
09.40.23 .  DFHPA1927  CICSHT61 * APPLID=is provided in the JCL PARM parameter
09.40.23 .  DFHPA1927  CICSHT61 AIEXIT=DFHZATDY          Autoinstall program for terminals/APPC
09.40.23 .  DFHPA1927  CICSHT61 AICONS=AUTO              Autoinstall for consoles
09.40.23 .  DFHPA1927  CICSHT61 CICSSVC=221              The default CICS SVC number 1
09.40.23 .  DFHPA1927  CICSHT61 CMDPROT=NO
09.40.23 .  DFHPA1927  CICSHT61 CSDRLS=NO                Use CSD in RLS-mode
09.40.23 .  DFHPA1927  CICSHT61 DFLTUSER=BELL            CICS default userid 2
09.40.23 .  DFHPA1927  CICSHT61 DSALIM=7M
09.40.23 .  DFHPA1927  CICSHT61 EDSALIM=800M
09.40.23 .  DFHPA1927  CICSHT61 *GRPLIST=DFHLIST
09.40.23 .  DFHPA1927  CICSHT61 GRPLIST=(DFHLIST,NBLISTT,MQSLIST) Initialize with group lists for TOR 3
09.40.23 .  DFHPA1927  CICSHT61 GMTEXT='This is a CICS ID Test Region'  4
09.40.23 .  DFHPA1927  CICSHT61 * The IRC & ISC parameters required for MRO
09.40.23 .  DFHPA1927  CICSHT61 IRCSTRT=YES              Start interregion communication 5
09.40.23 .  DFHPA1927  CICSHT61 ISC=YES                  Include the intersystem communication progrm 5
09.40.23 .  DFHPA1927  CICSHT61 *
09.40.23 .  DFHPA1927  CICSHT61 KEYRING=Keyring.Name
09.40.23 .  DFHPA1927  CICSHT61 MAXSOCKETS=300           Maximum number of IP sockets allowed
09.40.23 .  DFHPA1927  CICSHT61 MSGLVL=1
09.40.23 .  DFHPA1927  CICSHT61 MXT=500                  Set maximum tasks to 500
09.40.23 .  DFHPA1927  CICSHT61 PGAIPGM=ACTIVE           Activate program autoinstall 6
09.40.23 .  DFHPA1927  CICSHT61 *PLTPI=NO                 No PLT for this run
09.40.23 .  DFHPA1927  CICSHT61 PLTPI=N2                 PLT for start-up programs
09.40.23 .  DFHPA1927  CICSHT61 RLS=YES                  RLS support required
09.40.23 .  DFHPA1927  CICSHT61 SEC=NO                   Run without security
09.40.23 .  DFHPA1927  CICSHT61 SPOOL=YES                JES spool support -- required for DFH0STAT
09.40.23 .  DFHPA1927  CICSHT61 * SYSIDNT is provided in the JCL PARM parameter
09.40.23 .  DFHPA1927  CICSHT61 STGPROT=YES              Storage protection on
09.40.23 .  DFHPA1927  CICSHT61 SYDUMAX=2                Limit the number of system dumps per dump code
09.40.23 .  DFHPA1927  CICSHT61 TRANISO=YES              Transaction isolation on
09.40.23 .  DFHPA1927  CICSHT61 TCPIP=YES                Initialize with CICS TCPIP Sockets Support
09.40.23 .  DFHPA1927  CICSHT61 TCT=5$                   Dummy TCT for autoinstalled VTAM terminals
09.40.23 .  DFHPA1927  CICSHT61 * TS=(,9,6)
09.40.23 .  DFHPA1927  CICSHT61 TRTABSZ=512              Internal trace table
09.40.23 .  DFHPA1927  CICSHT61 .END
09.40.23 .  DFHPA1103  CICSHT61 END OF FILE ON SYSIN.
09.40.24 .  +DFHTR0103 TRACE TABLE SIZE IS 512K
09.40.24 .  +DFHSM0122I CICSHT61 Limit of DSA storage below 16MB is 7,168K. 7
09.40.24 .  +DFHSM0123I CICSHT61 Limit of DSA storage above 16MB is 256M. 7
09.40.24 .  +DFHSM0115I CICSHT61 Storage protection is active. 8
09.40.24 .  +DFHSM0125I CICSHT61 Transaction isolation is active.
09.40.24 .  +DFHDM0101I CICSHT61 CICS is initializing.
09.40.26 .  +DFHWB0109I CICSHT61 Web domain initialization has started.
09.40.26 .  +DFHSO0100I CICSHT61 Sockets domain initialization has started.
09.40.26 .  +DFHRX0100I CICSHT61 RX domain initialization has started.
09.40.26 .  +DFHRX0101I CICSHT61 RX domain initialization has ended.
09.40.26 .  +DFHLG0101I CICSHT61 Log manager domain initialization has started.
09.40.26 .  +DFHEJ0101 CICSHT61
                 Enterprise Java domain initialization has started. Java is a
                 trademark of Oracle
09.40.26 .  +DFHDH0100I CICSHT61 Document domain initialization has started.
```

Figure 31. Console messages for startup of CICSHT61- 1/2

```
09.40.26 .  +DFHSI1500 CICSHT61 CICS startup is in progress for CICS Transaction Server for z/OS, Version 5 Release 4
09.40.26 .  +DFHXS1100I CICSHT61 Security initialization has started.
09.40.26 .  +DFHXS1102I CICSHT61 Security is inactive.
09.40.27 .  +DFHSI1501I CICSHT61 Loading CICS nucleus.
09.40.32 .  +DFHDU0304I CICSHT61 Transaction Dump Data set DFHDMPB opened.
09.40.34 .  +DFHXS1101I CICSHT61 Security initialization has ended.
09.40.34 .  +DFHRM0141 CICSHT61 Recovery manager autostart override record is not present. Normal processing continues.
09.40.34 .  +DFHWB0110I CICSHT61 Web domain initialization has ended.
09.40.34 .  +DFHDH0101I CICSHT61 Document domain initialization has ended.
09.40.34 .  +DFHMN0105I CICSHT61 Using default Monitoring Control Table. 9
09.40.34 .  +DFHMN0110I CICSHT61 CICS Monitoring is inactive.
09.40.35 .  +DFHSI1502I CICSHT61 CICS startup is Warm.
09.40.35 .  +DFHTS0100I CICSHT61 Temporary Storage initialization has started.
09.40.36 .  +DFHLG0103I CICSHT61 System log (DFHLOG) initialization has started.
09.40.36 .  +DFHLG0104I CICSHT61
                   System log (DFHLOG) initialization has ended. Log stream
                   CICSHT##.CICSHT61.DFHLOG is connected to structure ****************.
09.40.36 .  +DFHLG0103I CICSHT61 System log (DFHSHUNT) initialization has started.
09.40.36 .  +DFHSI1503I CICSHT61 Terminal data sets are being opened.
09.40.37 .  +DFHLG0104I CICSHT61
                   System log (DFHSHUNT) initialization has ended. Log stream
                   CICSHT##.CICSHT61.DFHSHUNT is connected to structure
                   ****************.
09.40.37 .  +DFHLG0102I CICSHT61 Log manager domain initialization has ended.
09.40.37 .  +DFHTS0101I CICSHT61 Temporary Storage initialization has ended.
09.40.38 .  +DFHKE0406I CICSHT61
                   CICS is about to wait for predecessors defined in the MVS automatic
                   restart management policy for this region.
09.40.38 .  +DFHKE0412I CICSHT61 CICS WAITPRED call to automatic restart manager has completed.
09.40.38 .  +DFHCP0101I CICSHT61 CPI initialization has started.
09.40.38 .  +DFHPR0104I CICSHT61 Partner resource manager initialization has started.
09.40.38 .  +DFHAI0101I CICSHT61 AITM initialization has started.
09.40.38 .  +DFHER5730 CICSHT61 User recovery beginning
09.40.38 .  +DFHLG0745I CICSHT61 System log full scan has started.
09.40.38 .  +DFHFC0100I CICSHT61 File Control initialization has started.
09.40.38 .  +DFHTD0100I CICSHT61 Transient Data initialization has started.
09.40.38 .  +DFHFC0562 CICSHT61 The RLS control ACB has been successfully registered by CICS.
09.40.38 .  +DFHFC0570 CICSHT61 File control RLS access has been enabled.
09.40.39 .  +DFHLG0748I CICSHT61 System log selective scan has started.
09.40.39 .  +DFHLG0749I CICSHT61 System log scan has completed.
09.40.39 .  +DFHER5731 CICSHT61 No active user records on the system log
09.40.39 .  +DFHER5732 CICSHT61 User recovery completed
09.40.39 .  +DFHTD0101I CICSHT61 Transient Data initialization has ended.
09.40.39 .  +DFHFC0101I CICSHT61 File Control initialization has ended.
09.40.39 .  +DFHCP0102I CICSHT61 CPI initialization has ended.
09.40.39 .  +DFHPR0105I CICSHT61 Partner resource manager initialization has ended.
09.40.39 .  +DFHAI0102I CICSHT61 AITM initialization has ended.
09.40.43 .  +DFHSI1519I CICSHT61 The interregion communication session was successfully started.
09.40.45 .  +DFHAP1203I CICSHT61 Language Environment is being initialized.
09.40.59 .  +DFHAP1211I CICSHT61 Language Environment initialization completed.
09.41.05 .  +DFHWB1007 CICSHT61 Initializing CICS Web environment.
09.41.05 .  +DFHWB1008 CICSHT61 CICS Web environment initialization is complete.
09.41.06 .  +DFHSI8430I CICSHT61 About to link to PLT programs during the third stage of initialization.
09.41.06 .  +CICSHT61 - PLTPI PHASE 2 PROCESSING START
09.41.06 .  +DFHSI8434I CICSHT61 Control returned from PLT programs during the third stage of initialization.
09.41.10 .  +DFHSO0120 CICSHT61 8 TCBs are initialized for SSL processing.
09.41.10 .  +DFHSO0101I CICSHT61 Sockets domain initialization has ended.
09.41.10 .  +DFHSI1517 CICSHT61 Control is being given to CICS.
09.41.10 .  +DFHEJ0102 CICSHT61 Enterprise Java domain initialization has ended.
09.41.11 .  +CICSHT61 SAMA      START OF BATCH STREAM FROM SAMA
```

Figure 32. Console messages for startup of CICSHT61- 2/2

Note:

1. The SVC value is only an example; specify the SVC value that is defined in your SVC table.
2. You must define the default CICS user ID using the **DFLTUSER** system initialization parameter, before you can attempt to bring a CICS region up. On a production system, the default user must not have access to any unnecessary CICS transactions. The resource access authorizations that you give to the default user must be clearly limited to those resources that you intend to be universally available, and therefore do not have to be restricted in any way. For information about defining the attributes of the default user ID, see Defining the default CICS user ID to RACF.
3. The resource group lists to be loaded when CICS starts up. The groups of resource definitions in these group lists are installed only during an initial or

cold start. For a warm restart, the resource definitions still installed at shutdown are reinstated from the CICS global catalog.

4. This specifies the message text that is to be displayed on the screen by the CSGM (good morning) transaction when a terminal is logged on to CICS through z/OS Communications Server, or by the CESN transaction if this transaction is used to sign on to CICS.
5. Interregion communication is started during system initialization (IRCSTRT=YES), and the CICS programs required are made available (ISC=YES).
6. The program autoinstall function is enabled at startup.
7. The DFHSM0122 message shows the limit of dynamic storage areas (DSAs) below 16 MB. This limit is set by the **DSALIM** system initialization parameter. The DFHSM0123 message shows the limit of the DSAs above 16 MB but below 2 GB. This limit is set by the **EDSALIM** system initialization parameter. For information about these storage areas, see DSALIM system initialization parameter and EDSALIM system initialization parameter.
8. Storage protection is active, because the system initialization parameter **STGPROT=YES** was specified or allowed to default to YES.
9. If you specify **MCT=NO** (the default) as a system initialization parameter, the CICS monitoring domain builds a default monitoring control table. This ensures that default monitoring control table entries are always available for use when monitoring is active. The next message indicates that monitoring is inactive.

CICS Transaction Server resource usage collection for software pricing

CICS TS supports the collection of software usage information.

For information on the use of the Sub-Capacity Reporting Tool (SCRT), see Using the Sub-Capacity Reporting Tool.

CICS registers information using the IFAUSAGE macro during the initialization of each CICS address space and the address spaces for named counter, temporary storage, and coupling facility data table servers. The region status servers is a variant of the coupling facility data table, and also registers information. The CICS version and product identifier are included in the registered information. This information is collected in SMF 89 records.

Chapter 3. Setting up CICS data sharing servers

To initialize CICS data sharing servers, you must set up AXM system services and define the appropriate data sharing server.

CICS data sharing servers support:

- Temporary storage data sharing
- Coupling facility data tables
- Named counters

  Users implementing CICSPlex SM sysplex optimised workload management will require a region status server to be configured. This server is a bespoke type of CFDT server into which CICS regions broadcast generic system status data which is subsequently interrogated by CICSPLEX SM for making dynamic routing decisions. The table structure managed by this server may not be modified, adjusted, or reconfigured for user application utilization.
- "Defining and starting AXM system services" describes how to define and start AXM system services
- "Setting up and running a temporary storage server" on page 152 describes how to set up and run a CICS temporary storage data sharing server
- "Setting up and running a coupling facility data table server" on page 168 describes how to set up and run a CICS coupling facility data table server
- "Coupling facility server operations" on page 223 gives an overview of the operations which are common to all three CICS coupling facility servers.
- "Setting up and running a named counter server" on page 206 describes how to set up and run a CICS named counter server.

Defining and starting AXM system services

The authorized cross-memory (AXM) server environment provides the runtime environment for CICS data sharing server regions. You must define and start AXM system services to run any of the CICS data sharing servers.

About this task

To establish AXM cross-memory connections for an MVS image, you must define an MVS subsystem called AXM. You can choose to create the subsystem statically or dynamically.

Procedure

- To define the MVS subsystem statically, add an entry with the required parameters to the IEFSSN*xx* member of SYS1.PARMLIB:

  ```
  SUBSYS SUBNAME(AXM) INITRTN(AXMSI)
  ```

  Defining AXM in the IEFSSN*xx* member ensures that AXM system services automatically become available when you IPL MVS.

  The AXM subsystem initialization routine, AXMSI, sets up the appropriate definitions from the master scheduler region. Note that AXM uses the subsystem definition only as a means of scheduling AXM initialization in the master scheduler address space. The MVS subsystem interface for AXM is not activated or used.

© Copyright IBM Corp. 1989, 2017

- To define the MVS subsystem dynamically, enter the following command:

  ```
  SETSSI ADD,SUBNAME=AXM,INITRTN=AXMSI
  ```

  If initialization of the AXM subsystem fails for any reason, for example, because of an error in the command or because AXMSI is not in a linklist library, MVS does not allow another attempt because the subsystem is then already defined. In this case, use a different subsystem name, such as AXM1, because AXM does not rely on a specific subsystem name. If you start AXM successfully the first time, further attempts are ignored.

What to do next

When the AXM subsystem has started successfully, you can create a data sharing server.

Setting up and running a temporary storage server

CICS transactions running in an AOR access temporary storage (TS) pools through a temporary storage server that supports a named pool. In each MVS image in the sysplex, you must set up one temporary storage server for each pool that is defined in the coupling facility.

Procedure

1. Define a TS pool as a temporary storage list structure in a coupling facility.
2. Define and start the temporary storage job, for a shared TS pool to run in an mvs batch region.
3. When the temporary storage region is running, you can issue commands to control the queue server for the pool.
4. Optionally, you can upload and reload queue pools.

Overview of the temporary storage data sharing server

CICS transactions running in an AOR access TS pools through a temporary storage data sharing server that supports a named pool.

In each z/OS image in the sysplex, you need one TS server for each pool defined in a coupling facility which can be accessed from that z/OS image. All TS pool access is performed by cross-memory calls to the TS server for the named pool.

An AOR can access more than one TS server concurrently. This multiserver access is required if you create multiple pools, because each TS server provides access to only one pool of TS queues.

CICS maps temporary storage requests to a TS server using the POOLNAME attribute of a TSMODEL resource definition.

Figure 33 on page 153 illustrates a parallel sysplex with three CICS AORs linked to the temporary storage server address space(s).

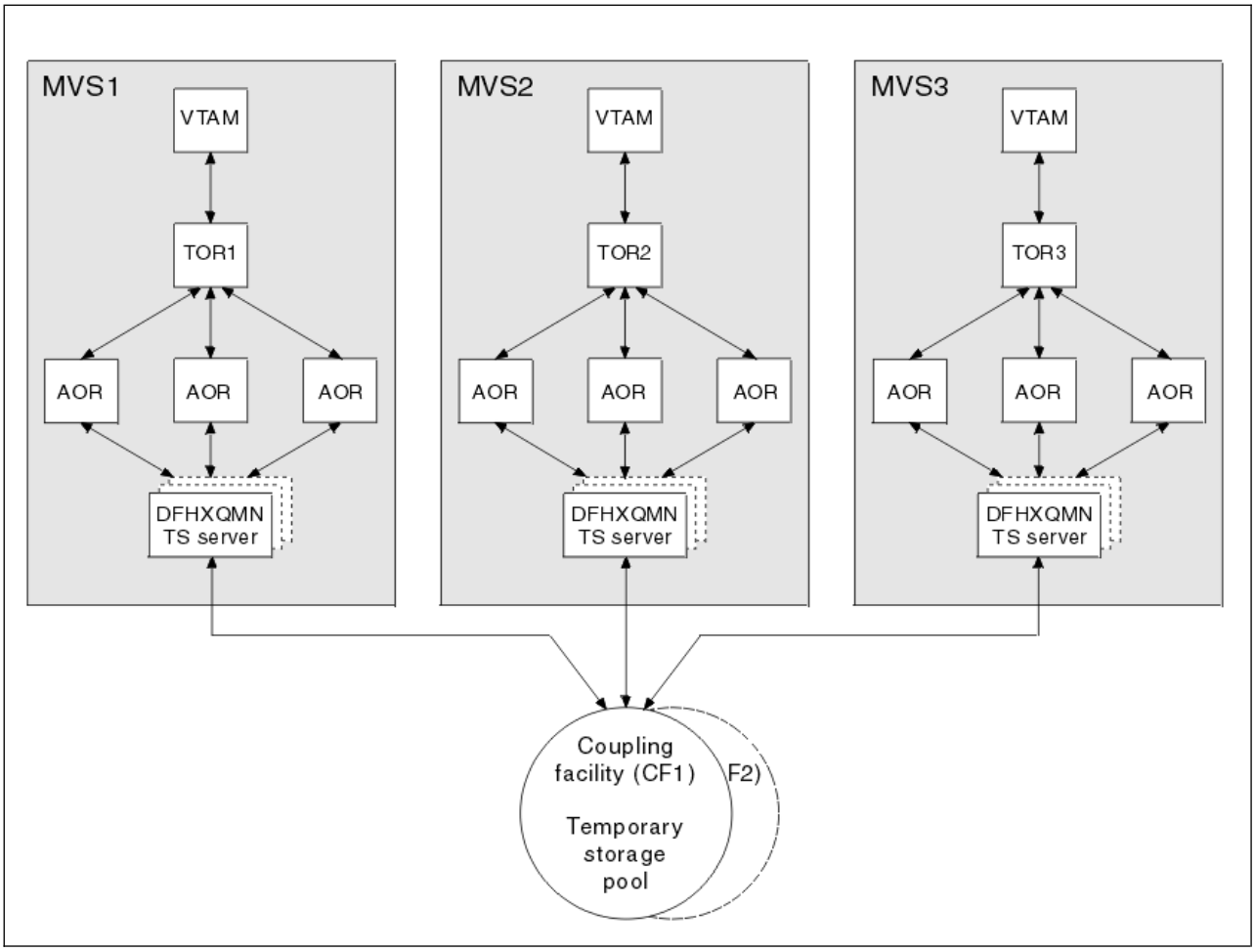


Figure 33. Conceptual view of a Parallel Sysplex with TS data sharing.

Security

The server must be authorized to access the coupling facility list structure in which the temporary storage pool is defined; XES checks this. The server must also be authorized to act as a temporary storage server; AXM checks this. For information on how to define the necessary authorizations see Authorizing access to temporary storage servers.

Defining temporary storage pools for temporary storage data sharing

You define a temporary storage (TS) pool as a temporary storage list structure in a coupling facility by using coupling facility resource manager (CFRM) policy statements.

About this task

To use TS data sharing, main or auxiliary storage for your TS queues is replaced by one or more TS pools, where the scope and function of each TS pool is similar to a queue-owning region (QOR).

Each TS pool is defined, using MVS cross-system extended services (XES), as a keyed list structure in a coupling facility. Therefore, you must define the pool using the coupling facility resource manager (CFRM) policy statements. You use

the CFRM policy definition utility, IXCMIAPU, to specify the size of the list structures required, and their placement in a coupling facility. For an example of this utility, see member IXCCFRMP in the SYS1.SAMPLIB library (see Administrative data utility for CFRM policy data in z/OS MVS Setting Up a Sysplex). An example of a definition statement is shown in Figure 34.

```
STRUCTURE  NAME(DFHXQLS_PRODTSQ1)
  SIZE(8192)
  INITSIZE(10240)
  PREFLIST(FACIL01,FACIL02)
```

Figure 34. Example of defining the estimated size of a list structure

The name of the list structure for a TS data sharing pool is created by appending the TS pool name to the prefix DFHXQLS_, giving DFHXQLS_*poolname*. When defined, you must activate the CFRM policy using the MVS operator command `SETXCF START`.

When a list structure is allocated, an initial size and a maximum size can be specified in the CFRM policy. All structure sizes are rounded up to the next storage increment for the coupling facility level (CFLEVEL) at allocation time. For example, sizes are rounded up to the nearest 1 MB for CFLEVEL 16. Provided that space is available in the coupling facility, a list structure can be dynamically expanded from its initial size towards its maximum size, or contracted to free up coupling facility space for other purposes. If the initial structure allocation becomes full, the structure does not expand automatically, even if the structure allocated is less than the specified maximum size. To expand a list structure when the allocation becomes full, you can expand it (up to its maximum size) using the following SETXCF command:

```
SETXCF START,ALTER,STRNAME=DFHXQLS_poolname,SIZE=nnnn
```

Defining the CFRM policy statements for a list structure does not create the list structure. A TS server creates the list structure during its initialization. See “Setting up and running a temporary storage server” on page 152.

If the server connection fails, there are implications for coupling facility data tables. See “Server connection management” on page 227 and Named counter recovery.

For further information about defining list structures, see the following z/OS information:

- z/OS MVS Setting Up a Sysplex
- z/OS MVS Programming Sysplex Services Guide
- z/OS MVS Programming Sysplex Services Reference

Storage calculations for temporary storage data sharing

You can use the z Systems™ Coupling Facility Structure Sizer tool (CFSizer) to calculate storage requirements for temporary storage list structures in a coupling facility.

A coupling facility structure contains both stored data and the information needed to manage and access that data, in a similar way to a key-sequenced data set. The data for each entry in the coupling facility is stored as a chain of fixed-size (usually 256-byte) elements, which means that the exact length for variable-length data must be stored separately. To do this, CICS includes a length prefix in the stored data, so space calculations must allow for each entry using a whole number of elements. The amount of internal control information depends on the level of

functionality and performance of the coupling facility control code for the current coupling facility level (CFLEVEL). The storage requirements can increase for a higher CFLEVEL. For more information, see "Coupling facility storage management" on page 224.

CFSizer is a web-based application that takes these factors into account and communicates with a coupling facility at a current CFLEVEL to calculate storage requirements. See CFSizer.

To use CFSizer to calculate storage requirements for temporary storage list structures, enter the following information:

Maximum number of queues
: The maximum number of data lists that are reserved when CICS allocates the structure list. This value determines the maximum number of large queues that can be stored in the structure and corresponds to the **MAXQUEUES** server parameter. See "List structure parameters" on page 159.

  A large queue is one where the total size of the data items exceeds 32K. For a small queue with multiple items that does not exceed 32K, all the queue items are stored as the data portion of the queue index entry. If the queue exceeds 32K, it is converted to a form where one item per entry is stored in a separate list in the structure and is referred to by the queue index entry.

  Specify a large enough number to handle large queues, but not so large that unused preallocated list headers use a large amount of coupling facility storage. The valid range is from 1 to 999999. The default is 1000.

Average rounded item size
: The average amount of storage required for each TS queue item. Each item has a two-byte length prefix and is stored as one or more 256-byte elements. This value determines the entry to element ratio that is used to calculate the required structure size. The valid range is from 1 to 32768. The default is 256.

  If all queue items are approximately the same size, calculate this value by taking the average data size, adding two, and rounding up to the next multiple of 256. The amount of element storage required is two bytes more than the data item size because of the length prefix on each item.

  If queue items are different sizes, round up each size first before you take the average. For example, if half the items are 100 bytes and half are 300 bytes, round up the sizes to 256 and 512 respectively, then average them. The resulting average rounded item size is 384, which is more accurate than using the average item size of 200 and then rounding it up to 256.

Total number of items in all queues
: The total number of entries in all the TS queues.

Target usage percent
: The percentage of the structure space that the given total number of items are expected to use. Specify a number in the range of 1 to 100. The default is 75. This value ensures the following:

  - Free space exists for temporary expansion.
  - If the initial free space is not enough, there is time to expand the structure in response to warning messages (which normally start at 80%).
  - Activity to alter entry to element ratios is reduced.

Maximum expansion percent
The percentage that the structure can expand. If a non-zero value is specified, the maximum structure size will be greater than the initial structure size by an amount such that the total amount of data can increase by this percentage. For example, if the value 200 is specified, the initial size is enough to store the specified total number of items, and the maximum size is enough to store three times that number of items.

Defining TS server regions

A shared TS pool consists of an XES list structure, which is accessed through a cross-memory queue server region. You start a shared TS pool in an MVS image by starting up a queue server region for that pool.

About this task

To start the TS server region for a shared TS pool, you can use either a batch job or a started task. The job or task must invoke the queue server region program, DFHXQMN, which is in an APF-authorized library.

Procedure

1. Specify the DFHXQMN program either in a SYSIN data set defined in the JCL, or in the **PARM** parameter on the EXEC statement.
2. Specify the mandatory and optional startup parameters for the DFHXQMN program.
   a. You must specify a SYSPRINT DD statement for the print file.
   b. You must specify a SYSIN DD statement for the server parameters.
   c. You must specify the TS pool name.
   d. You must concatenate the license activation data set (the SDFHLIC library) to the STEPLIB DD statement.
   e. It is recommended that you specify TIME=NOLIMIT. The server task remains in a wait during most normal processing, because server processing is performed under the TCB of the client CICS region. If you omit this parameter, your server job could fail with abend S522 (wait limit exceeded), depending on the JWT value specified in the SMFPRM*xx* member of SYS1.PARMLIB.
   f. Specify additional parameters as required. For example, you might want to control the maximum number of queues that are to be supported in the pool and the number of buffers the server is to allocate.

Sample startup job for a TS server

Figure 35 on page 157 shows an example of the JCL you might use to start a TS server.

```
//PRODTSQ1 JOB  ...
//TSSERVER EXEC PGM=DFHXQMN,REGION=64M,TIME=NOLIMIT Start TS data sharing server
//STEPLIB  DD   DSN=CICSTS54.CICS.SDFHAUTH,DISP=SHR    Authorized library
//         DD   DSN=CICSTS54.CICS.SDFHLIC,DISP=SHR     License activation data set
//SYSPRINT DD   SYSOUT=*        Messages and statistics
//SYSIN    DD   *
POOLNAME=PRODTSQ1               Pool name
MAXQUEUES=5000                  Allow up to 5000 large queues
BUFFERS=1000                    1000 buffers (32K each, 32M total)
/*
```

Figure 35. Sample startup job for a TS queue server

Queue server REGION parameter

The queue server REGION parameter JCL needs to specify at least enough virtual storage for the specified number of buffers plus the storage used to process queue requests.

Each buffer occupies a little more than 32K bytes, and each connected CICS region can have up to ten queue requests active at a time, each using 5K to 10K bytes, so to be safe the REGION size should allow at least 32K per buffer and 100K for each connected CICS region, plus a margin of about 10% for other storage areas.

During server initialization, the server acquires all of the available storage above the 16M line, as determined by the REGION size, then releases 5% of it for use by operating system services. It also acquires 5% of the free storage below the line for use in routines which require 24-bit addressable storage, for example sequential file read and write routines.

After server initialization, AXM page allocation services are used to manage server region storage. Server statistics indicate how much storage is allocated and used within the storage areas above and below the 16M line, which are called AXMPGANY and AXMPGLOW in the statistics.

If a task in the server region or a cross-memory request runs out of storage, this is likely to result in AXM terminating that task or request using a simulated abend with system completion code 80A to indicate a GETMAIN failure. Although the server can usually continue processing other requests in this case, running out of storage in a critical routine can cause the server to terminate, so it is best to ensure that the REGION size is large enough to eliminate the risk.

TS queue server parameters

Parameters are specified in the form KEYWORD=value. You can optionally specify keywords in mixed case to improve readability.

If you specify more than one parameter in the PARM field, or on the same SYSIN input line, the parameters must be separated by commas. Any text following one or more spaces is taken as a descriptive comment. Any parameter line starting with an asterisk or a space is assumed to be a whole line comment.

You can enter some parameter keywords in more than one form, such as an abbreviation. The standard form of each keyword is generally the longest form of the first word shown.

The main parameters used are listed on the server print file during start-up.

The following parameters are all valid as initialization parameters (in the SYSIN file, or the PARM field), and some can be modified by the server SET command.

You can display any parameter with the server DISPLAY command. Display the values of all parameters using DISPLAY ALLPARMS.

Specify the following keywords to give combined lists of information:

- PARMS for main parameter values
- STATS for all available statistics
- INIT to select parameters and statistics whose values are usually displayed when initialization is complete

Primary parameters:

These parameters are usually specified for all servers:

POOLNAME=*pool_name*
specifies the name, of 1 to 8 characters, of the queue pool used to form the server name and the name of the coupling facility list structure DFHXQLS_poolname. This parameter is valid only at initialization, and must always be specified.

This keyword can also be coded as **POOL**.

BUFFERS={100|number}
specifies the number of queue buffers to allocate for the server address space.

A queue index buffer holds a queue index entry plus up to 32K of queue data (for a small queue). When a READ or WRITE request completes the queue index information is retained in the buffer. This can avoid the need to reread the queue index if the same queue is referenced from the same MVS image before the buffer has been reused. If no buffer is available at the time of a request, the request is made to wait until one becomes free.

The number of buffers should preferably be at least ten for each CICS region that can connect to the server in this MVS image. This avoids the risk of buffer waits. Additional buffers may be used to reduce the number of coupling facility accesses by keeping recently used queue index entries in storage. In particular, if the current version of a queue index entry is in storage at the time a queue item is read, the request requires only one coupling facility access instead of two. If the current version of a queue index entry is in storage when a second or subsequent item is written to the same queue, the request requires only one coupling facility access instead of three.

It is not worth defining extra buffers beyond the point where this might cause MVS paging, as it is more efficient to reread the index entry than to page in the buffer from auxiliary storage. This parameter is valid only at initialization.

The valid range is from 1 to 999999.

This keyword can also be coded as **BUF**.

FUNCTION={SERVER|UNLOAD|RELOAD}
Information about this parameter is given in “Unloading and reloading queue pools” on page 166.

STATSOPTIONS={NONE|SMF|PRINT|BOTH}
specifies the statistics options that determine whether interval statistics are produced and whether statistics are sent to SMF, the print file, or both.

This keyword can also be coded as **STATSOPT**.

ENDOFDAY={00:00|hhmm}
specifies the time when end of day statistics are to be collected and reset. If statistics options specify NONE, end of day statistics are written to the print file. The valid range is from 00:00 to 24:00.

This keyword can also be coded as **EOD**.

STATSINTERVAL={3:00|hhmm}
specifies the statistics interval, within the range of 1 minute to 24 hours. It is ignored if STATSOPTIONS=NONE.

The valid range is from 00:01 to 24:00 (although it may be specified in seconds).

This keyword can also be coded as **STATSINT**.

Automatic restart manager (ARM) parameters:

During server initialization, the server unconditionally registers with ARM except when the server program is invoked with either the UNLOAD or the RELOAD functions. The server will not start if the registration fails.

Use the following parameters to override default processing for the automatic restart manager:

ARMELEMENTNAME=*elementname*
specifies the automatic restart manager element name, up to 16 characters, to identify the server to ARM for automatic restart purposes. The permitted characters for the element name are A to Z 0-9 $ # @ and the underscore symbol (_).

The default identifier is of the form DFHXQ*nn_poolname*, where XQ represents the server type, *nn* is the &SYSCLONE value for the system (which can be either one or two characters), and *poolname* is the name of the pool served by the server.

This parameter is only valid at server initialization.

This keyword can be abbreviated to **ARMELEMENT** or **ARMELEMNAME**.

ARMELEMENTTYPE=*elementtype*
specifies the automatic restart manager element type, up to 8 characters for use in ARM policies as a means of classifying similar elements. The permitted characters for the element type are A to Z 0-9 $ # and @.

The default element type is SYSCICSS.

This parameter is only valid at server initialization.

This keyword can be abbreviated to **ARMELEMTYPE**.

List structure parameters:

The list structure parameters specify the attributes that are used only for initial allocation of the temporary storage list structure for a TS pool. Initial allocation occurs the first time a server is started for the TS pool.

The list structure parameters are as follows:

MAXQUEUES={1000|*number*}
Specifies the maximum number of data lists to be reserved when the structure is allocated, which determines the maximum number of large queues that can

be stored in the structure. This parameter also determines how many list headers are defined when the structure is created.

Specify a large enough number to handle large queues, but not so large that unused preallocated list headers use a large amount of coupling facility storage.

You cannot change this number without reallocating the structure. Therefore, if the structure is being allocated at less than its maximum size, the value here should be based on the maximum possible size of the structure rather than its initial size.

This parameter corresponds to the "Maximum number of queues" value in the CFSizer tool (see "Storage calculations for temporary storage data sharing" on page 154.

The valid range is from 1 to 999999.

This keyword can also be coded as **MAXQ**.

POOLSIZE={0|*number*{K|M|G}}
Specifies the maximum amount of storage to be allocated for the list structure, expressed as kilobytes (*n*K), megabytes (*n*M), or gigabytes (*n*G).

This parameter is used when the list structure is created with a specified value of less than that specified for the list structure in the coupling facility resource management (CRFM) policy.

The default value 0 specifies that no maximum limit is applied here and the maximum that is specified in the CFRM policy applies.

A non-zero value is rounded up by MVS to the next storage increment for the coupling facility level (CFLEVEL). For example, the value is rounded up to the nearest 1 MB for CFLEVEL 16.

The valid range is from 0 to 16777215M. However, you must specify a value that is less than the maximum size of a structure in z/OS, otherwise a z/OS error occurs. For example, in z/OS, Version 1 Release 12, the maximum size of a structure is 1048576 MB (1 TB). For more details, see the information about CFRM parameters in *z/OS MVS Setting Up a Sysplex*.

For information about defining temporary storage list structures, see "Defining temporary storage pools for temporary storage data sharing" on page 153.

Debug trace parameters:

These parameters are used only for intensive debug tracing.

Note that using these options in a production environment may significantly impact performance and cause the print file to grow very rapidly, using up spool space.

Trace messages from cross-memory requests may be lost if they are generated faster than the trace print subtask can print them. In such cases, the trace indicates only how many messages were lost.

TRACECF={OFF|ON}
specifies the coupling facility interface debug trace options, OFF or ON. This option produces trace messages on the print file indicating the main parameters to the coupling facility request interface and the result from the IXLLIST macro.

This keyword can also be coded as **CFTR** or **CFTRACE**.

TRACERQ={OFF|ON}
specifies the queue request debug trace options, OFF or ON. This option produces trace messages on the print file indicating the main parameters on entry to the shared queue request or shared queue inquire interface and the results on exit.

This keyword can also be coded as **RQTR** or **RQTRACE**.

Tuning parameters:

These parameters are provided for tuning purposes, but usually you can omit these and let the TS server use the default values.

ELEMENTRATIO={1|*number*}
An MVS coupling facility resource management (CFRM) parameter that specifies the element part of the entry-to-element ratio when the structure is first allocated. This value determines the proportion of the structure space that is initially set aside for data elements.

The ideal value for the entry-to-element ratio is the average size of data for each entry divided by the element size. However, the server automatically adjusts the ratio according to the actual entry and element usage.

This parameter is valid only at server initialization, and is used only when the structure is first allocated. The valid range is from 1 to 255.

You can abbreviate this keyword to **ELEMRATIO**.

For further information about this parameter in the coupling facility, see *z/OS MVS Programming: Sysplex Services Guide*.

ELEMENTSIZE={256|*number*}
An MVS CFRM parameter that specifies the element size for structure space, which must be a power of 2. This parameter is used only for coupling facility log streams. The valid range is 256 to 4096. For current coupling facility implementations, there is no reason to change this parameter from the default value of 256.

This parameter is valid only at server initialization and is used only when the structure is first allocated.

You can abbreviate this keyword to **ELEMSIZE**.

ENTRYRATIO={1|*number*}
An MVS CFRM parameter that specifies the entry part of the entry-to-element ratio when the structure is first allocated. This value determines the proportion of structure space that is initially set aside for list entry controls.

It is not essential to specify this parameter because the server automatically adjusts the ratio, based on actual usage, to improve space utilization if necessary.

This parameter is valid only at server initialization and is used only when the structure is first allocated. The valid range is from 1 to 255.

For further information about this parameter in the coupling facility, see *z/OS MVS Programming: Sysplex Services Guide*.

LASTUSEDINTERVAL={00:10|*hh*:*mm*}
A CICS parameter that specifies how often the last used time for large queues is updated.

For small queues, the last used time is updated on every reference. For large queues, updating the last used time requires an extra coupling facility access, so this update occurs only if the queue has not previously been accessed in this interval of the current time. Therefore, the last used time interval returned by an **INQUIRE** command can be greater than the true value by an amount up to the value of this parameter. The last used time is used mainly to determine whether the queue is obsolete, so a suitable value for the **LASTUSEDINTERVAL** parameter is normally a few minutes.

See INQUIRE TSQUEUE / TSQNAME.

The valid range is from 00:00 to 24:00. This value can also be specified in seconds.

You can abbreviate this keyword to **LASTUSEDINT**.

SMALLQUEUEITEMS={9999|*number*}
A CICS parameter that specifies the maximum number of items that can be stored in the small queue format in the queue index entry data area. This parameter can force a queue to be converted to the large queue format if it has a large number of small items. It can be more efficient to write the items separately than to rewrite the whole small queue data area each time.

The valid range is from 1 to 32767.

SMALLQUEUESIZE={32K|*number*}
A CICS parameter that specifies the maximum data size for a small queue, including the two-byte length prefix on each data item. If a queue exceeds the maximum size when a second or subsequent item is written to it, the queue is converted to the large queue format.

This parameter can force queues to be converted to the large queue format at a smaller size than 32K. This is to prevent large amounts of data being written to the small queue format. On systems where asynchronous coupling facility processing causes contention for hardware resources, using this parameter can improve performance. However, on most systems, it is more efficient to defer conversion until the maximum size of 32K is reached.

The valid range is from 4096 to 32768.

Warning parameters:

These parameters modify the threshold at which warning messages and automatic ALTER actions occur when the structure becomes nearly full:

ELEMENTWARN={80|number}
specifies the percentage of elements in use at which warnings and automatic ALTER actions should be first triggered.

The valid range is from 1 to 100.

This keyword can also be coded as **ELEMWARN**.

ELEMENTWARNINC={5|number}
specifies the percentage increase (or decrease) of elements in use before the next warning should be triggered (reduced to 1 when next increase would otherwise reach 100). Additional messages are issued as the number of elements in use changes. The messages stop when the number of elements in use falls at least by this percentage below the initial warning level.

The valid range is from 1 to 100.

This keyword can also be coded as **ELEMWARNINC**.

ENTRYWARN={80|number}
specifies the percentage of entries in use at which warnings and automatic ALTER actions should be first triggered.

The valid range is from 1 to 100.

ENTRYWARNINC={5|number}
specifies the percentage increase (or decrease) of entries in use before next warning should be triggered (reduced to 1 when next increase would otherwise reach 100). Additional messages are issued as the number of elements change. The messages stop when the number of entries in use falls at by least the specified percentage below the initial warning level.

The valid range is from 1 to 100.

Automatic ALTER parameters:

Define the following parameters to modify the conditions under which the server attempts an automatic ALTER action when the structure becomes nearly full.

For details of the queue server automatic ALTER process, see "Queue server automatic ALTER processing."

ALTERELEMMIN={100|number}
specifies the minimum number of excess elements that must be present for an automatic ALTER to be issued to convert those elements to entries.

The valid range is from 1 to 999999999.

ALTERELEMPC={1|number}
specifies the minimum percentage of excess elements that must be present for an automatic ALTER to be issued to increase the proportion of entries.

The valid range is from 0 to 100.

ALTERENTRYMIN={100|number}
specifies the minimum number of excess entries that must be present for an automatic ALTER to be issued to convert those entries to elements.

The valid range is from 0 to 999999999.

ALTERENTRYPC={1|number}
specifies the minimum percentage of excess entries which must be present for an automatic ALTER to be issued to increase the proportion of elements.

The valid range is from 0 to 100.

ALTERMININTERVAL={00:10|hhmm}
specifies the minimum time interval to be left between automatic ALTER attempts when the structure is nearly full (above the element or entry warning level).

The valid range is from 00:00 to 24:00.

This keyword can also be coded as **ALTERMININT**.

Queue server automatic ALTER processing

The queue server monitors the total number of elements and entries in use in the structure, using information returned by the coupling facility on every request.

When the numbers in use exceed the specified thresholds, a warning message, DFHXQ0411 or DFHXQ0412, is issued, and is repeated each time the number in use increases beyond further thresholds.

Each time the warning is issued, the server tests whether an automatic ALTER for the entry to element ratio should be performed. The test is done by calculating how many excess elements or entries will be left when the other runs out completely. This is based on the ratio between the current numbers of elements and entries in use.

An IXLALTER request is issued to alter the entry to element ratio to the actual current ratio between the number of entries and elements in use if:

- The number of excess elements or entries exceeds the number specified in the ALTERELEMMIN or ALTERENTRYMIN parameter, and
- The same number expressed as a percentage of the total exceeds the value specified in the ALTERELEMPC or ALTERENTRYPC parameter

Only one ALTER request may be active at a time for a given structure. If the ALTER process is started by one server, the ALTER of another server will be rejected. However, the system automatically notifies all servers when the ALTER completes, giving the new numbers of elements and entries so that each server can update its own status information.

A further ALTER attempt is suppressed until at least the minimum ALTER interval (specified by the ALTERMININTERVAL parameter) has elapsed, if:

- Any form of ALTER has already been used recently (by any server or by an operator SETXCF ALTER command), and
- The structure space usage has remained above warning levels since the previous attempt.

Shared TS queue server commands

Commands can be issued to control a queue server using the MVS MODIFY (F) command specifying the job or started task name of the server region.

The general form of a queue server command is as follows:

```
F server,cmd parameter,parameter...   comments
```

The MVS STOP command is equivalent to issuing the server command STOP using the MVS MODIFY command.

The queue server supports the following commands:

SET keyword=value
changes one or more server parameter values. This applies to all parameters other than those indicated as being for initialization only. The command can be abbreviated to T, as for the MVS SET command.

DISPLAY keyword
displays one or more parameter values, or statistics summary information, on the console. The valid parameter keywords for DISPLAY and PRINT are described later in this section. The command can be abbreviated to D, as for the MVS DISPLAY command.

PRINT keyword
produces the same output as DISPLAY but only on the print file.

STOP
terminates the server, waiting for any active connections to terminate first, and preventing any new connections.

The command can be abbreviated to P, as for the MVS STOP command.

CANCEL {RESTART={NO|YES}}
Terminate the server immediately. You can specify whether automatic restart should be requested.

For information about CANCEL RESTART see "The CANCEL command options" on page 166.

The server also responds to XES events such as an operator SETXCF command to alter the structure size. If the server can no longer access the coupling facility, it automatically issues a server CANCEL command to close itself down immediately.

DISPLAY and PRINT keywords

DISPLAY or PRINT can display the values of any system initialization parameters.

These keywords can also display the following additional information:

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

CONNECTIONS
List of the job names and APPLIDs for the regions currently connected to this server. You can also code this keyword as **CONN**.

Statistics summaries

BUFSTATS
Queue index buffer pool statistics.

You can also code this keyword as **BUFST**.

CFSTATS
Coupling facility interface I/O and response statistics.

You can also code this keyword as **CFST** or **STATSCF**.

POOLSTATS
Usage statistics for the pool list structure as a whole.

You can also code this keyword as **POOLST**.

STORAGESTATS
Main storage allocation statistics for the server address space.

You can also code this keyword as **STGST**, **STGSTATS**, or **STORAGEST**.

Keywords representing combined lists of information

PARAMETERS
Main parameter values.

You can also code this keyword as **PARM**, **PARMS**, or **PARAM**.

ALLPARAMETERS
All parameter values.

You can also code this keyword as **ALLPARMS**.

STATISTICS
All available statistics.

You can also code this keyword as **STAT** or **STATS**.

INITIALIZED
Selected parameters and statistics whose values are usually displayed when initialization is complete.

You can also code this keyword as **INIT**.

ARM
Display all ARM-related parameter values:
- ARMELEMENTNAME
- ARMELEMENTTYPE
- ARMREGISTERED

You can also code this keyword as **ARMSTATUS**.

The CANCEL command options

You can use the CANCEL command to request an automatic restart.

Specify the following parameter:

RESTART={NO∨YES}
Terminate the server immediately, specifying whether or not automatic restart should be requested. The default is RESTART=NO.

If the server encounters an unrecoverable problem with the coupling facility connection, consisting either of lost connectivity or a structure failure, it cancels itself using the CANCEL RESTART=YES command. This terminates the existing connection and shuts down the server. A new instance of the server job is then started.

A server can also be restarted explicitly using either the server command CANCEL RESTART=YES or the MVS command CANCEL jobname,ARMRESTART.

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

Unloading and reloading queue pools

The contents of a queue pool can be unloaded to a sequential file and subsequently reloaded by running the server program using the FUNCTION parameter.

About this task

This can be used to preserve the queue pool across planned coupling facility maintenance, or to move the queue pool to a different coupling facility. The server does not support automatic structure REBUILD, but the unload and reload process is more flexible. The reloaded queue pool does not need the same name as the original pool.

FUNCTION={UNLOAD|RELOAD}
requests the server to perform the special functions UNLOAD or RELOAD. When the unload or reload processing has completed (normally or abnormally) the server program terminates.

If this parameter is omitted, the server program initializes the cross-memory queue server environment.

When UNLOAD or RELOAD is specified, the server program requires exclusive use of the list structure. If the structure is currently being used by a normal server, the attempt to unload or reload is rejected. Similarly, if a normal server attempts to start while an unload or reload function is in progress, the attempt is rejected because shared access to the structure is not available.

All normal server parameters can be specified on UNLOAD and RELOAD, but many of these, such as the number of queue buffers, are ignored because they do not apply to unload or reload processing.

The UNLOAD function requires a DD statement for file name DFHXQUL describing the sequential data set to which the queue pool is to be unloaded. The format of the unloaded file is:

```
RECFM=F,LRECL=4096,BLKSIZE=4096.
```

An upper limit for the total size of the data set in bytes can be estimated from the pool usage statistics produced by the server. The total data size in bytes is obtained by multiplying the number of elements in use by the element size (usually 256), and for each queue there is also some control information which typically occupies fewer than 100 bytes per queue. The size is normally smaller than this because unused space in data elements is not included in the unloaded file. See Figure 36 for an example of UNLOAD JCL.

```
//UNLDTSQ1 JOB  ...
//TSUNLOAD EXEC PGM=DFHXQMN         CICS TS queue server program
//STEPLIB  DD   DSN=CICSxxx.SDFHAUTH,DISP=SHR   Authorized library
//SYSPRINT DD   SYSOUT=*            Options, messages and statistics
//DFHXQUL  DD   DSN=TSQ1.UNLOADED.QPOOL,   Unloaded queue pool
//         DISP=(NEW,CATLG),
//         SPACE=(4096,(10000,1000))    Estimated size in 4K blocks
//SYSIN    DD   *
FUNCTION=UNLOAD                   Function to be performed is UNLOAD
POOLNAME=PRODTSQ1                 Pool name
/*
```

Figure 36. Example of UNLOAD JCL

The RELOAD function requires a DD statement for file name DFHXQRL describing the sequential data set from which the queue pool is to be reloaded. The structure is allocated if necessary during reloading, in which case the same server parameters may be used to control structure attributes as for normal server execution. The RELOAD process bypasses any queues that are already found in the queue pool, because, for example, the structure was too small and the reload job had to be restarted after using ALTER to increase the size.

Note that when a pool is nearly full (with less than about 5% free entries and elements) there is no guarantee that it can be unloaded and reloaded into a structure of exactly the same size. The amount of space available is affected by the current ratio of entries to elements, which can only be controlled approximately by the automatic ALTER process.

If the structure reaches the warning level during reloading, the automatic ALTER process attempts to adjust the entry to element ratio, and the reload process will automatically wait for the ALTER to complete if no space is available while an ALTER is still in progress.

If RELOAD fails because it runs out of space, the resulting messages include the numbers of queues reloaded and blocks read up to the time of the failure. Compare these values with those in the messages from the original UNLOAD to determine how many more queues and how much more data remained to be loaded. See Figure 37 on page 168 for an example of RELOAD JCL.

```
//RELDTSQ1 JOB  ...
//TSRELOAD EXEC PGM=DFHXQMN         CICS TS queue server program
//STEPLIB  DD   DSN=CICSxxx.SDFHAUTH,DISP=SHR   Authorized library
//SYSPRINT DD   SYSOUT=*            Options, messages and statistics
//DFHXQRL  DD   DSN=TSQ1.UNLOADED.QPOOL,DISP=OLD  Unloaded queue pool
//SYSIN    DD   *
FUNCTION=RELOAD                     Function to be performed is RELOAD
POOLNAME=PRODTSQ1                   Pool name
POOLSIZE=50M                        Increased pool size
MAXQUEUES=10000                     Increased number of big queues
/*
```

Figure 37. Example of RELOAD JCL

Setting up and running a coupling facility data table server

CICS coupling facility data tables provide rapid sharing of working data within a sysplex, with update integrity. The data tables are held in coupling facility structures, with access through a named server. You must set up one server for each pool in an MVS image.

Before you begin

Before you can start a server for named coupling facility data table pool, define the coupling facility structure to be used for the pool. See "Defining a coupling facility data table pool" on page 172 for information about defining a coupling facility list structure.

About this task

You can put related groups of coupling facility data tables in separate pools; for example, you might want to have one pool for production and another for test. A pool is defined as a list structure in the coupling facility resource management (CRFM) policy. The pool name is used to form the server name with the prefix DFHCF and is specified in the startup JCL for the server.

Procedure

1. Define and start a coupling facility data table server job, to run in an mvs batch region.
2. When the server is running you can perform various operating tasks:
   - You can control the server region using MVS commands.
   - You can delete or empty the pool for the coupling facility data tables.
   - You can unload and reload the contents of a pool to and from a sequential data set.

Overview of a coupling facility data table server

CICS coupling facility data tables is designed to provide rapid sharing of working data within a sysplex, with update integrity.

The data is held in a table that is similar in many ways to a shared user-maintained data table, and the API used to store and retrieve the data is based on the file control API used for user-maintained data tables.

Figure 38 illustrates a Parallel Sysplex® with three CICS AORs linked to the coupling facility data table servers.

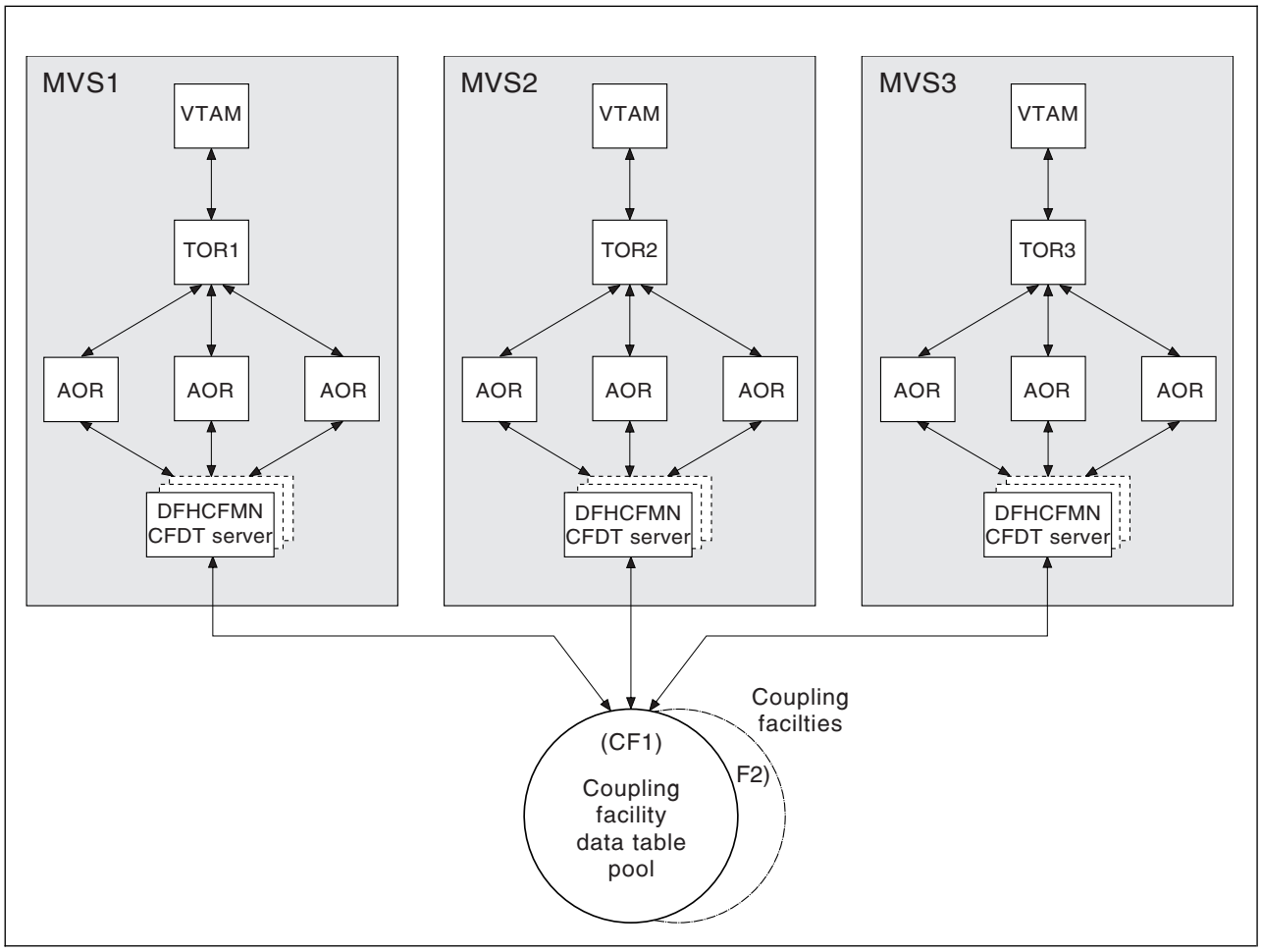


Figure 38. Conceptual view of a Parallel Sysplex with coupling facility data table servers

Coupling facility data tables

Coupling facility data tables (CFDTs) provide a method of file data sharing, using CICS file control, without the need for a file-owning region, and without the need for VSAM RLS support. CICS CFDT support provides rapid sharing of working data within a sysplex, with update integrity.

The data is held in a coupling facility, in a table that is similar to a shared user-maintained data table. You must define the resources required for CFDTs in an MVS coupling facility resource management (CFRM) policy.

Because read access and write access have similar performance, this form of table is useful for scratchpad data. Typical uses might include sharing scratchpad data between CICS regions across a sysplex, or sharing of files for which changes do not have to be permanently saved. There are many different requirements for scratchpad data, and most of these can be implemented using CFDTs.

CFDTs are useful for grouping data into different tables, where the items can be identified and retrieved by their keys. For example:

- You can use a field in a CFDT to maintain the next free order number for use by an order processing application.
- You can maintain a list of the numbers of lost credit cards in a CFDT.

To an application, a CFDT appears much like a sysplex-wide user-maintained data table, because it is accessed in the same way, by using the file control API. However, in a CFDT there is a maximum key-length restriction of 16 bytes.

Coupling facility data table models

CFDTs can use the contention model or the locking model. You specify the model that each table uses on its file resource definition, so different tables can use different models.

- The contention model gives optimal performance but generally requires programs written to exploit it. This is because the CHANGED condition code (which indicates that the data has changed since the application program issued a read-for-update request) is specifically for this model. Programs that are not written for the contention model might not be able to handle this condition correctly. The CHANGED response can occur on a REWRITE or DELETE command. There is also a situation with the contention model in which the NOTFND response can be returned on a REWRITE or DELETE.

  This model is nonrecoverable: CFDT updates are not backed out if a unit of work fails.
- The locking model is API-compatible with programs that conform to the UMT subset of the file control API (this subset is almost the full file control API). This model can be one of the following:
  - Nonrecoverable.

    Locks do not last until syncpoint, and CFDT updates are not backed out if a unit of work fails.
  - Recoverable.

    CFDTs are recoverable after a unit-of-work failure or a CICS region failure (updates made by units of work that were in-flight at the time of the CICS failure are backed out).

    The recoverable locking model supports indoubt and backout failures. If a unit of work fails when backing out an update to the CFDT, or if it fails indoubt during syncpoint processing, the locks are converted to retained locks and the unit of work is shunted.

Server connections

A client CICS region establishes a single cross-memory connection to a server the first time a request refers to the pool for that server. If the server connection fails, there are implications for coupling facility data tables. See “Server connection management” on page 227 and .

Coupling facility data table structures and servers

Coupling facility data tables (CFDTs) are held in coupling facility list structures. Access to a CFDT is through a named CFDT server.

The CFDT server can be thought of as similar to a shared data tables file-owning region. For information about shared data tables support, see The data table sharing environment. You can use CFDT support to separate related groups of CFDTs by storing them in separate pools. For example, you can have one pool for production and another for test.

A CFDT pool is a coupling facility list structure, and access to it is provided by a coupling facility data table server. Within each MVS image, there must be one CFDT server for each CFDT pool accessed by CICS regions in the MVS image.

The names of the servers are formed by adding the server address space name to the pool name. For example, for FACILTY class RACF profiles, the server name is DFHCF.*poolname*.

Coupling facility list structures are defined in the coupling facility resource management (CFRM) policy. Structure names for CFDTs take the form DFHCFLS_*poolname*. The CFDT pool name is then specified in the startup JCL for the CFDT server.

Access using file control API

A CFDT is accessed from CICS through file control commands.

The file name specified on the command indicates the name of the table and pool in which it resides. The table name is either specified on the file definition or is the same as the file name, and the pool name is specified on the file definition. The table is uniquely identified by the pool name and table name, so that two tables with the same name can exist in different pools, and are entirely separate entities.

Automatic connection to CFDT pools

CICS connects to the CFDT server for a given pool the first time that a CFDT in that pool is referenced. CICS also reconnects automatically to the CFDT server when the server restarts after a failure.

CFDT servers are protected against misuse by CICS regions that call them, thus ensuring system integrity. Protection is provided so that the calling region cannot modify sensitive parameters to authorized functions.

Similarly, CICS is protected from any side effects if a CFDT server fails. If a CICS region issues a file control request to a CFDT server that has failed, the resulting MVS abend is trapped and returned to the application program as a SYSIDERR condition.

CFDT creation and deletion

CICS automatically creates a coupling facility data table (CFDT) when a first reference requires the CFDT to be opened. This CFDT is then used by the same region, or other CICS regions, that issue subsequent open requests for other files that name the same coupling facility data table.

CICS can optionally load the CFDT automatically from a source VSAM (KSDS) data set when it is first opened. Unlike user-maintained data tables, with CFDTs you can specify that there is no associated source data set, allowing you to create an empty CFDT.

Your application programs have access to a CFDT as soon as it is created, although there are some restrictions on the keys that can be accessed while data is being loaded.

When a CFDT is loaded, it becomes an independent entity, separate from the behavior of the CICS regions that access the table, or caused the table to be loaded.

Even when all CICS regions have terminated, either normally or abnormally, a CFDT remains in the coupling facility until you take explicit action to delete the structure or the coupling facility.

To delete the CFDT contents or structure, you can use the following command:

```
MODIFY cfdt_server, DELETE TABLE=name
```

CFDT administration

CICS provides some utility functions that you can use to obtain summary information and periodic statistics from a coupling facility data table server about CFDTs defined in a pool. See Coupling facility data table statistics and Coupling Facility Data Table Pools report in Reference.

You can use this information when you administer coupling facility data table pools, and to evaluate capacity. See "Coupling facility data table server parameters" on page 176 for details.

Defining a coupling facility data table pool

You define the list structure for a coupling facility data table (CFDT) in a coupling facility resource manager (CFRM) policy in a sysplex couple data set.

CICS accesses a CFDT pool, which can contain one or more CFDTs, through a server region using cross-memory services. For information about setting up and starting a coupling facility data table region, see "Setting up and running a coupling facility data table server" on page 168.

From the application perspective, a pool and its server are similar to a file-owning region, and the pool can contain any number of tables, provided that each table has a unique table name.

Before a CFDT server can use its pool, the active CFRM policy must contain a definition of the list structure to be used for the pool. The CFRM structure definition specifies the size of the list structure and the preference list of coupling facilities in which it can be stored. You must add a statement that specifies the list structure to a selected CFRM policy, and then activate the policy.

You use the CFRM policy definition utility, IXCMIAPU, to specify the size of the list structures required, and their placement in a coupling facility.

You create the name of the list structure for a CFDT pool by adding the prefix DFHCFLS_ to the pool name, giving DFHCFLS_*poolname*.

Using IXCMIAPU to update a policy

You can use the administrative data utility, IXCMIAPU, to update an administrative policy in the CFRM couple data set. The utility adds or changes policy data in the administrative policies only: it does not change any information in the system copy of the active CFRM policy.

For an example of a job to run this utility, see member IXCCFRMP in the SYS1.SAMPLIB library (see Administrative data utility for CFRM policy data in z/OS MVS Setting Up a Sysplex.

The following example shows a policy statement for a coupling facility data table pool.

```
STRUCTURE NAME(DFHCFLS_PRODCFT1)
   SIZE(79872)
   INITSIZE(32768)
   PREFLIST(FACIL01,FACIL02)
```

Activating a CFRM policy

After you define a CFRM policy, you must activate that policy. Use the following MVS command:

```
SETXCF START,POLICY,POLNAME=policyname,TYPE=CFRM
```

Activating a CFRM policy that contains a definition of a list structure does not create the structure. The structure is created the first time an attempt is made to connect to it, which occurs when the first coupling facility data table (CFDT) server that refers to the corresponding pool is started.

When the CFDT server creates a list structure, the structure is allocated with an initial size, which can be increased up to a maximum size, as specified in the CFRM policy. All structure sizes are rounded up to the next storage increment for the coupling facility level (CFLEVEL) at allocation time. For example, sizes are rounded up to the nearest 1 MB for CFLEVEL 16.

Provided that space is available in the coupling facility, you can dynamically expand a list structure from its initial size up to its maximum size, or contract it to free up coupling facility space for other purposes.

If the initial structure allocation becomes full, the structure does not expand automatically, even if the structure allocated is less than the specified maximum size. To expand a list structure when the allocation becomes full, you can expand it (up to its maximum size) using the following **SETXCF** command:

```
SETXCF START,ALTER,STRNAME=DFHCFLS_poolname,SIZE=nnnn
```

If you dynamically increase the size of a list structure in this way, also update the CRFM **INITSIZE** parameter to reflect the new size, so that the structure does not revert to its original size if you subsequently re-create or reload it.

Storage calculations for coupling facility data tables:

You can use the z Systems Coupling Facility Structure Sizer tool (CFSizer) to calculate storage requirements for data tables list structures in a coupling facility.

A coupling facility structure contains both stored data and the information needed to manage and access that data, in a similar way to a key-sequenced data set. The data for each entry in the coupling facility is stored as a chain of fixed-size (usually 256-byte) elements, which means that the exact length for variable-length data must be stored separately. To do this, CICS includes a length prefix in the stored data, so space calculations must allow for each entry using a whole number of elements. The amount of internal control information depends on the level of functionality and performance of the coupling facility control code for the current coupling facility level (CFLEVEL). The storage requirements can increase for a higher CFLEVEL. For more information, see “Coupling facility storage management” on page 224.

CFSizer is a web-based application that takes these factors into account and communicates with a coupling facility at a current CFLEVEL to calculate storage requirements. See CFSizer.

To use CFSizer to calculate storage requirements for data table list structures, enter the following information:

Maximum number of tables
The maximum number of data tables that can be stored in the data tables structure. This value corresponds to the MAXTABLES server parameter. See "List structure parameters" on page 179.

Specify a large enough number to handle all your data tables, but not so large that unused preallocated list headers use a large amount of coupling facility storage. The valid range is from 1 to 999999. The default is 1000.

Average rounded record size
The average amount of storage required for each data record. Each record has a two-byte length prefix and is stored as one or more 256-byte elements. This value determines the entry to element ratio that is used to calculate the required structure size. The valid range is from 1 to 32768. The default is 512.

If all data records are similar sizes, calculate this value by taking the average data size, adding two, and rounding up to the next multiple of 256. The amount of element storage required is two bytes more than the data record size because of the length prefix on each item.

If data records are different sizes, round up each size first before you take the average. For example, if half the records are 100 bytes and half are 300, round up the sizes to 256 and 512 respectively, then average them. The resulting average rounded item size is 384, which is more accurate than using the average item size of 200 and then rounding it up to 256.

Records in a contention model data table with a record length of up to 63 bytes are a special case. These records are stored in the entry adjunct area, so for average record length purposes, the data length is effectively zero.

Total records
The total number of records to be stored in all data tables.

Target usage percent
The percentage of the structure space that the given total number of items are expected to use. Specify a number in the range of 1 to 100. The default is 75. This value ensures the following:

- Free space exists for temporary expansion.
- If the initial free space is not enough, there is time to expand the structure in response to warning messages (which normally start at 80%).
- Activity to alter entry to element ratios is reduced.

Maximum expansion percent
The percentage that the structure can expand. If a non-zero value is specified, the maximum structure size will be greater than the initial structure size by an amount such that the total amount of data can increase by this percentage. For example, if the value 200 is specified, the initial size is enough to store the specified total number of items, and the maximum size is enough to store three times that number of items.

You use the results of these calculations to set the coupling facility resource manager (CFRM) **INITSIZE** and **SIZE** parameters in the CRFM policy.

For information about the CICS reserved space parameters that enable the coupling facility data table (CFDT) server to avoid a structure full condition, see "Reserved space parameters" on page 184 and "Avoiding structure full conditions" on page 185.

Defining and starting a coupling facility data table server region

You activate a coupling facility data table pool in an MVS image by starting up a coupling facility data table server region for that pool.

About this task

You can start the server as a started task, started job, or as a batch job. The job or task must invoke the coupling facility data table server region program, DFHCFMN, from an APF-authorized library. DFHCFMN is in the CICS authorized library, CICSTS54.CICS.SDFHAUTH.

Procedure

1. Specify the DFHCFMN program either in a SYSIN data set defined in the JCL, or in the **PARM** parameter on the EXEC statement.
2. Specify the mandatory and optional startup parameters for the DFHCFMN program. If you specify a startup parameter in both the SYSIN data set and the **PARM** parameter, the **PARM** value overrides the SYSIN value because the **MVS START** command can override the **PARM** value.
   a. You must specify a SYSPRINT DD statement for the print file.
   b. You must specify a SYSIN DD statement for the server parameters.
   c. You must specify the TS pool name.
   d. You must concatenate the license activation data set (the SDFHLIC library) to the STEPLIB DD statement.
   e. It is recommended that you specify the **REGION** parameter. This parameter ensures that the coupling facility data table server region has enough storage to process the maximum number of data table requests that can run concurrently.
   f. It is recommended that you specify TIME=NOLIMIT. The server task remains in a wait during most normal processing, because server processing is performed under the TCB of the client CICS region. If you omit this parameter, your server job could fail with abend S522 (wait limit exceeded), depending on the JWT value specified in the SMFPRM*xx* member of SYS1.PARMLIB.
   g. Specify additional parameters as required. For example, you might want to control the maximum number of queues that are to be supported in the pool and the number of buffers the server is to allocate.

Results

Tip: The easiest way to ensure that all pool-related parameters are consistent across MVS images is to use the same SYSIN parameter data set, or an identical copy of it, for all servers accessing the same pool, and to specify in the PARM field any parameters that vary between servers.

Coupling facility data table server JCL example

```
//PRODCFD1 JOB  ...
//CFSERVER EXEC PGM=DFHCFMN,REGION=40M,TIME=NOLIMIT    CICS CFDT Server
//STEPLIB  DD   DSN=CICSTS54.CICS.SDFHAUTH,DISP=SHR    Authorized library
//         DD   DSN=CICSTS54.CICS.SDFHLIC,DISP=SHR     License activation data set
//SYSPRINT DD   SYSOUT=*                               Messages and statistics
//SYSIN    DD   *
POOLNAME=PRODCFD1                                     Pool name
MAXTABLES=100                                         Allow up to 100 tables
/*
```

Figure 39. Sample JCL to start a CFDT server address space

Coupling facility data table server parameters

Parameters are specified in the form KEYWORD=*value*, where keywords can optionally be specified in mixed case to improve readability.

If you specify more than one parameter in the PARM field or on the same SYSIN input line, the parameters must be separated by a comma. Any text following one or more spaces is taken as a descriptive comment. Any parameter line which starts with an asterisk or a space is assumed to be a whole line comment.

You can enter some parameter keywords in more than one form, such as in abbreviated or truncated form.

The main parameters are listed on the server print file during start-up.

The parameters are all valid as initialization parameters (in the SYSIN file or PARM field), and some can also be modified by the SET command.

REGION parameter:

The JCL **REGION** parameter ensures that the coupling facility data table server region has enough storage to process the maximum number of data table requests that can run concurrently.

The number of coupling facility data table requests that each connected CICS region can have active at a time is limited to about 10. Each request requires about 40 KB, therefore the **REGION** size should specify at least 400 KB for each connected CICS region, plus a margin of about 10% for other storage areas. Thus, for a server supporting up to five CICS regions, specify `REGION=2200K`.

During server initialization, the server acquires all the available storage above 16 MB, as determined by the **REGION** parameter, then releases 5% of it for use by operating system services. It also acquires 5% of the free storage below 16MB for use in routines that require 24-bit addressable storage, for example sequential file read and write routines.

After initialization, the server uses AXM page allocation services to manage its storage. Server statistics indicate how much storage is allocated and used within the storage areas above and below 16 MB, which are called AXMPGANY and AXMPGLOW in the statistics.

If a task in the server region or a cross-memory request runs out of storage, this is likely to result in AXM terminating that task or request using a simulated abend with system completion code 80A to indicate a GETMAIN failure. Although the

server can usually continue processing other requests in this case, running out of storage in a critical routine can cause the server to terminate. Therefore, it is best to ensure that the **REGION** size is large enough to eliminate this risk.

Pool name parameter:

This parameter, POOLNAME, is always required:

POOLNAME=*name*
specifies the 8-character name of the coupling facility data table pool. This is appended by the server to the prefix DFHCF to create its own server name, as in DFHCF.*poolname*, and also to the prefix DFHCFLS_ to create the name of the coupling facility list structure, as in DFHCFLS_*poolname*.

This parameter is valid only at server initialization, and must always be specified.

This keyword can be abbreviated to **POOL**.

Security parameters:

You can use the security parameters to specify whether to use the optional security mechanism that the server provides, to check that CICS regions are authorized to open a coupling facility data table. You can also use these parameters to override standard processing for this optional security.

SECURITY={YES|NO}
Specifies whether individual coupling facility data table security checks are required.

YES The server performs a security check against each CICS region that attempts to open a coupling facility data table. Access is controlled through profiles defined in the general resource class named on the **SECURITYCLASS** parameter.

This function requires an external security manager, such as RACF, that supports the FASTAUTH function in cross-memory mode.

NO The server does not perform a security check against each CICS region that attempts to open a coupling facility data table.

This is the only security check performed by the server that is optional. The other file security checks are always performed by the server, as described in Security for coupling facility data tables.

This parameter is valid only at server initialization.

This keyword can be abbreviated to **SEC**.

SECURITYCLASS={FCICSFCT|*class***}**
Specifies the name of the RACF general resource class that the server uses for security checks on coupling facility data table access by CICS regions. The name can be up to 8 characters, and is the name of the class in which the CFDT resource profile and its access list are defined.

This parameter is valid only at server initialization.

This keyword can be abbreviated to **SECCLASS**.

SECURITYPREFIX={NO|YES}
When SECURITY=YES is specified, specifies whether the resource name that is passed to RACF for the coupling facility data table security check is prefixed with the server region user ID.

Note: For this security check, the resource name used by the server is the either the name specified on the TABLENAME attribute of the CICS file resource definition, or the FILE name if TABLENAME is not specified.

YES The server prefixes the resource name with the server region user ID (the default) or an alternative prefix specified on the **SECURITYPREFIXID** parameter.

NO The server passes to RACF only the 8-character resource name, without any prefix.

This parameter is valid only at server initialization.

This keyword can be abbreviated to **SECPREFIX** or **SECPRFX**.

SECURITYPREFIXID=*identifier*
Specifies a prefix for the server to use for security checks on coupling facility data table access by CICS regions, instead of the server region user ID. The prefix can be up to 8 characters, and should correspond to the prefix used to define profile names of CFDTs to RACF. This parameter is effective only when SECURITYPREFIX=YES is specified.

This parameter is valid only at server initialization.

This keyword can be abbreviated to **SECPREFIXID**.

Statistics parameters:

Use the following parameters to specify server statistics options:

ENDOFDAY={00:00|*hh:mm***}**
specifies the time of day, in hours and minutes, when the server is to collect and reset end-of-day statistics.

Note: If the STATSOPTIONS parameter specifies NONE, the server still writes end-of-day statistics to the print file.

The valid range of times is from 00:00 to 24:00.

This keyword can be abbreviated to **EOD**.

STATSINTERVAL={03:00|*hh:mm***}**
specifies the statistics collection interval, in the range 1 minute to 24 hours. This parameter is ignored if the STATSOPTIONS parameter specifies NONE.

The time interval can range from 00:01 to 24:00.

This keyword can be abbreviated to **STATSINT**.

STATSOPTIONS={NONE|SMF|PRINT|BOTH}
specifies whether the server is to produce interval statistics, and the destination for the statistics it produces.

NONE
The server does not produce any interval statistics.

SMF The server produces interval statistics and writes them to the current SMF data set only.

PRINT
The server produces interval statistics and writes them to the server's print file only.

BOTH The server produces interval statistics and writes them to the current SMF data set and to the server's print file.

This keyword can be abbreviated to **STATSOPT**.

Automatic restart manager (ARM) parameters:

During server initialization, the server unconditionally registers with ARM except when the server program is invoked with either the UNLOAD or the RELOAD functions. The server will not start if the registration fails.

Use the following parameters to override default processing for the automatic restart manager:

ARMELEMENTNAME=*elementname*
specifies the automatic restart manager element name, up to 16 characters, to identify the server to ARM for automatic restart purposes. The permitted characters for the element name are A to Z 0-9 $ # @ and the underscore symbol (_).

The default identifier is of the form DFHCF*nn_poolname*, where CF represents the server type, *nn* is the &SYSCLONE value for the system (which can be either one or two characters), and *poolname* is the name of the pool served by the server.

This parameter is only valid at server initialization.

This keyword can be abbreviated to **ARMELEMENT** or **ARMELEMNAME**.

ARMELEMENTTYPE=*elementtype*
specifies the automatic restart manager element type, up to 8 characters for use in ARM policies as a means of classifying similar elements. The permitted characters for the element type are A to Z 0-9 $ # and @.

The default element type is SYSCICSS.

This parameter is only valid at server initialization.

This keyword can be abbreviated to **ARMELEMTYPE**.

List structure parameters:

The list structure parameters specify the attributes that are used only for initial allocation of the data tables list structure for a for a coupling facility data table (CFDT) pool. Initial allocation occurs the first time a server is started for the CFDT pool.

The list structure parameters are as follows:

MAXTABLES={1000|*number*}
Specifies the maximum number of data lists to be reserved when the structure is allocated, which determines the maximum number of tables that can be stored in the structure. This parameter also determines how many list headers are defined when the structure is created.

Specify a large enough number to handle all your data tables, but not so large that unused preallocated list headers use a large amount of coupling facility storage.
You cannot change this number without reallocating the structure, which means first deleting the existing structure (see "Deleting or emptying coupling facility data table pools" on page 192). Therefore, if the structure is being allocated at less than its maximum size, the value here should be based on the maximum possible size of the structure rather than its initial size.
This parameter corresponds to the "Maximum number of tables" value in the

CFSizer tool (see “Storage calculations for coupling facility data tables” on page 173).
This parameter is valid only at server initialization and is used only when the structure is first allocated. The valid range is from 1 to 999999.
You can abbreviate this keyword to **MAXT**.

Note: In addition to the data lists defined by **MAXTABLES** there are control lists used internally by CICS so the total number of lists is greater than the number specified in **MAXTABLES**.

POOLSIZE={0|*number*{K|M|G}}
Specifies the initial amount of coupling facility storage to be allocated for the pool list structure, expressed as kilobytes (*n* K), megabytes (*n* M) or gigabytes (*n* G).

Usually, you can omit this parameter and specify the structure size by using the **INITSIZE** parameter in the coupling facility resource manager (CFRM) policy. However, this parameter can be useful if the structure is reallocated or reloaded but the CFRM policy has not been updated to reflect the required size.

0 The special value 0 means that the server obtains an initial allocation using the parameters specified in the CFRM policy. If the CFRM policy specifies an **INITSIZE** value for the structure, this determines the initial allocation. Otherwise, the CFRM **SIZE** value (the maximum size of the structure) is allocated.

number
A non-zero value specifies an initial amount of storage to be allocated, overriding the **INITSIZE** parameter in the CFRM policy. This value is rounded up by MVS to the next storage increment for the coupling facility level (CFLEVEL). For example, for CFLEVEL 16, the value is rounded up to the nearest 1 MB.

The value must be less than the CFRM **SIZE** parameter, otherwise the value of **POOLSIZE** is ignored and the initial allocation uses the parameters specified in the CFRM policy.

The valid range is from 0 to 16777215M. However, you must specify a value that is less than the maximum size of a structure in z/OS, otherwise a z/OS error occurs. For example, in z/OS, Version 1 Release 12, the maximum size of a structure is 1048576 MB (1 TB). For more details, see the information about CFRM parameters in *z/OS MVS Setting Up a Sysplex*.

This parameter is valid only at server initialization and is used only when the structure is first allocated.

Debug trace parameters:

These parameters are provided only for intensive debug tracing.

Using these options in a production environment could have a significant impact on performance and cause the print file to grow very rapidly, using up spool space.

Trace messages from cross-memory requests can be lost if they are generated faster than the trace print subtask can print them. In this event, the trace only indicates how many messages were lost.

CFTRACE={OFF|ON}
specifies the coupling facility interface debug trace option.

OFF Coupling facility interface debug trace is disabled.

ON Coupling facility interface debug trace produces trace messages on the print file, indicating the main parameters to the coupling facility request interface, and the result from the IXLLIST macro.

This keyword can also be specified as **TRACECF**.

RQTRACE={OFF|ON}
specifies the table request debug trace option.

OFF Table request debug trace is disabled.

ON Table request debug trace produces trace messages on the print file, indicating the main parameters on entry to each cross-memory request, and the results on exit.

This keyword can also be specified as **TRACERQ=**.

Tuning parameters:

These parameters are provided for tuning purposes, but usually you can omit them and let the coupling facility data table (CFDT) server use the default values.

ELEMENTRATIO={1|*number*}
An MVS coupling facility resource management (CFRM) parameter that specifies the element part of the entry-to-element ratio when the structure is first allocated. This value determines the proportion of the structure space that is initially set aside for data elements.

The ideal value for the entry-to-element ratio is the average size of data for each entry divided by the element size. However, the server automatically adjusts the ratio according to the actual entry and element usage.

This parameter is valid only at server initialization, and is used only when the structure is first allocated. The valid range is from 1 to 255.

You can abbreviate this keyword to **ELEMRATIO**.

For further information about this parameter in the coupling facility, see *z/OS MVS Programming: Sysplex Services Guide*.

ELEMENTSIZE={256|*size*}
An MVS CFRM parameter that specifies the data element size for the list structure, which must be a power of 2. This parameter is used only for coupling facility log streams. For current coupling facility implementations, there is no reason to use any value other than the default value of 256.

This parameter is valid only at server initialization and is only used when the structure is first allocated. The valid range is from 256 to 4096.

You can abbreviate this keyword to **ELEMSIZE**.

ENTRYRATIO={1|*number*}
An MVS CFRM parameter that specifies the entry part of the entry-to-element ratio when the structure is first allocated. This value determines the proportion of structure space that is initially set aside for list entry controls.

It is not essential to specify this parameter because the server automatically adjusts the ratio, based on actual usage, to improve space utilization if necessary.

This parameter is valid only at server initialization and is used only when the structure is first allocated. The valid range is from 1 to 255.

For further information about this parameter in the coupling facility, see *z/OS MVS Programming: Sysplex Services Guide*.

Lock wait parameters:

Use these parameters to modify the time intervals for the server lock wait retry mechanism.

This is provided mainly as a "wake-up" mechanism to ensure that requests waiting for a record lock do not wait indefinitely in certain rare cases where a system failure or structure full condition could cause a lock release notification message to be lost. In addition, this mechanism also ensures that if a CICS task is purged while it has a request waiting for a lock, the waiting request in the server is woken up as soon as the lock wait retry interval expires. The request process then finds that the CICS task is no longer waiting for it, therefore it terminates rather than continuing to wait and hold on to server resources until the lock becomes free.

There are two times involved: a scan time interval and a wait time. The server starts its lock scan interval timing when the first request is made to wait.

This mechanism has very little effect on normal processing and the default lock wait retry parameter values are designed to suit the majority of installations.

LOCKSCANINTERVAL={5|*number*}
specifies the time interval after which requests waiting for record locks are scanned to check for lock wait timeout.

This affects the overall duration of the lock wait timeout, because a request that starts waiting for a lock during a given scan interval is timed as if from the start of the interval. The lock scan interval should be less than the lock wait interval, and ideally should be such that the lock wait interval is an exact multiple of the lock scan interval.

You can specify this value as a number of seconds or in the time format *hh:mm:ss*.

The valid range is from 1 (1 second) to 24:00 (24 hours).

This keyword can be abbreviated to **LOCKSCANINT**.

LOCKWAITINTERVAL={10|*number*}
specifies the maximum time that a request should wait for a record lock before being reinvoked to retry. The actual time a request waits depends on how far into a scan interval it started its wait. For example, in a server using the scan and wait default intervals, if 4 seconds have already elapsed when a request starts to wait, the maximum time it can wait is 6 seconds. When the server checks the timeout value for the request, it assumes that the request has been waiting for the full scan period. Thus, for the default values, a request is reinvoked to retry after waiting between five and ten seconds.

This value can either be specified as a number of seconds or in time interval format *hh:mm:ss*.

The valid range is from 1 (1 second) to 24:00 (24 hours).

This keyword can be abbreviated to **LOCKWAITINT**.

Warning parameters:

Use these parameters to modify the thresholds at which warning messages are issued, and an automatic structure alter occurs, when the structure becomes nearly full.

ELEMENTWARN={80|*number*}
specifies the percentage of list structure elements in use at which warning messages and an automatic structure alter should be first triggered.

The valid range is from 1 to 100 per cent.

This keyword can be abbreviated to **ELEMWARN**.

ELEMENTWARNINC={5|*number*}
specifies the percentage increase (or decrease) of elements in use before the next warning message should be triggered (reduced to 1 when the next increase would otherwise reach 100). After the first warning, additional messages are issued as the number of elements increases and decreases. These messages stop when the number of elements in use has fallen at least this percentage below the initial warning level.

The valid range is from 1 to 100 per cent.

This keyword can be abbreviated to **ELEMWARNINC**.

ENTRYWARN={80|*number*}
specifies the percentage of list structure entries in use at which warning messages and an automatic structure alter action should be first triggered.

The valid range is from 1 to 100 per cent.

ENTRYWARNINC={5|*number*}
specifies the percentage increase (or decrease) of entries in use before the next warning message should be triggered (reduced to 1 when the next increase would otherwise reach 100). After the first warning, additional messages are issued as the number of elements increases and decreases. These messages stop when the number of entries in use has fallen at least this percentage below the initial warning level.

The valid range is from 1 to 100 per cent.

Automatic structure alter parameters:

Use these parameters to modify the conditions under which the server attempts an automatic alter when the structure becomes nearly full.

For information about the structure alter process, see "Coupling facility data table server automatic structure alter" on page 186.

ALTERELEMMIN={100|*number*}
specifies the minimum number of excess elements that must be present to cause the server to issue an automatic alter to convert those elements to entries.

The valid range is from 0 to 999999999.

ALTERELEMPC={5|*number*}
specifies the minimum percentage of excess elements that must be present to cause the server to issue an automatic later to increase the proportion of entries.

The valid range is from 0 to 100 per cent.

ALTERENTRYMIN={100|*number*}
specifies the minimum number of excess entries that must be present to cause the server to issue an automatic alter to convert those entries to elements.

The valid range is from 0 to 999999999.

ALTERENTRYPC={5|*number*}
specifies the minimum percentage of excess entries that must be present to cause the server to issue an automatic alter to increase the proportion of elements.

The valid range is from 0 to 100 per cent.

ALTERMININTERVAL={00:10|hh:mm}
specifies the minimum time interval to be left between automatic structure alter attempts when the structure is nearly full (above the element or entry warning level). This is to prevent excessive numbers of alter attempts to try to optimize space when the structure is nearly full.

The valid range is from 00:00 to 24:00.

This keyword can be abbreviated to **ALTERMININT**.

Reserved space parameters:

Use these parameters to control the amount of structure space that is reserved for rewrites and server internal operations (such as tracking units of work and notifying other servers when a lock is released).

For information about the effect of these parameters, see "Avoiding structure full conditions" on page 185.

ELEMENTRESERVEMIN={300|*number*}
specifies the minimum number of list structure data elements (normally 256 bytes each) to be reserved for rewrites and server internal operations.

The valid range is from 0 to 999999999.

This keyword can be abbreviated to **ELEMRESERVEMIN**.

ELEMENTRESERVEPC={5|*number*}
specifies the percentage of list structure data elements to be reserved for rewrites and internal server use. If this percentage evaluates to less than the minimum number specified on the ELEMENTRESERVEMIN parameter, the minimum number is used instead.

The valid range is from 0 to 100.

This keyword can be abbreviated to **ELEMRESERVEPC** or **ELEMRESPC**.

ENTRYRESERVEMIN={100|*number*}
specifies the minimum number of list structure entries to be reserved for rewrites and server internal operations.

The valid range is from 0 to 999999999.

This keyword can be abbreviated to **ENTRYRESMIN**.

ENTRYRESERVEPC={5|*number*}
specifies the percentage of list structure elements to be reserved for rewrites and internal server use. If this percentage evaluates to less than the minimum number specified on the ENTRYRESERVEMIN parameter, the minimum number is used instead.

The valid range is from 0 to 100.

This keyword can be abbreviated to **ENTRYRESPC**.

Avoiding structure full conditions

If the coupling facility data table structure is allowed to become completely full, this not only prevents the addition of new records or tables, but can also have a significant impact on performance and application function.

In particular, rewrite requests can be rejected even when the size of the new data is less than or equal to the original, and server internal operations can fail, causing internal timeouts and retries.

The parameters ELEMENTRESERVEMIN, ELEMENTRESERVEPC, ENTRYRESERVEMIN and ENTRYRESERVEPC are provided to reduce the risk of the structure becoming totally full, by reserving a number of entries and elements, which can only be used for operations that normally only need extra space temporarily, such as rewrites or unit of work control operations. If a server is asked to write a new record or create a new table when the number of entries or elements remaining in the structure (as returned by each coupling facility access request) is less than or equal to the specified reserve level, the request is rejected with an indication that no space is available. Before rejecting the request, the server issues a dummy read request in order to find out the latest usage levels for the structure, in case more space has recently become available.

Using the reserved space parameters means that, even if the structure fills up very rapidly (for example, because a table is being loaded that is too large for the available space), enough space should remain to allow rewrites of existing records and allow internal communication between servers to continue normally.

Note that this mechanism cannot prevent the structure from eventually becoming totally full, as recoverable rewrites are allowed to use the reserved space temporarily, and rewrites that increase the data length will gradually use up the reserved elements. If action is not taken to prevent the structure from becoming totally full, the following effects can occur:

- An attempt to close a table or change the table status could encounter a temporary structure full condition. In this case, the attempt is retried indefinitely, because it must be completed in order to preserve table integrity (the only alternative being to terminate the server). The retry process normally succeeds quickly, but there is a theoretical case where this can cause a loop until another server causes temporarily unavailable resources to be released.
- Rewrites with the same (or smaller) data size for a table using the contention update model are retried indefinitely if they initially fail because of a structure full condition. This is done to protect the application against having to handle this unexpected form of failure. Again, the retry should normally succeed quickly, but there is a theoretical possibility that this could loop for a while.
- Rewrites for a table using the locking or recoverable update model could be rejected with a structure full condition even if the data size is not increased. No retry is attempted in this case.
- Units of work can be backed out because the server is unable to create unit of work control entries for commit processing.
- There may not be sufficient structure space to send lock release messages, in which case waiting tasks are not woken up immediately but continue to wait for up to the timeout interval specified on the LOCKWAITINTERVAL parameter before finding out that the lock has been released.

Coupling facility data table server automatic structure alter

The coupling facility data table server monitors the total number of elements and entries in use in the structure, using information returned by the coupling facility on every request.

When the numbers in use exceed the specified warning thresholds, the server issues a warning message, and this warning message is repeated each time the number in use increases beyond further thresholds.

Each time the server issues a warning, it also tests whether an automatic structure alter for the entry-to-element ratio should be issued. If any form of alter has already been issued recently (by any server or through an operator `SETXCF ALTER` command) and the structure space usage has remained above warning levels since the previous attempt, any further structure alter attempt is suppressed until at least the minimum interval (specified through the ALTERMININTERVAL parameter) has elapsed.

The server checks whether an automatic structure alter should be issued, by calculating how many excess elements or entries will be left when the other runs out completely, based on the ratio between the current numbers of elements and entries in use. If the number of excess elements or entries exceeds the number specified in the ALTERELEMMIN or ALTERENTRYMIN parameter, and the same number expressed as a percentage of the total exceeds the value specified in the ALTERELEMPC or ALTERENTRYPC parameter, an IXLALTER request is issued to alter the entry to element ratio to the actual current ratio between the number of entries and elements in use.

Only one alter request can be active at a time for a given structure. This means a server may well find that another server has already started the structure alter process, in which case its own alter is rejected. However, the system automatically notifies all servers when the structure alter is completed, giving the new numbers of elements and entries so that each server can update its own status information.

Controlling coupling facility data table server regions

You can issue commands to control a coupling facility data table server, using the **MVS MODIFY (F)** command to specify the job or started task name of the server region, followed by the server command.

About this task

The general form of a coupling facility data table server command, using the short form F, is as follows:

```
F job_name,command parameters... comments
```

You can use the following commands to control the coupling facility data table server region.

Procedure

- To modify the server initialization parameters, use the **SET** command:

  ```
  SET keyword=operand[,keyword=operand,...]
  ```

  The command can be abbreviated to **T**, as for the **MVS SET** command. See "The SET command options" on page 187 for details.

- To display the values of one or more parameter values or statistics summary information on the console, use the **DISPLAY** command:

  ```
  DISPLAY keyword[=operand][,keyword[=operand,]...]
  ```

  The valid keywords for **DISPLAY** are all the initialization parameters, plus an additional set described under "DISPLAY and PRINT command options" on page 188.

  The command can be abbreviated to **D**, as for the **MVS DISPLAY** command.
- To print the output that the **DISPLAY** command produces, use the **PRINT** command:

  ```
  PRINT keyword[=operand][,keyword[=operand,]...]
  ```

  The command produces the same output as DISPLAY, supporting the same keywords, but on the print file only.
- To delete a table, use the **DELETE TABLE**=*name* command. The table must not be in use for this command to succeed. You can abbreviate the command to **DEL**.
- To stop the server normally, use either the **STOP** command or the **MVS STOP** command. The server waits for any active connections to end first and prevents any new connections while it is waiting. The command can be abbreviated to **P**; for example `P jobname`.
- To request an automatic restart, use the **CANCEL** command. This command terminates the server immediately. You can specify whether the server will automatically restart. For information about **CANCEL RESTART** see "The CANCEL command options" on page 166.
- The server also responds to XES events such as an operator **SETXCF** command to alter the structure size. If the server can no longer access the coupling facility, it automatically issues a server **CANCEL** command to close itself down immediately.

The SET command options

You can use the SET command to modify groups of server initialization parameters.

These system initialization parameter groups are:

- The statistics parameters
- The debug trace parameters
- The lock wait parameters
- The warning parameters
- The automatic ALTER parameters.

The following **SET** keywords are used to modify the server's recovery status of an inactive CICS region that had unresolved units of work when it last terminated:

RESTARTED=*applid*
: Establish a temporary recoverable connection for the given APPLID. This resolves any units of work that were in commit or backout processing when the region last terminated, and indicates whether there are any remaining indoubt units of work.

  This keyword can be abbreviated to **RESTART** or **REST**.

COMMITTED={*applid*|*applid.uowid***}**
: Establish a temporary recoverable connection for the specified APPLID and commit all indoubt units of work, or, if *uowid* is also specified, commit that specific unit of work.

This command should be used **only** when it is not possible to restart the original CICS region to resolve the work normally, because it can result in inconsistency between coupling facility data table resources and other CICS resources updated by the same unit of work.

This keyword can be abbreviated to **COMMIT** or **COMM**.

BACKEDOUT={*applid*|*applid.uowid***}**
Establish a temporary recoverable connection for the specified APPLID and back out all indoubt units of work, or, if *uowid* is also specified, back out that specific unit of work.

This command should be used *only* when it is not possible to restart the original CICS region to resolve the work normally, because it can result in inconsistency between coupling facility data table resources and other CICS resources updated by the same unit of work.

This keyword can be abbreviated to **BACKOUT** or **BACK**.

Use the following **SET** parameters to modify options relating to a specific table:

TABLE=*name*
specifies the table to which the following table-related parameters in the same command are to be applied. This parameter is required before any table-related parameters.

MAXRECS=*number*
Modify the maximum number of records that can be stored in the table specified by the preceding **TABLE** parameter.

If the maximum number is set to a value less than the current number of records in the table, no new records can be stored until records have been deleted to reduce the current number to within the new maximum limit. For a recoverable table, this also means that records cannot be updated, because the recoverable update process adds a new record on the rewrite operation then deletes the original record when the transaction completes.

This keyword can also be specified as **MAXNUMRECS**.

AVAILABLE={YES|NO}
Specify whether the table named by the preceding **TABLE** parameter is available for new OPEN requests. If the table is made unavailable, a CICS region that subsequently issues an OPEN request for the table receives a response indicating that it is unavailable, but regions that currently have the table open are not affected. Even when a table is marked as unavailable, a server can implicitly open it on behalf of a CICS region to allow recoverable work to be resolved during restart processing.

This keyword can be abbreviated to **AVAIL**.

Examples of the SET command: The following example changes the statistics options:

```
SET STATSOPT=BOTH,EOD=21:00,STATSINT=06:00
```

The following example modifies the maximum number of records allowed in the specified table:

```
SET TABLE=PAYECFT1,MAXRECS=200000
```

DISPLAY and PRINT command options

You can use the DISPLAY (and PRINT) commands to display the values of any initialization parameters plus some additional information.

Some of the parameters that provide additional information support generic names. You specify generic names using the following wildcard characters:

- An * (asterisk symbol). Use this anywhere in the parameter value to represent from 0 to 8 characters of any value. For example, CICSH* to represent all the CICS APPLIDs in a CICSplex identified by the letter H.
- A % (per cent symbol). Use this anywhere in the parameter value to represent only one character of any value. For example, CICS%T* to represent all the TOR APPLIDs in all CICSplexes.

The parameters supported by the DISPLAY and PRINT commands are as follows:

APPLIDS
Display the APPLID and MVS system name for every CICS region that currently has a recoverable connection to the pool. This command returns information not only for the server to which the MODIFY command is issued, but for all other servers connected to the same pool.

This keyword can be abbreviated to **APPLID**, **APPLS** or **APPL**.

APPLID={*applid*|*generic***}**
Display the APPLID and MVS system name for each region that currently has a recoverable connection to the server's pool, and whose APPLID matches *applid* or *generic*. This command returns information not only for the server to which the MODIFY command is issued, but for all other servers connected to the same pool.

applid Use this for a specific APPLID, which should match only one region in the sysplex.

generic Use a suitable generic value when you want to obtain information about several regions.

If *applid* or *generic* is not specified, the server treats this as equivalent to the command DISPLAY APPLIDS.

This keyword can also be specified as **APPLIDS**, **APPLS** or **APPL**.

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

CONNECTIONS
Display the jobnames and applids of the regions currently connected to the server to which the command is issued.

This keyword can be abbreviated to **CONN**.

TABLES
Display the names of all tables currently allocated in the pool.

TABLE={*name*|*generic_name***}**
Display information about the attributes and status of a specific table, or of a set of tables whose names match the generic name.

If no table name is specified, this is treated as equivalent to DISPLAY TABLES.

TABLEUSERS
Display the CICS APPLIDs of the regions that are currently using each of the tables currently defined in the pool.

This keyword can be abbreviated to **TABLEU**.

TABLEUSERS={*name*|*generic_name***}**
Display the CICS APPLIDs of the regions that are currently using the specified table, or using each of the set of tables whose names match the generic name.

If no table name is specified, this is treated as equivalent to DISPLAY TABLEUSERS.

This keyword can be abbreviated to **TABLEU**

UOWIDS
Display the applids of all regions that currently have unresolved recoverable units of work, together with the number of units of work that are currently in doubt, or are in the process of being committed or backed out. The information displayed does not include units of work that have not yet started the resolution process; that is, units of work that are still in flight.

This keyword can be abbreviated to **UOWS**.

UOWIDS={*applid*|*generic_applid***}**∨**{***applid***.*|*generic_applid***.*}**
Display, for the specified regions if they currently have unresolved recoverable units of work, information about those units of work. The information displayed does not include units of work that have not yet started the resolution process; that is, units of work that are still in flight. The information returned depends on the form of operand used.

applid|*generic_applid*
This form of operand displays the number of units of work that are currently in doubt, or are in the process of being committed or backed out.

If you specify *applid*, the server displays UOW information for a specific APPLID, which should correspond to only one region in the sysplex.

If you specify *generic_applid* the server displays UOW information for all the APPLIDs that match the generic APPLID specified.

*applid***.***|*generic_applid***.***
This form of operand displays:

- The state and local UOWID of each individual unit of work, followed by
- A summary of the number of units of work that are currently in doubt, or are in the process of being committed or backed out.

If you specify *applid*.*, the server displays the UOW information for a specific APPLID, which should correspond to only one region in the sysplex.

If you specify *generic_applid*.*, the server displays UOW information for all the APPLIDs that match the generic APPLID specified.

This keyword can be abbreviated to **UOWS**.

UOWID=*applid.uowid*
Display the state of an individual unresolved unit of work, identified by its applid and local unit of work ID (UOWID). Enter the local UOWID as 16 hexadecimal digits.

This keyword can be abbreviated to **UOW**.

DISPLAY and PRINT options for statistics summaries:

Use the following parameters to display or print statistics:

CFSTATS
Display statistics for coupling facility interface accesses and responses from the server.

This keyword can also be specified as **CFST** or **STATSCF**.

POOLSTATS
Display usage statistics for the pool list structure as a whole. This is based on information returned by coupling facility access requests, therefore it is only as current as the most recent request made through the server to which the command is issued.

This keyword can be abbreviated to **POOLST**.

TABLESTATS
Display statistics for requests, processed by the server to which the command is issued, for each table plus a summary of all requests processed, including those that are not table-specific, such as unit of work control.

Note that only tables with a non-zero number of requests since the start of the current statistics interval are shown.

This keyword can also be specified as **TABLEST**.

TABLESTATS={*name*|*generic_name***}**
Display request statistics for the specified table or tables.

name A specific table name in the pool accessed by the server. Returns statistics for this table only.

generic_name
A generic name that you can use to obtain statistics about a number of tables. Returns statistics for any table name that matches the generic name.

This keyword can be abbreviated to **TABLEST**.

STORAGESTATS
Display main storage allocation statistics for the server address space.

This keyword can be abbreviated to **STORAGEST** or **STGST**.

DISPLAY and PRINT options for combined lists of information:

These keywords represent combined lists of information:

PARAMETERS
Display the main parameter values. These are POOLNAME, SECURITY, SECURITYPREFIX, statistics options, and list structure options.

This keyword can be abbreviated to **PARM** or **PARMS**.

ALLPARAMETERS
Display all parameter values.

This keyword can be abbreviated to **ALLPARMS**.

STATISTICS
Display all available statistics.

This keyword can be abbreviated to **STAT** or **STATS**.

INITIALIZED
Display the parameters and statistics that are usually displayed when initialization is complete. This is equivalent to PARM, POOLSTATS, STGSTATS.

This keyword can be abbreviated to **INIT**.

ARM
Display all ARM-related parameter values:

- ARMELEMENTNAME
- ARMELEMENTTYPE
- ARMREGISTERED

This keyword can be coded as **ARMSTATUS**.

The CANCEL command options

You can use the CANCEL command to request an automatic restart.

Specify the following parameter:

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

If the server encounters an unrecoverable problem with the coupling facility connection, consisting either of lost connectivity or a structure failure, it cancels itself using the CANCEL RESTART=YES command. This terminates the existing connection and shuts down the server. A new instance of the server job is then started.

A server can also be restarted explicitly using either the server command CANCEL RESTART=YES or the MVS command CANCEL jobname,ARMRESTART.

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

Deleting or emptying coupling facility data table pools

You can delete a coupling facility data table pool using the MVS **SETXCF** command to delete its coupling facility list structure.

About this task

For example:

```
SETXCF FORCE,STRUCTURE,STRNAME=DFHCFLS_poolname
```

You can delete a structure only when there are no servers connected to the pool, otherwise MVS rejects the command.

When you attempt to start a server for a pool that has been deleted (or attempt to reload the pool), it is allocated as a new structure. The newly allocated structure uses size and location attributes specified by the currently active CFRM policy, and other values determined by the server initialization parameters (in particular, **MAXTABLES**).

Unloading and reloading coupling facility data table pools

You can unload, and reload, the complete contents of a coupling facility data table pool to and from a sequential data set by invoking the server program with the **FUNCTION** parameter, using the UNLOAD and RELOAD options.

About this task

You can use this function, for example, to do the following:

- Preserve the coupling facility data table pool during planned coupling facility maintenance, or
- Move the pool to a different coupling facility.
- Increase the size of the pool's list structure.

  If the maximum number of tables specified in the original pool was too small, or the pool has reached its maximum size and needs to be expanded further, unload the pool, then delete the structure so that the reload process can reallocate it with more space.

Alternatively, you can use the system-managed rebuild facility to dynamically move a structure to another coupling facility connected to the same system. This allows servers to remain active, but temporarily suspends processing while the rebuild is in progress. For more information, see "System-managed list structure rebuild" on page 230.

FUNCTION={UNLOAD|RELOAD}

Specify the function for which the server is being initialized.

UNLOAD

Unload the entire contents of the coupling facility data table pool specified on the POOLNAME parameter to a sequential data set. When the unload processing has completed (normally or abnormally) the server program terminates.

The UNLOAD function requires a DD statement for DDNAME DFHCFUL describing the sequential data set to which the table pool is to be unloaded. The format of the unloaded data set is:

RECFM=F
LRECL=4096
BLKSIZE=4096

You can obtain an estimate of the upper limit for the total size of the data set, in bytes, from the pool usage statistics produced by the server:

- From the statistics, multiply the number of elements in use by the element size (usually 256) to get a total number of bytes for the data size, although the space needed to unload the data is normally much less, because unused space in a data element is not unloaded.
- Add some space for the record keys, calculated using a two-byte prefix plus the keylength for each record, plus about 100 bytes per table for table control information. Thus, the maximum you should need for keys and control information is:

  ```
  (18 bytes x number of entries) + (100 bytes x number of tables)
  ```

RELOAD

Reload, into the coupling facility data table pool named on the POOLNAME parameter, a previously unloaded coupling facility data table pool.

You can reload a pool into a pool with a different name—it does not have to keep the same name as the original pool. When the reload processing has completed (normally or abnormally) the server program terminates.

The RELOAD function requires a DD statement for DDNAME DFHCFRL, describing the sequential data set from which the table pool is to be reloaded.

The structure is allocated, if necessary, during reloading, in which case you can use the same server parameters to control structure attributes as for normal server startup. The reload process bypasses any tables or units of work that are already found in the pool (for example, because the structure was too small and the reload job had to be restarted after using ALTER to increase the structure size).

Note: If the unloaded pool structure was altered dynamically at any time after initial allocation (by using the SETXCF command to increase the size), ensure that the increased size is allocated for the reloaded pool. The recommended way is to update the INITSIZE parameter for the structure in the current CFRM policy whenever you alter the structure size, and to activate the updated policy using the SETXCF START ,POLICY command. Alternatively, you can specify the required pool size in the POOLSIZE parameter in the reload JCL, but note that this does not override the CFRM INITSIZE parameter if it is exactly equal to the maximum pool size.

Note: If you omit the FUNCTION parameter, the server program initializes a coupling facility data table server address space.

For the UNLOAD and RELOAD function, the server program requires exclusive use of the list structure. If the structure is currently being used by a normal server, the unload or reload attempt is rejected. Similarly, if a normal server attempts to start while an unload or reload job is in progress, the attempt fails because shared access to the structure is not available.

You can specify all normal server parameters when unloading or reloading, but some of these (for example, security-related parameters) are ignored because they do not apply to unload or reload processing.

Note that when a pool is nearly full (with less than about 5% free entries and elements) there is no guarantee that it can be unloaded and reloaded into a structure of exactly the same size. This is because the amount of space available is affected by the current ratio of entries to elements, which is controlled only approximately by the automatic ALTER process.If the structure reaches the warning level during reloading, the automatic ALTER process attempts to adjust the entry to element ratio. The reload process automatically waits for the ALTER to complete if reloading runs out of space while an ALTER is still in progress.

If reloading fails because it runs out of space, the resulting messages include the numbers of tables reloaded and blocks read up to the time of the failure. You can compare these values with those in the messages from the original unload job, to determine how many more tables and how much more data remains to be loaded. If a table had been partially reloaded before running out of space, it is deleted so that the whole table is reloaded again if the reload is retried later. If reloading is interrupted for any other reason than running out of space, for example by an MVS system failure, reloading can still be restarted using the partially reloaded structure, but in that case the structure space occupied by any partially reloaded table will be unavailable, so it is normally better to delete the structure (using the MVS **SETXCF FORCE** command) and start reloading again with a newly allocated structure.

```
//UNLDCFD1 JOB  ...
//DTUNLOAD EXEC PGM=DFHCFMN        CICS CF data table server program
//STEPLIB  DD   DSN=CICSTS54.CICS.SDFHAUTH,DISP=SHR  Authorized library
//SYSPRINT DD   SYSOUT=*           Options, messages and statistics
//DFHCFUL  DD   DSN=CFDT1.UNLOADED.POOL,  Unloaded data table pool
//         DISP=(NEW,CATLG),
//         SPACE=(4096,(10000,1000))  Estimated size in 4K blocks
//SYSIN    DD   *
FUNCTION=UNLOAD                  Function to be performed is UNLOAD
POOLNAME=PRODCFD1                Pool name
/*
```

Figure 40. Unload JCL example

```
//RELDCFD1 JOB  ...
//DTRELOAD EXEC PGM=DFHCFMN        CICS CF data table server program
//STEPLIB  DD   DSN=CICSTS54.CICS.SDFHAUTH,DISP=SHR  Authorized library
//SYSPRINT DD   SYSOUT=*           Options, messages and statistics
//DFHCFRL  DD   DSN=CFDT1.UNLOADED.POOL,DISP=OLD  Unloaded table pool
//SYSIN    DD   *
FUNCTION=RELOAD                  Function to be performed is RELOAD
POOLNAME=PRODCFD1                Pool name
POOLSIZE=50M                     Increased pool size
MAXTABLES=500                    Increased max number of tables
/*
```

Figure 41. Reload JCL example

Setting up and running a region status server

You can use a CICS region status server to share CICS region status data in a sysplex rapidly to support optimized workload management. A region status server services only region status requests, rather than region status and user application requests.

About this task

CICS region status data is broadcast to the sysplex using a data table that is named after the hosting CICSplex for the region. Each region in the CICSplex is described by a single record in the CICSplex data table. The data tables are held in coupling facility structures, with access controlled by a coupling facility data table (CFDT) server. You must set up one CFDT server for each pool in an MVS image.

You can put related groups of region status tables in separate pools. For example, you might have one pool for production and another for test. A pool is defined as a list structure in the coupling facility resource management (CRFM) policy. The pool name is used to form the server name with the prefix DFHCF and is specified in the startup JCL for the server.

Note:

- You must use different pool names for your region status server if you have two separate CICSplexes of the same name in the sysplex.
- Ensure that the relevant authorization is provided. See Security for coupling facility data tables.

To set up and manage a region status server, use the following procedure.

Procedure

1. Ensure that you have a list structure for a region status server pool.

   For the best performance, define a new dedicated list structure for a region status server pool. For more information, see "Defining a list structure for a region status server."

   Optionally, you can use an existing CFDT pool to store your CICSplex data tables. However, the throughput of your optimized workloads might be impeded by any user application activity to the specified pool name, and any application throughput to the pool might be affected by the sysplex optimized workloads.

   Note: You must use a different pool name from the pool name that you use for the RS server if you already have a CFDT with the same name as your CICSplex.
2. Define and start a region status server job to run in an MVS batch region. For more information, see Defining and starting a region status server region.

What to do next

After you have successfully started your region status server, you can issue commands to manage the region status server and delete it if required. For more information, see Controlling region status servers and Deleting region status server pools.

Defining a list structure for a region status server

The region status server pool is defined in the list structure for a coupling facility data table. You define the list structure in a coupling facility resource manager (CFRM) policy.

About this task

You must allocate storage in the coupling facility to store CICS status.

CICS records the status of a CICS region in a coupling facility data table named after the CICSplex to which the region belongs. That table must belong to a CFDT pool that is named in the CICSplex definition for that CICSplex. The default name is DFHRSTAT. In each z/OS image, there must be a region status server for each region status pool that will serve the CICS regions belonging to that CICSplex. A CICSplex data table contains one region status record for each region in that CICSplex.

Define the structure in the current coupling facility resource management (CFRM) policy by using the IXCMIAPU utility. For an example of this utility, see member IXCCFRMP in the SYS1.SAMPLIB library. An example of a policy statement for a region status server pool is shown in Figure 43 on page 198.

You must authorize server access to the list structure. For details, see Authorizing server access to a list structure.

Procedure

1. Specify the name of the list structure. The name is formed by adding the prefix DFHCFLS_ to your chosen pool name, giving DFHCFLS_*poolname*. The default pool name as implemented by CPSM is DFHRSTAT.

You define and modify CICSplexes using the EYUSTARTCPLEXDEF view set. Using the CPLEXDEF detail view, you can modify the coupling facility (CF) tuning parameters for the region status (RS) server, which provide sysplex optimized workload routing.

Note: You can also modify the default region status (RS) pool name that will be used by all regions in the CICSplex. When you do not to use the default name DFHRSTAT, you must change the name before starting any other regions in the CICSplex. CPSM will not prevent you from changing the pool name while the CICSplex is active. If you make a change while the CICSplex is active, all CMAS and MAS regions in the CICSplex must be restarted as soon as possible. Failure to do so can result in inconsistent data in the CPSM WLM views and WLM optimization is deactivated until all the regions in the CICSplex are restarted.

2. Specify the size of the list structure. Although the record size and calculations shown in Figure 42 can be useful for your own information, you must use the IBM CFSizer tool to calculate the **INITSIZE** and **SIZE** parameters. The CFSizer tool takes minimums into consideration, but the calculations in Figure 42 do not. Failure to get valid sizing parameters using the CFSizer tool could result in a CICS runtime failure and DFHCF0403, DFHCF0409, and DFHCF0481 messages.

   When using the CFSizer tool, select **CICS Data Tables list structure** and specify the following values:

   Maximum number of tables
   Specify the number of CICSplexes that you have defined in CPSM and that will be using Sysplex Optimized Workloads. This is usually a very low number.

   Average rounded record size
   40

   Total records
   Specify the total number of CICS regions that will connect to all CICSplexes.

   Target usage percent
   Use the default.

   Maximum expansion percent
   Use the default.

   You can also specify ALLOWAUTOALT(YES), which enables automatic changes in the ratio of elements to entries, to make better use of the space within the structure.

A region status record is approximately 40 bytes long.

If PLEX1 contains 100 regions, PLEX2 contains 300 regions, and PLEX3 contains 1000 regions, the required structures are as follows:

- Poolname = DFHRSTAT, Table name = PLEX1, 100 regions x 40 bytes = 4 000 bytes total
- Poolname = DFHRSTAT, Table name = PLEX2, 300 regions x 40 bytes = 12 000 bytes total
- Poolname = DFHRSTAT, Table name = PLEX3, 1000 regions x 40 bytes = 40 000 bytes total

Figure 42. Example: Calculations of required structures

3. Specify the preference list of coupling facilities in which the policy can be stored.

4. When you have updated the CFRM new policy with the new structure definition, activate the policy using the following MVS command:

   ```
   SETXCF START,POLICY,POLNAME=policyname,TYPE=CFRM
   ```

   Where *policyname* is the CFRM policy being started, for example, DFHCFLS_DFHRSTAT. Note that defining the CFRM policy statements for a list structure does not create the list structure. The structure is created the first time an attempt is made to connect to it, which occurs when the first coupling facility data table (CFDT) server that refers to the corresponding pool is started.

Example

```
STRUCTURE  NAME(DFHCFLS_DFHRSTAT)
    SIZE(7168)
    INITSIZE(6144)
    PREFLIST(FACIL01,FACIL02)
```

Figure 43. Example definition of a list structure for region status servers

Defining and starting a region status server region

When you start a region status server, you activate a pool in an MVS image for that server.

Before you begin

Before you start a region status server region, you must define the region status server structure to be used for the pool. For information about defining a region status server list structure, see "Defining a list structure for a region status server" on page 196.

About this task

You can start the server as a started task, started job, or as a batch job. This task explains how to start a region status server job, to run in an MVS batch region. The job or task must start the region status server program, DFHCFMN, from the CICS authorized library, CICSTS54.CICS.SDFHAUTH.

Procedure

1. Specify the DFHCFMN program either in a SYSIN data set defined in the JCL, or in the **PARM** parameter on the EXEC statement.
2. Specify the mandatory and optional startup parameters for the DFHCFMN program. If you specify a startup parameter in both the SYSIN data set and the **PARM** parameter, the **PARM** value overrides the SYSIN value because the **MVS START** command can override the **PARM** value.
   a. You must specify a SYSPRINT DD statement for the print file.
   b. You must specify a SYSIN DD statement for the server parameters.

      Tip: To ensure that all pool-related parameters are consistent across MVS images, you must use the same SYSIN parameter data set, or an identical copy of it, for all servers accessing the same pool, and to specify in the PARM field any parameters that vary between servers.
   c. You must specify the region status pool name.
   d. You must concatenate the license activation data set (the SDFHLIC library) to the STEPLIB DD statement.

e. You can specify the **REGION** parameter. This parameter ensures that the coupling facility data table server region has enough storage to process the maximum number of data table requests that can run concurrently.

f. You can specify **TIME=NOLIMIT**. The server task remains in a wait, during most normal processing, because server processing is performed under the TCB of the client CICS region. If you omit this parameter, your server job might fail with abend S522 (wait limit exceeded), depending on the JWT value specified in the SMFPRM*xx* member of SYS1.PARMLIB.

g. Specify additional parameters as required. For example, you might want to control the maximum number of queues that are to be supported in the pool and the number of buffers that the server is to allocate. You might also need to add the required security access. See Authorizing a CICS region to a coupling facility data table.

Results

The region status server is running, ready to receive and broadcast region status data to the CICS regions connected to it. The CICS regions connect through the poolname that is specified in the CICSplex definition.

Region status server JCL example

```
//PRODRSS1 JOB  ...
//RSSERVER EXEC PGM=DFHCFMN,REGION=40M,TIME=NOLIMIT    CICS CFDT Server for RS
//STEPLIB  DD   DSN=CICSTS54.CICS.SDFHAUTH,DISP=SHR    Authorized library
//         DD   DSN=CICSTS54.CICS.SDFHLIC,DISP=SHR     License activation data set
//SYSPRINT DD   SYSOUT=*                               Messages and statistics
//SYSIN    DD   *
POOLNAME=DFHRSTAT                                     Pool name
MAXTABLES=100                                         Allow up to 100 tables
/*
```

Figure 44. Sample JCL to start a region status server address space

For an example of security parameters, see Authorizing a CICS region to a coupling facility data table.

Controlling region status servers

You can issue commands to control a region status server, using the MVS **MODIFY (F)** command to specify the job or started task name of the server region, followed by the server command.

About this task

The general form of an MVS modify command, using the short form F, is as follows:

```
F job_name,command parameters... comments
```

You use the MODIFY command to pass information to a job or started task. In this task, you use the following commands to control the region status servers.

Procedure

- To modify the server initialization parameters, use the MVS **SET** command:

  ```
  SET keyword=operand[,keyword=operand,...]
  ```

The **SET** command can be abbreviated to **T**, as for the MVS **SET** command. See "The SET command options" on page 187 for details.

- To display the values of one or more parameter values or statistics summary information on the console, use the **DISPLAY** command:

  ```
  DISPLAY keyword[=operand][,keyword[=operand,]...]
  ```

  The valid keywords for **DISPLAY** are all the initialization parameters, plus an additional set described under "DISPLAY and PRINT command options" on page 188.

  The **DISPLAY** command can be abbreviated to **D**, as for the MVS **DISPLAY** command.
- To print the output that the **DISPLAY** command produces, use the MVS **PRINT** command:

  ```
  PRINT keyword[=operand][,keyword[=operand,]...]
  ```

  The **PRINT** command produces the same output as DISPLAY, supporting the same keywords, but on the print file only.
- To delete a table, use the **DELETE TABLE**=*name* command. The table must not be in use for this command to succeed. You can abbreviate the command to **DEL**.
- To stop the server normally, use the **STOP** command. The server waits for any active connections to end first, and prevents any new connections while it is waiting. You can abbreviate the command to **P**. You can also use the **MVS STOP** command, which is equivalent to issuing the server STOP command through the **MVS MODIFY** command. The syntax of the **STOP** command is:

  ```
  STOP|P [jobname.]identifier[,A=asid]
  ```
- To terminate the server immediately, use the **CANCEL** command. You can also specify whether the server automatically restarts with the RESTART option. For information about CANCEL RESTART see "The CANCEL command options" on page 166.
- The server also responds to Cross System Extended Services (XES) events such as an operator **SETXCF** command to alter the structure size. If the server can no longer access the coupling facility, it automatically issues a server **CANCEL** command to close itself down immediately.

The SET command options

You can use the SET command to modify groups of server initialization parameters.

These system initialization parameter groups are:

- The statistics parameters
- The debug trace parameters
- The lock wait parameters
- The warning parameters
- The automatic ALTER parameters.

The following **SET** keywords are used to modify the server's recovery status of an inactive CICS region that had unresolved units of work when it last terminated:

RESTARTED=*applid*
: Establish a temporary recoverable connection for the given APPLID. This resolves any units of work that were in commit or backout processing when the region last terminated, and indicates whether there are any remaining indoubt units of work.

This keyword can be abbreviated to **RESTART** or **REST**.

COMMITTED={*applid*|*applid.uowid***}**
Establish a temporary recoverable connection for the specified APPLID and commit all indoubt units of work, or, if *uowid* is also specified, commit that specific unit of work.

This command should be used **only** when it is not possible to restart the original CICS region to resolve the work normally, because it can result in inconsistency between coupling facility data table resources and other CICS resources updated by the same unit of work.

This keyword can be abbreviated to **COMMIT** or **COMM**.

BACKEDOUT={*applid*|*applid.uowid***}**
Establish a temporary recoverable connection for the specified APPLID and back out all indoubt units of work, or, if *uowid* is also specified, back out that specific unit of work.

This command should be used *only* when it is not possible to restart the original CICS region to resolve the work normally, because it can result in inconsistency between coupling facility data table resources and other CICS resources updated by the same unit of work.

This keyword can be abbreviated to **BACKOUT** or **BACK**.

Use the following **SET** parameters to modify options relating to a specific table:

TABLE=*name*
specifies the table to which the following table-related parameters in the same command are to be applied. This parameter is required before any table-related parameters.

MAXRECS=*number*
Modify the maximum number of records that can be stored in the table specified by the preceding **TABLE** parameter.

If the maximum number is set to a value less than the current number of records in the table, no new records can be stored until records have been deleted to reduce the current number to within the new maximum limit. For a recoverable table, this also means that records cannot be updated, because the recoverable update process adds a new record on the rewrite operation then deletes the original record when the transaction completes.

This keyword can also be specified as **MAXNUMRECS**.

AVAILABLE={YES|NO}
Specify whether the table named by the preceding **TABLE** parameter is available for new OPEN requests. If the table is made unavailable, a CICS region that subsequently issues an OPEN request for the table receives a response indicating that it is unavailable, but regions that currently have the table open are not affected. Even when a table is marked as unavailable, a server can implicitly open it on behalf of a CICS region to allow recoverable work to be resolved during restart processing.

This keyword can be abbreviated to **AVAIL**.

Examples of the SET command: The following example changes the statistics options:

```
SET STATSOPT=BOTH,EOD=21:00,STATSINT=06:00
```

The following example modifies the maximum number of records allowed in the specified table:

```
SET TABLE=PAYECFT1,MAXRECS=200000
```

DISPLAY and PRINT command options

You can use the DISPLAY (and PRINT) commands to display the values of any initialization parameters plus some additional information.

Some of the parameters that provide additional information support generic names. You specify generic names using the following wildcard characters:

- An * (asterisk symbol). Use this anywhere in the parameter value to represent from 0 to 8 characters of any value. For example, CICSH* to represent all the CICS APPLIDs in a CICSplex identified by the letter H.
- A % (per cent symbol). Use this anywhere in the parameter value to represent only one character of any value. For example, CICS%T* to represent all the TOR APPLIDs in all CICSplexes.

The parameters supported by the DISPLAY and PRINT commands are as follows:

APPLIDS
Display the APPLID and MVS system name for every CICS region that currently has a recoverable connection to the pool. This command returns information not only for the server to which the MODIFY command is issued, but for all other servers connected to the same pool.

This keyword can be abbreviated to **APPLID**, **APPLS** or **APPL**.

APPLID={*applid*|*generic***}**
Display the APPLID and MVS system name for each region that currently has a recoverable connection to the server's pool, and whose APPLID matches *applid* or *generic*. This command returns information not only for the server to which the MODIFY command is issued, but for all other servers connected to the same pool.

applid Use this for a specific APPLID, which should match only one region in the sysplex.

generic Use a suitable generic value when you want to obtain information about several regions.

If *applid* or *generic* is not specified, the server treats this as equivalent to the command DISPLAY APPLIDS.

This keyword can also be specified as **APPLIDS**, **APPLS** or **APPL**.

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

CONNECTIONS
Display the jobnames and applids of the regions currently connected to the server to which the command is issued.

This keyword can be abbreviated to **CONN**.

TABLES
Display the names of all tables currently allocated in the pool.

TABLE={*name*|*generic_name***}**
Display information about the attributes and status of a specific table, or of a set of tables whose names match the generic name.

If no table name is specified, this is treated as equivalent to DISPLAY TABLES.

TABLEUSERS
Display the CICS APPLIDs of the regions that are currently using each of the tables currently defined in the pool.

This keyword can be abbreviated to **TABLEU**.

TABLEUSERS={*name*|*generic_name***}**
Display the CICS APPLIDs of the regions that are currently using the specified table, or using each of the set of tables whose names match the generic name.

If no table name is specified, this is treated as equivalent to DISPLAY TABLEUSERS.

This keyword can be abbreviated to **TABLEU**

UOWIDS
Display the applids of all regions that currently have unresolved recoverable units of work, together with the number of units of work that are currently in doubt, or are in the process of being committed or backed out. The information displayed does not include units of work that have not yet started the resolution process; that is, units of work that are still in flight.

This keyword can be abbreviated to **UOWS**.

UOWIDS={*applid*|*generic_applid***}**∨**{***applid***.*|*generic_applid***.*}**
Display, for the specified regions if they currently have unresolved recoverable units of work, information about those units of work. The information displayed does not include units of work that have not yet started the resolution process; that is, units of work that are still in flight. The information returned depends on the form of operand used.

applid|*generic_applid*
This form of operand displays the number of units of work that are currently in doubt, or are in the process of being committed or backed out.

If you specify *applid*, the server displays UOW information for a specific APPLID, which should correspond to only one region in the sysplex.

If you specify *generic_applid* the server displays UOW information for all the APPLIDs that match the generic APPLID specified.

*applid***.***|*generic_applid***.***
This form of operand displays:

- The state and local UOWID of each individual unit of work, followed by
- A summary of the number of units of work that are currently in doubt, or are in the process of being committed or backed out.

If you specify *applid*.*, the server displays the UOW information for a specific APPLID, which should correspond to only one region in the sysplex.

If you specify *generic_applid*.*, the server displays UOW information for all the APPLIDs that match the generic APPLID specified.

This keyword can be abbreviated to **UOWS**.

UOWID=*applid.uowid*
Display the state of an individual unresolved unit of work, identified by its applid and local unit of work ID (UOWID). Enter the local UOWID as 16 hexadecimal digits.

This keyword can be abbreviated to **UOW**.

DISPLAY and PRINT options for statistics summaries:

Use the following parameters to display or print statistics:

CFSTATS
Display statistics for coupling facility interface accesses and responses from the server.

This keyword can also be specified as **CFST** or **STATSCF**.

POOLSTATS
Display usage statistics for the pool list structure as a whole. This is based on information returned by coupling facility access requests, therefore it is only as current as the most recent request made through the server to which the command is issued.

This keyword can be abbreviated to **POOLST**.

TABLESTATS
Display statistics for requests, processed by the server to which the command is issued, for each table plus a summary of all requests processed, including those that are not table-specific, such as unit of work control.

Note that only tables with a non-zero number of requests since the start of the current statistics interval are shown.

This keyword can also be specified as **TABLEST**.

TABLESTATS={*name*|*generic_name***}**
Display request statistics for the specified table or tables.

name A specific table name in the pool accessed by the server. Returns statistics for this table only.

generic_name
A generic name that you can use to obtain statistics about a number of tables. Returns statistics for any table name that matches the generic name.

This keyword can be abbreviated to **TABLEST**.

STORAGESTATS
Display main storage allocation statistics for the server address space.

This keyword can be abbreviated to **STORAGEST** or **STGST**.

DISPLAY and PRINT options for combined lists of information:

These keywords represent combined lists of information:

PARAMETERS
Display the main parameter values. These are POOLNAME, SECURITY, SECURITYPREFIX, statistics options, and list structure options.

This keyword can be abbreviated to **PARM** or **PARMS**.

ALLPARAMETERS
Display all parameter values.

This keyword can be abbreviated to **ALLPARMS**.

STATISTICS
Display all available statistics.

This keyword can be abbreviated to **STAT** or **STATS**.

INITIALIZED
Display the parameters and statistics that are usually displayed when initialization is complete. This is equivalent to PARM, POOLSTATS, STGSTATS.

This keyword can be abbreviated to **INIT**.

ARM
Display all ARM-related parameter values:

- ARMELEMENTNAME
- ARMELEMENTTYPE
- ARMREGISTERED

This keyword can be coded as **ARMSTATUS**.

The CANCEL command options

You can use the CANCEL command to request an automatic restart.

Specify the following parameter:

RESTART={NO∨YES}
Terminate the server immediately, specifying whether or not automatic restart should be requested. The default is RESTART=NO.

If the server encounters an unrecoverable problem with the coupling facility connection, consisting either of lost connectivity or a structure failure, it cancels itself using the CANCEL RESTART=YES command. This terminates the existing connection and shuts down the server. A new instance of the server job is then started.

A server can also be restarted explicitly using either the server command CANCEL RESTART=YES or the MVS command CANCEL jobname,ARMRESTART.

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

Deleting region status server pools

You can delete a region status server pool by deleting its coupling facility list structure. You might do this for a service upgrade, or when a clean sysplex restart is required.

Before you begin

You can delete a structure only when no servers are connected to the pool; otherwise, MVS rejects the command.

About this task

For example:

```
SETXCF FORCE,STRUCTURE,STRNAME=DFHCFLS_poolname
```

You can verify that the pool has been successfully deleted by issuing the XCF command shown here:

```
D XCF STRUCTURE,STRNAME=DFHCFLS_poolname
```

Note that if you delete a region status server structure while CICS regions and workload are running, you disable CICSPlex SM WLM optimized functions.

What to do next

When you attempt to start a server for a pool that has been deleted (or attempt to reload the pool), it is allocated as a new structure. The newly allocated structure uses size and location attributes specified by the currently active CFRM policy and other values determined by the server initialization parameters (in particular, **MAXTABLES**).

Setting up and running a named counter server

CICS provides an efficient way to generate unique sequence numbers for use by applications in a Parallel Sysplex environment; for example, to allocate a unique number for orders or invoices. A named counter server maintains each sequence of numbers as a named counter.

About this task

The following procedure outlines the steps to set up and manage a named counter server. Details for each step are in the topics that follow.

Procedure

1. Define a named counter options table.
2. Define a list structure.
3. Define and start a named counter server job, to run in an mvs batch region.
4. Manage named counter server regions.
   - Issue commands to control a named counter server.
   - Change the size of named counter pools.
   - Delete or empty named counter pools.
   - Unload and reload named counter pools.
   - Dump named counter pool list structures.

Named counter server overview

Each time a number is assigned, the corresponding named counter is automatically incremented so that the next request gets the next number in sequence. Named counters are the Sysplex equivalent of COUNTER in the Common System Area (CSA) of a single region CICS system.

A named counter server provides a full set of functions to define and use named counters. Each named counter consists of:

- A 16-byte name
- A current value
- A minimum value
- A maximum value.

The values are internally stored as 8-byte (double word) binary numbers, but the user interface allows them to be treated as any length from 1 to 8 bytes, typically 4 bytes.

Named counters are stored in a pool of named counters, where each pool is a small coupling facility list structure, with keys but no data. The pool name forms

part of the list structure name. Each named counter is stored as a list structure entry keyed on the specified name, and each request for the next value requires only a single coupling facility access.

Warning: The counters are lost if the coupling facility fails.

Named counter structures and servers

Within each MVS image, there must be one named counter server for each named counter pool accessed by CICS regions and batch jobs in the MVS image.

Named counter pools are defined as a list structure in the coupling facility resource management (CFRM) policy. The pool name, which is used to form the server name with the prefix DFHNC, is specified in the startup JCL for the server.

Figure 45 illustrates a parallel sysplex with three CICS AORs linked to anamed counter server.

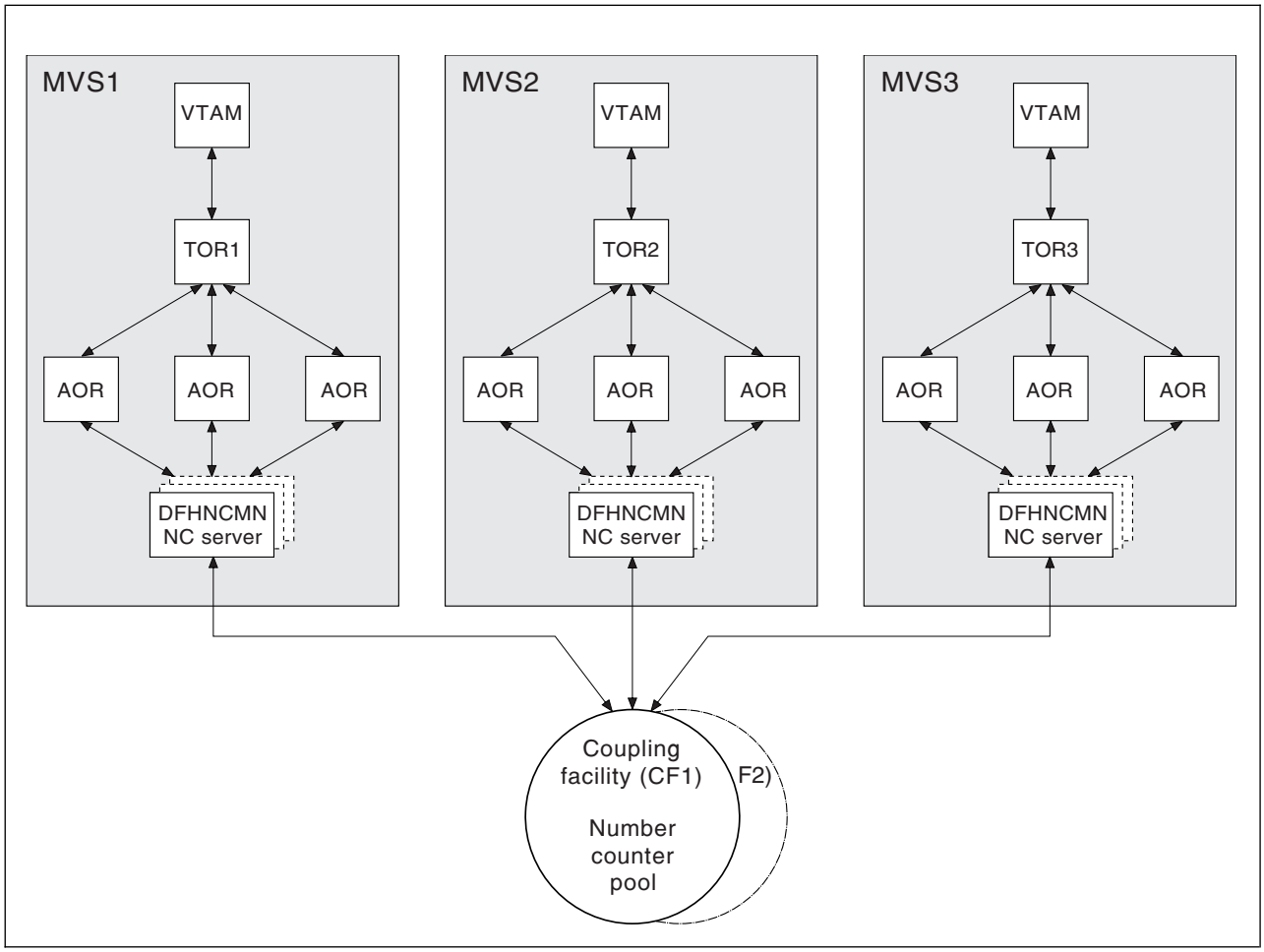


Figure 45. Conceptual view of a parallel sysplex with named counters

Application program access to named counters

CICS provides a command level API for the named counter facility. To reference a named counter, an application program can specify either the actual name of the pool in which the named counter is stored, or it can specify a dummy pool

selection parameter. The dummy pool value is mapped to the actual pool name by the **POOL** parameter that is specified in the options table, DFHNCOPT.

Specifying a dummy pool makes it easy to use a different pool (for example, to isolate test pools from production pools) without having to change the pool selection parameter in the application program.

To vary the pool used by a CICS region, either load a different copy of the options table from STEPLIB, or use a common options table where the pool name selection is conditional on the job name and CICS APPLID, in addition to the pool name selection parameter. The options table also supports invocation of a user-specified program to select the appropriate pool given the pool selection parameter.

Security

The server must be authorized to access the coupling facility list structure in which the named counter pool is defined. The server must also be authorized to act as a named counter server.

For information on how to define the necessary authorizations see Authorizing access to named counter pools and servers.

Note: You cannot control access to individual named counters.

Defining a named counter options table

The named counter callable interface determines the actual pool name in response to a DFHNCTR call by referring to the DFHNCOPT options table.

About this task

When a pool selector value is encountered for the first time, the pool name is determined via the options table. The name is then saved and used for all subsequent requests for the same pool selector from the same TCB. This continues for the life of the TCB or until the NC_FINISH function is used specifying that pool selector value. CICS supplies a default DFHNCOPT in source form, which you can customize and generate using the DFHNCO macro. A typical use of the options table is to enable production and test regions to use a different counter pool without needing to change the pool name in application programs.

To avoid the need to maintain multiple versions of the options table, you can use table entries to select pools based not only on the pool selection parameter specified on the DFHNCTR call, but also on the job name and APPLID of the CICS region. You can also specify the name of a user exit program to be called to make the pool selection.

Define an options table using one or more invocations of the DFHNCO macro. Each invocation generates an options table entry that defines the pool name or user exit program to be used whenever any selection conditions specified on the entry satisfy an application program request. The first entry automatically generates the table header, including the CSECT statement. Follow the last entry with an END statement specifying the table module entry point, DFHNCOPT.

The options table parameters

The DFHNCOPT options table parameters are illustrated in the following figure.

```
DFHNCO [POOLSEL={(generic_values)|*},]
       [JOBNAME={(generic_values)|*},]
       [APPLID={(generic_values)|*},]
       {POOL={YES|NO|name} | CALL=programname}

        Terminate the last DFHNCO entry with the
        following END statement:

END    DFHNCOPT
```

Figure 46. DFHNCOPT options table

The POOLSEL, JOBNAME, and APPLID parameters specify optional selection conditions to determine whether the entry applies to the current request. You can specify each of these operands as

- A single generic name
- A list of names in parentheses, the list containing two or more generic names, each name separated by a comma.

Each name comprises the characters that can appear on the appropriate parameter, plus the wild-card characters * to match any sequence of zero or more non-blank characters, and % to match any single non-blank character. When multiple generic name are specified, the selection condition is satisfied if any one of them matches. A blank pool selector value can be matched using a null POOLSEL operand, for example POOLSEL= or POOLSEL=().

POOLSEL={(generic1,generic2,...,...)∨* }
Specifies that this options table entry applies only when the pool selection parameter specified by the application program matches one of the generic names specified on this parameter.

Specifying POOLSEL=, or POOLSEL=() is the equivalent of specifying 8 blanks.

If you omit the POOLSEL keyword, it defaults to *.

JOBNAME={(generic1,generic2,...,...)∨* }
Specifies that this options table entry applies only when the caller's job name matches one of the generic names specified on this parameter.

If you omit the JOBNAME keyword, it defaults to *.

APPLID={(generic1,generic2,...,...)∨* }
Specifies that this options table entry applies only when the caller's CICS APPLID matches one of the generic names specified on this parameter.

If you omit the APPLID keyword, it defaults to *.

POOL={YES∨NO∨*name*}
Specifies the pool name to be used. This parameter is mutually exclusive with the CALL parameter. The options are:

YES specifies that the server is to use the pool selection parameter specified by the application program as the actual pool name. An all-blank pool selection parameter means the server is to use the default pool name. For the call interface, the default name is DFHNC001. For the EXEC CICS API, the default name is specified by the NCPLDFT system initialization parameter.

NO specifies that the server is not to use any pool and is to reject the request with an error.

name specifies the actual pool name that the server is to use. If *name* is omitted, this indicates that the default pool is to be used. (For the CALL interface, the default pool is always DFHNC001, but for the EXEC CICS interface you can specify the default pool using the NCPLDFT system initialization parameter.)

CALL=*programname*

specifies the name of a user exit program to be called to determine the actual pool name to be used. This parameter is mutually exclusive with the POOL parameter.

The program named can be link-edited with the options table, which generates a weak external reference (WXTRN), or it can be loaded dynamically the first time it is used. The program is called using standard MVS linkage in AMODE 31, with a standard save area and parameter list pointing to four fields, in the following order:

- The 8-byte actual pool name result field
- The 8-byte pool selection parameter.
- The 8-byte job name
- The 8-byte APPLID if running under CICS, otherwise blanks

The end-of-list bit is set in the last parameter address.

The program should be reentrant and should be linked with RMODE ANY, so that it (and the option table if linked with the program) can be loaded above the line. Temporary working storage can be acquired and released using MVS GETMAIN and FREEMAIN. As this program is only called when a new pool selection value is used, the use of GETMAIN and FREEMAIN should not affect performance.

The exit program cannot use any CICS services. If it is used in a CICS region, it must avoid using any MVS services which could result in a long wait, as it will normally be executed under the CICS quasi-reentrant (QR) TCB.

The user exit program indicates its result by setting one of the following return codes in register 15:

0 Use the pool name that is successfully set, in the first field of the parameter list, by the user exit program.

4 The program cannot determine the pool name on this invocation. Continue options table processing at the next entry, as for the case where selection conditions were not met.

8 Reject the request (as if POOL=NO was specified).

The default options table, supplied in CICSTS54.CICS.SDFHLINK, contains the following entries:

```
DFHNCO   POOLSEL=DFHNC*,POOL=YES
DFHNCO   POOL=
END      DFHNCOPT
```

With the default options table in use, any pool selector parameter that specifies a string beginning with DFHNC is taken to be an actual pool name, indicated by POOL=YES in the table entry. Any other value, including a value of all spaces, is assigned the default pool name, indicated by the POOL= table entry without a POOLSEL parameter.

The source for this default table is supplied in CICSTS54.CICS.SDFHSAMP.

Making an options table available to CICS

To ensure that your CICS region can load the named counter options table, install the link-edited table into an APF-authorized library in STEPLIB. Alternatively you can install the table in a suitable library in the LINK list.

Defining a list structure for a named counter server

Define one or more coupling facility list structures for the named counter facility, where each list structure represents a pool of named counters. Each named counter pool is accessed through a cross-memory server region.

Before you begin

A coupling facility structure contains both stored data and the information needed to manage and access that data, in a similar way to a key-sequenced data set. The amount of internal control information depends on the level of functionality and performance of the coupling facility control code for the current coupling facility level (CFLEVEL), and might increase for a higher CFLEVEL. Ensure that you consider storage requirements for your list structures. For more information, see "Named counter list structure storage" on page 212.

About this task

Define the structure in the current coupling facility resource management (CFRM) policy by using the utility IXCMIAPU. For an example of this utility, see member IXCCFRMP in the SYS1.SAMPLIB library (see Administrative data utility for CFRM policy data in z/OS MVS Setting Up a Sysplex).

Procedure

1. Specify the name of the list structure. The name is formed by adding the prefix DFHNCLS_ to your chosen pool name, giving DFHNCLS_*poolname*.
2. Specify the size of the list structure. You can allocate an initial and maximum size using the INITSIZE and SIZE parameters in the CRFM policy definition. For an accurate estimate of storage requirements, use the IBM CFSizer tool. See CFSizer.
3. Specify the preference list of coupling facilities in which the policy can be stored. An example definition of a coupling facility list structure for named counters is as follows:

```
STRUCTURE  NAME(DFHNCLS_PRODNC1)
   SIZE(2048)
   INITSIZE(1024)
   PREFLIST(FACIL01,FACIL02)
```

4. When you have updated the CFRM new policy with the new structure definition, activate the policy using the following MVS command:

```
SETXCF START,POLICY,POLNAME=policyname,TYPE=CFRM.
```

Results

You have defined the CFRM policy statements for a list structure. This action does not create a list structure. The structure is created the first time an attempt is made to connect to it, which occurs when the first named counter server that refers to the corresponding pool is started.

Before you start the named counter server, ensure that you have defined and started the authorized cross-memory (AXM) server environment (see “Defining and starting AXM system services” on page 151).

Named counter list structure storage

You can use thez Systems Coupling Facility Structure Sizer (CFSizer) tool to calculate storage requirements for named counter list structures in a coupling facility. If you increase or decrease the structure size, consider whether you need to update other parameters.

A coupling facility structure contains both stored data and the information needed to manage and access that data, in a similar way to a key-sequenced data set. The amount of internal control information depends on the level of functionality and performance of the coupling facility control code for the current coupling facility level (CFLEVEL), and might increase for a higher CFLEVEL. For more information, see “Coupling facility storage management” on page 224.

CFSizer is a web-based application that takes these factors into account and communicates with a coupling facility at a current CFLEVEL to calculate storage requirements. See CFSizer.

If you enter the number of counters you require, the CFSizer tool calculates the size of a structure that can contain at least that number of counters. However, for practical operation, some free space must be available so that the structure does not become full, and to avoid warning messages about low space. It is advisable to use no more than approximately 75% of the structure size. Estimate the maximum number of counters that you require, then increase that number by a third to include the free space in the calculation.

The space required for a named counter pool depends on the number of different named counters that you need, but the minimum size can be enough for most needs. For example, for CFLEVEL 16, a 1 MB structure can hold up to 1000 named counters.

All structure sizes are rounded up to the next storage increment for the coupling facility level (CFLEVEL) at allocation time. For example, sizes are rounded up to the nearest 1 MB for CFLEVEL 16.

Provided that space is available in the coupling facility, you can use the MVS **SETXCF** command to increase the structure size dynamically from its initial size towards its maximum size, making the new space available immediately to any currently active servers. If too much space is allocated, you can reduce the structure size to free up coupling facility storage for other purposes. However, this could take some time if the coupling facility has to move existing data out of the storage that is being freed. If you alter the size in this way, update the **INITSIZE** parameter in the coupling facility resource management (CFRM) policy to reflect the new size, so that the structure does not revert to its original size if it is subsequently re-created or reloaded.

Defining and starting a named counter server region

You activate a named counter pool in an MVS image by starting up a named counter server region for that pool.

About this task

You can start the server as a started task, started job, or as a batch job. The job or task must invoke the named counter server region program, DFHNCMN, and must run from an APF-authorized library. DFHNCMN is in the CICS authorized library, CICSTS54.CICS.SDFHAUTH.

Procedure

1. Specify the DFHNCMN program either in a SYSIN data set defined in the JCL, or in the **PARM** parameter on the EXEC statement.
2. Specify the mandatory and optional startup parameters for the DFHNCMN program. If you specify a startup parameter in both the SYSIN data set and the **PARM** parameter, the **PARM** value overrides the SYSIN value because the **MVS START** command can override the **PARM** value.
   a. You must specify a SYSPRINT DD statement for the print file.
   b. You must specify a SYSIN DD statement for the server parameters.
   c. You must specify the TS pool name.
   d. You must concatenate the license activation data set (the SDFHLIC library) to the STEPLIB DD statement.
   e. It is recommended that you specify the **REGION** parameter. This parameter ensures that the coupling facility data table server region has enough storage to process the maximum number of data table requests that can run concurrently.
   f. It is recommended that you specify TIME=NOLIMIT. The server task remains in a wait during most normal processing, because server processing is performed under the TCB of the client CICS region. If you omit this parameter, your server job could fail with abend S522 (wait limit exceeded), depending on the JWT value specified in the SMFPRM*xx* member of SYS1.PARMLIB.
   g. Specify additional parameters as required. For example, you might want to control the maximum number of queues that are to be supported in the pool and the number of buffers the server is to allocate.

   Tip: A good way to ensure that all pool-related parameters are consistent across MVS images is to use the same SYSIN parameter data set, or an identical copy of it, for all servers accessing the same pool, and to specify any parameters that vary between servers in the PARM field.

Example

```
//MVSnNC1 JOB  ...
//NCSERVER EXEC PGM=DFHNCMN,REGION=32M,TIME=NOLIMIT   named counter server
//STEPLIB  DD   DSN=CICSTS54.CICS.SDFHAUTH,DISP=SHR    Authorized library
//         DD   DSN=CICSTS54.CICS.SDFHLIC,DISP=SHR     License activation data set
//SYSPRINT DD   SYSOUT=*                              Messages and statistics
//SYSIN    DD   *
POOLNAME=MVSnNC1                              Pool name
/*
```

Figure 47. Sample JCL to start a named counter server address space

Named counter server parameters

Parameters are specified in the form KEYWORD=*value*, where keywords can optionally be specified in mixed case to improve readability.

If you specify more than one parameter in the PARM field or on the same SYSIN input line, the parameters must be separated by a comma. Any text following one or more spaces is taken as a descriptive comment. Any parameter line which starts with an asterisk or a space is assumed to be a whole line comment.

You can enter some parameter keywords in more than one form, such as in abbreviated or truncated form.

The main parameters are listed on the server print file during startup.

Named counter server REGION parameter:

Use the JCL REGION parameter to ensure that the named counter server region has enough storage to process the maximum number of named counter requests that can be executing concurrently.

The named counter server typically uses less than one megabyte of storage above 16 MB but below 2 GB, and less than 20 KB below 16 MB.

During server initialization, the server acquires all the available storage above 16 MB but below 2 GB, as determined by the REGION parameter, then releases 5% of it for use by operating system services. It also acquires 5% of the free storage below 16 MB for use in routines that require 24-bit addressable storage.

After initialization, the server uses AXM page allocation services to manage its storage. Server statistics indicate how much storage is allocated and used in the storage areas above and below 16 MB, which are called AXMPGANY and AXMPGLOW in the statistics.

If a task in the server region or a cross-memory request runs out of storage, this is likely to result in AXM terminating that task or request using a simulated abend with system completion code 80A to indicate a GETMAIN failure. Although the server can usually continue processing other requests in this case, running out of storage in a critical routine can cause the server to terminate. Therefore, it is best to ensure that the REGION size is large enough to eliminate this risk.

Pool name parameter:

This parameter, POOLNAME, is always required:

POOLNAME=*name*
: specifies the 8-character name of the named counter pool. This is appended by the server to the prefix DFHNC to create its own server name, as in DFHNC.*poolname*, and also to the prefix DFHNCLS_ to create the name of the coupling facility list structure, as in DFHNCLS_*poolname*.

  This parameter is valid only at server initialization, and must always be specified.

  This keyword can be abbreviated to **POOL**.

Statistics parameters:

Use the following parameters to specify server statistics options:

ENDOFDAY={00:00|*hh:mm***}**
: specifies the time of day, in hours and minutes, when the server is to collect and reset end-of-day statistics.

Note: If the STATSOPTIONS parameter specifies NONE, the server still writes end-of-day statistics to the print file.

The valid range of times is from 00:00 to 24:00.

This keyword can be abbreviated to **EOD**.

STATSINTERVAL={03:00|*hh:mm***}**
specifies the statistics collection interval, in the range 1 minute to 24 hours. This parameter is ignored if the STATSOPTIONS parameter specifies NONE.

The time interval can range from 00:01 to 24:00.

This keyword can be abbreviated to **STATSINT**.

STATSOPTIONS={NONE|SMF|PRINT|BOTH}
specifies whether the server is to produce interval statistics, and the destination for the statistics it produces.

NONE
The server does not produce any interval statistics.

SMF The server produces interval statistics and writes them to the current SMF data set only.

PRINT
The server produces interval statistics and writes them to the server's print file only.

BOTH The server produces interval statistics and writes them to the current SMF data set and to the server's print file.

This keyword can be abbreviated to **STATSOPT**.

Automatic restart manager (ARM) parameters:

During server initialization, the server unconditionally registers with ARM except when the server program is invoked with either the UNLOAD or the RELOAD functions. The server will not start if the registration fails. You can use the **ARMELEMENTNAME** and **ARMELEMENTTYPE** parameters to override default processing for the automatic restart manager.

ARMELEMENTNAME=*elementname*
specifies the automatic restart manager element name, up to 16 characters, to identify the server to ARM for automatic restart purposes. The permitted characters for the element name are A to Z 0-9 $ # @ and the underscore symbol (_).

The default identifier is of the form DFHNC*nn_poolname*, where NC represents the server type, *nn* is the &SYSCLONE value for the system (which can be either one or two characters), and *poolname* is the name of the pool served by the server.

This parameter is only valid at server initialization.

This keyword can be abbreviated to **ARMELEMENT** or **ARMELEMNAME**.

ARMELEMENTTYPE=*elementtype*
specifies the automatic restart manager element type, up to 8 characters for use in ARM policies as a means of classifying similar elements. The permitted characters for the element type are A to Z 0-9 $ # and @.

The default element type is SYSCICSS.

This parameter is only valid at server initialization.

This keyword can be abbreviated to **ARMELEMTYPE**.

List structure parameter:

The list structure parameter specifies the attribute that is used only for initial allocation of resources when the list structure is created for a named counter pool. Initial allocation occurs the first time a server is started for the named counter pool.

POOLSIZE={0|*number*{K|M|G}}

Specifies the initial amount of coupling facility storage to be allocated for the pool list structure, expressed as kilobytes (*n* K), megabytes (*n* M) or gigabytes (*n* G).

Usually, you can omit this parameter and specify the structure size by using the **INITSIZE** parameter in the coupling facility resource manager (CFRM) policy. However, this parameter can be useful if the structure is reallocated or reloaded but the CFRM policy has not been updated to reflect the required size.

0
The special value 0 means that the server obtains an initial allocation using the parameters specified in the CFRM policy. If the CFRM policy specifies an **INITSIZE** value for the structure, this determines the initial allocation. Otherwise, the CFRM **SIZE** value (the maximum size of the structure) is allocated.

number
A non-zero value specifies an initial amount of storage to be allocated, overriding the **INITSIZE** parameter in the CFRM policy. This value is rounded up by MVS to the next storage increment for the coupling facility level (CFLEVEL). For example, for CFLEVEL 16, the value is rounded up to the nearest 1 MB.

The value must be less than the CFRM **SIZE** parameter, otherwise the value of **POOLSIZE** is ignored and the initial allocation uses the parameters specified in the CFRM policy.

The valid range is from 0 to 16777215M. However, you must specify a value that is less than the maximum size of a structure in z/OS, otherwise a z/OS error occurs. For example, in z/OS, Version 1 Release 12, the maximum size of a structure is 1048576 MB (1 TB). For more details, see the information about CFRM parameters in *z/OS MVS Setting Up a Sysplex*.

This parameter is valid only at server initialization and is used only when the structure is first allocated.

Debug trace parameters:

These parameters are provided only for intensive debug tracing.

Using these options in a production environment could have a significant impact on performance and cause the print file to grow very rapidly, using up spool space.

Trace messages from cross-memory requests can be lost if they are generated faster than the trace print subtask can print them. In this event, the trace only indicates how many messages were lost.

CFTRACE={OFF|ON}
specifies the coupling facility interface debug trace option.

OFF Coupling facility interface debug trace is disabled.

ON Coupling facility interface debug trace produces trace messages on the print file, indicating the main parameters to the coupling facility request interface, and the result from the IXLLIST macro.

This keyword can also be specified as **TRACECF**.

RQTRACE={OFF|ON}
specifies the request debug trace option.

OFF Request debug trace is disabled.

ON Request debug trace produces trace messages on the print file, indicating the main parameters on entry to each cross-memory request, and the results on exit.

This keyword can also be specified as **TRACERQ=**.

Warning parameters:

Use these parameters to modify the thresholds at which warning messages are issued when the structure becomes nearly full.

ENTRYWARN={80|*number*}
specifies the percentage of list structure entries in use at which warning messages should be first triggered.

The valid range is from 1 to 100 per cent.

ENTRYWARNINC={5|*number*}
specifies the percentage increase (or decrease) of entries in use before the next warning message should be triggered (reduced to 1 when the next increase would otherwise reach 100). After the first warning, additional messages are issued as the number of elements increases and decreases. These messages stop when the number of entries in use has fallen at least this percentage below the initial warning level.

The valid range is from 1 to 100 per cent.

Controlling named counter server regions

You can issue commands to control a named counter server, using the **MVS MODIFY (F)** command to specify the job or started task name of the server region, followed by the server command.

About this task

The general form of a named counter server command, using the short form F, is as follows:

```
F server_job_name,command
parameters...  comments
```

SET *keyword=operand[,keyword=operand,...]*
Change one or more server parameter values.

Procedure

- To change one or more server parameter values, use the **SET** command. You can abbreviate the command to **T**, as for the **MVS SET** command. See "The SET command options" for details.
- To display one or more parameter values or statistics summary information on the console, use the **DISPLAY** command. The syntax is as follows:

  ```
  DISPLAY keyword[=operand][,keyword[=operand,]...]
  ```

  The valid keywords for **DISPLAY** are all the initialization parameters, plus an additional set described under "DISPLAY and PRINT command options." You can abbreviate the command to **D**, as for the **MVS DISPLAY** command.
- To print the parameter values, use the **PRINT** command. This command produces the same output as the **DISPLAY** command, supporting the same keywords, but on the print file only.
- To stop the server normally, use the **STOP** command. The server waits for any active connections to end first, and prevents any new connections while it is waiting. You can abbreviate the command to **P**. You can also use the **MVS STOP** command, which is equivalent to issuing the server STOP command through the **MVS MODIFY** command. The syntax of the **STOP** command is:

  ```
  STOP|P [jobname.]identifier[,A=asid]
  ```
- To terminate the server immediately, use the **CANCEL** command. You can also specify whether the server automatically restarts with the RESTART option. For information about CANCEL RESTART see "The CANCEL command options" on page 166.
- The server also responds to XES events such as an operator **SETXCF** command to alter the structure size. If the server can no longer access the coupling facility, it automatically issues a server **CANCEL** command to close itself down immediately.

The SET command options

You can use the **SET** command to modify groups of server initialization parameters.

The groups of server initialization parameters are as follows:

- The statistics parameters
- The debug trace parameters
- The warning parameters

See "Named counter server parameters" on page 213 for details of these keywords.

Examples of the SET command: The following example changes the statistics options:

```
SET STATSOPT=BOTH,EOD=21:00,STATSINT=06:00
```

DISPLAY and PRINT command options

You can use the **DISPLAY** and **PRINT** commands to display the values of any initialization parameters plus some additional information.

The parameters supported by the **DISPLAY** and **PRINT** commands are as follows:

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

CONNECTIONS
Display the job names and APPLIDs of the regions currently connected to the server to which the command is issued.

This keyword can be abbreviated to **CONN**.

COUNTERS
Display the names of all the named counters currently allocated in a pool.

COUNTERS={*name*|*generic_name***}**
Display the details of a specific named counter, or set of named counters whose names match the generic name. Generic names are specified using the wildcard characters * (asterisk symbol) and % (per cent symbol).

If no named counter is specified, this is treated as equivalent to DISPLAY COUNTERS.

This keyword can be abbreviated to **COUNTER**.

DISPLAY and PRINT options for statistics summaries

Use the following parameters to display or print statistics:

CFSTATS
Display statistics for coupling facility interface accesses and responses from the server.

This keyword can also be specified as **CFST** or **STATSCF**.

POOLSTATS
Display usage statistics for the pool list structure as a whole. This is based on information returned by coupling facility access requests, therefore it is only as current as the most recent request made through the server to which the command is issued.

This keyword can be abbreviated to **POOLST**.

STORAGESTATS
Display main storage allocation statistics for the server address space.

This keyword can be abbreviated to **STORAGEST** or **STGST**.

DISPLAY and PRINT options for combined lists of information

These keywords represent combined lists of information:

PARAMETERS
Display the main parameter values:

- POOLNAME
- STATSOPT
- ENDOFDAY
- STATSINTERVAL
- POOLSIZE

This keyword can be abbreviated to **PARM** or **PARMS**.

ALLPARAMETERS
Display all parameter values, which are those listed for PARAMETERS, as well as the following:

- CFTRACE
- RQTRACE
- ENTRYWARN
- ENTRYWARNINC

This keyword can be abbreviated to **ALLPARMS**.

STATISTICS
Display all available statistics. This keyword can be abbreviated to **STAT** or **STATS**.

INITIALIZED
Display the parameters and statistics that are usually displayed when initialization is complete, which are those listed for PARAMETERS, as well as the following:

- POOLSTATS
- STGSTATS

This keyword can be abbreviated to **INIT**.

ARM
Display all ARM-related parameter values:

- ARMELEMENTNAME
- ARMELEMENTTYPE
- ARMREGISTERED

This keyword can be coded as **ARMSTATUS**.

The CANCEL command options

You can use the CANCEL command to request an automatic restart.

Specify the following parameter:

RESTART={NO∨YES}
Terminate the server immediately, specifying whether or not automatic restart should be requested. The default is RESTART=NO.

If the server encounters an unrecoverable problem with the coupling facility connection, consisting either of lost connectivity or a structure failure, it cancels itself using the CANCEL RESTART=YES command. This terminates the existing connection and shuts down the server. A new instance of the server job is then started.

A server can also be restarted explicitly using either the server command CANCEL RESTART=YES or the MVS command CANCEL jobname,ARMRESTART.

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

Deleting or emptying named counter pools

You can delete a named counter pool using the MVS **SETXCF** command to delete its coupling facility list structure.

About this task

For example:

```
SETXCF FORCE,STRUCTURE,STRNAME=DFHNCLS_poolname
```

You can delete a structure only when there are no servers connected to the pool, otherwise MVS rejects the command.

When you attempt to start a server for a pool that has been deleted (or attempt to reload the pool), it is allocated as a new structure. The newly allocated structure uses size and location attributes specified by the currently active CFRM policy.

Changing the size of named counter pools

If the structure is becoming full, and the current pool size is less than the maximum, you can use the SETXCF START,ALTER command to increase the pool size.

About this task

For example:

```
SETXCF START,ALTER,STRNAME=DFHNCLS_poolname,SIZE=size
```

SIZE is expressed in kilobytes.

Unloading and reloading named counter pools

You can unload, and reload, the complete contents of a named counter pool to and from a sequential data set by invoking the server program with the **FUNCTION** parameter, using the UNLOAD and RELOAD options.

About this task

You can use this function, for example, to:

- Preserve the named counter pool during planned coupling facility maintenance, or
- Move the pool to a different coupling facility.

FUNCTION={UNLOAD|RELOAD}

Specify the function for which the server is being initialized.

UNLOAD

Unload the entire contents of the named counter pool specified on the POOLNAME parameter to a sequential data set. When the unload processing has completed (normally or abnormally) the server program terminates.

The UNLOAD function requires a DD statement for DDNAME DFHNCUL describing the sequential data set to which the table pool is to be unloaded. The format of the unloaded data set is:

```
RECFM=F
LRECL=4096
BLKSIZE=4096
```

RELOAD

Reload, into the named counter pool named on the POOLNAME parameter, a previously unloaded named counter pool.

The RELOAD function requires a DD statement for DDNAME DFHNCRL, describing the sequential data set from which the table pool is to be reloaded.

The structure is allocated, if necessary, during reloading, in which case you can use the same server parameters to control structure attributes as for normal server startup. The reload process bypasses named counters that are already found in the pool (for example, because the structure was too small and the reload job had to be restarted after using ALTER to increase the structure size).

Note: If the unloaded pool structure was altered dynamically at any time after initial allocation (by using the SETXCF command to increase

the size), ensure that the increased size is allocated for the reloaded pool. The recommended way is to update the INITSIZE parameter for the structure in the current CFRM policy whenever you alter the structure size, and to activate the updated policy using the SETXCF START,POLICY command. Alternatively, you can specify the required pool size in the POOLSIZE parameter in the reload JCL.

Note: If you omit the FUNCTION parameter, the server program initializes a named counter server address space.

For the UNLOAD and RELOAD function, the server program requires exclusive use of the list structure. If the structure is currently being used by a normal server, the unload or reload attempt is rejected. Similarly, if a normal server attempts to start while an unload or reload job is in progress, the attempt fails because shared access to the structure is not available.

You can specify all normal server parameters when unloading or reloading, but some of these (for example, statistics-related parameters) are ignored because they do not apply to unload or reload processing.

If reloading fails because it runs out of space, the resulting messages include the numbers of named counters reloaded and blocks read up to the time of the failure. You can compare these values with those in the messages from the original unload job, to determine how many more named counters remain to be loaded.

Unload JCL example

The JCL in the following example shows you how to unload a named counter pool.

```
//UNLDNCD1 JOB  ...
//NCUNLOAD EXEC PGM=DFHNCMN        CICS named counter server program
//STEPLIB  DD   DSN=CICSTS54.CICS.SDFHAUTH,DISP=SHR  Authorized library
//SYSPRINT DD   SYSOUT=*           Options, messages and statistics
//DFHNCUL  DD   DSN=NC1.UNLOADED.POOL,  Unloaded named counter pool
//         DISP=(NEW,CATLG),
//         SPACE=(4096,(10000,1000))  Estimated size in 4K blocks
//SYSIN    DD   *
FUNCTION=UNLOAD                  Function to be performed is UNLOAD
POOLNAME=PRODNC1                 Pool name
/*
```

Figure 48. Unload JCL example

Reload JCL example

The following JCL example shows you how to reload a named counter pool.

```
//RELDNCD1 JOB  ...
//NCRELOAD EXEC PGM=DFHNCMN        CICS named counter server program
//STEPLIB  DD   DSN=CICSTS54.CICS.SDFHAUTH,DISP=SHR  Authorized library
//SYSPRINT DD   SYSOUT=*           Options, messages and statistics
//DFHNCRL  DD   DSN=NC1.UNLOADED.POOL,DISP=OLD  Unloaded pool
//SYSIN    DD   *
FUNCTION=RELOAD                  Function to be performed is RELOAD
POOLNAME=PRODNC1                 Pool name
/*
```

Figure 49. Reload JCL example

Dumping named counter pool list structures

You can use the MVS **DUMP** command to obtain a dump of the coupling facility list structure for a named counter pool.

About this task

For information on dumping and formatting a list structure, see Using dumps in problem determination.

Coupling facility server operations

The operations that you can perform on all three CICS coupling facility servers, for temporary storage, coupling facility data tables and named counters, are similar and the following information describes all three unless otherwise indicated.

Monitoring coupling facility server messages

The server issues various messages during execution, some of which might indicate that serious problems are developing, for example that the coupling facility structure is becoming full.

It is important to understand the types of messages that the server issues and to ensure that system status messages are monitored for possible problems.

Server messages

Messages that are issued by the server code itself start with the five-letter server prefix (DFHXQ, DFHCF or DFHNC).

These messages fall into the following groups:

- Operator console system status messages, which are issued by WTO (Write To Operator) with routing codes 2 (operator information) and 11 (programmer information), and descriptor code 4 (system status).

  These messages provide information about important status changes, primarily about the beginning and end of server initialization and the beginning and end of server termination, and about problems.

  Any server message that is issued in this way during normal running indicates a potentially serious problem, and should not be ignored. If this process is automated, a simple rule is to ignore the specific list of status messages for normal initialization and termination, and to treat any other server message as a warning.

  These messages can be issued either from the server address space or from a client address space. If the server code that requests the message is running in

cross-memory mode, the message is passed back to a routine that issues the WTO in primary mode in the client address space. This avoids restrictions which apply to WTO messages that are issued in cross-memory mode. For example, cross-memory mode WTO messages do not appear in any job log.

- Command responses, which are issued by WTO with routing codes 2 and 11 and descriptor code 5 (immediate command response).

  These messages contain responses to server commands that are issued via the system MODIFY or STOP command, for example to display statistics.

- Job log messages:

  These messages typically contain diagnostic information, for example details of an abend or an attempted security violation. They are written to the job log, by WTO with routing code 11 (programmer information).

- Trace and statistics messages:

  These messages are written only to the server SYSPRINT file.

All messages that are issued by the server code, whether they are running under the server address space or in cross-memory mode from the client address space, are also copied to the server SYSPRINT file.

AXM messages

The AXM environment code issues operator messages from the server and client address spaces and from the master address space during AXM system services initialization.

These messages are issued using WTO with routing codes 2 (operator information) and 11 (programmer information, not used when running in the master address space), and descriptor code 4 (system status). AXM message numbers are of the form "AXMxxnnnns", where the first five characters "AXMxx" are the name of the module issuing the message, "nnnn" is the numeric part of the message number, and "s" is the suffix letter. The suffix letter is "I" if the message is a routine informational message, or is omitted if the message indicates an error. The suffix letter can be used by automation tools to distinguish routine informational messages from error situations.

AXM messages that are issued from the server environment (via AXM run-time environment routines linked with the server load module), are copied to the server SYSPRINT file as well. AXM also writes informational messages to the SYSPRINT file. These contain information such as initialization information and closedown statistics for storage management, and the main procedure entry point of the server module for diagnostic purposes.

Coupling facility storage management

The type of coupling facility structure that CICS uses for its temporary storage data sharing pools, coupling facility data table pools, and named counter pools is a keyed list structure.

A coupling facility list structure contains an array of numbered lists that contain entries with 16-byte keys. Each list is like a keyed file. Each entry has a fixed prefix area that contains its key and other control information, which includes an adjunct area of 64 bytes for program use. The prefix is followed by a chain of up to 128 256-byte data elements. The maximum data size is 32K bytes.

For named counters, the structure contains a single list, each entry uses only the prefix area, and there are no data elements.

Storage for entry prefixes, known as entry controls, and data elements is allocated from two storage pools within the structure, which are shared between all lists in the structure.

The storage in a list structure can be divided into two types:

- Fixed controls.

  Storage for fixed controls is preallocated at a fixed size for the life of the structure. This storage contains structure control information, including data buffer space, and an array of list headers, up to the maximum number of lists defined when the structure was created. For CICS, this number is the small number of lists used for CICS internal purposes, and the value specified for the parameters **MAXQUEUES** or **MAXTABLES**.

  The fixed controls include enough internal control areas to manage the maximum number of elements and entries that can exist in the structure, given the maximum structure size and the range of possible ratios of entries to elements, and an array of list headers to handle the maximum number of lists that can exist in the structure.

  You cannot increase the maximum size or the maximum number of lists without reallocating the structure, so you must be careful to specify large enough values when the structure is first allocated. However, if you specify a relatively large maximum size or number of lists, a large amount of storage is preallocated for fixed controls, so the initial size of the structure needed to store a given amount of data will be significantly larger than it would be with less generous allowance for expansion.
- Variable controls.

  Variable controls are partitioned into storage areas for entry controls (one per entry) and for data elements. You can adjust this partitioning dynamically by altering the ratio of entries to elements, converting storage for one type to the other type. CICS automatically issues a request to alter the ratio when one type of variable storage is running out but there is enough storage of the other type. You can alter the total size of the storage for variable controls dynamically by changing the size of the structure. This size must be within the range of sizes that are defined for the structure in the active coupling facility resource management (CFRM) policy, which are set at the time that the structure is created. You can change the structure size by using the system operator command `SETXCF ALTER,SIZE`. Alternatively, the operating system can change the structure size, depending on the automatic alter options that are specified in the CFRM policy when the structure is allocated.

In the CICS pool structures, each queue item, data table record, or named counter normally occupies one entry in the structure, together with the appropriate number of data elements. CICS uses additional entries for internal control purposes to maintain the index of currently defined queues or data tables, and to track which lists are currently in use and which previously used lists are now free and available for reuse. Even after all queues or data tables in a pool have been deleted, some entries in the control lists might remain in use.

The amount of storage required for internal control information depends on the level of functionality and performance of the coupling facility control code for the current coupling facility level (CFLEVEL). The storage requirements can increase for a higher CFLEVEL.

A good way to calculate storage requirements for the CICS pool structures is to use the web-based IBM CFSizer tool. See CFSizer. This tool works as follows:

1. It prompts for parameters that describe the amount of information to be stored in the structure.
2. It converts this information to numbers of lists, entries, and elements.
3. It communicates with a coupling facility at a current CFLEVEL to determine the amount of storage required to store the information with the specified amount of free space and room for expansion.

All structure sizes are rounded up to the next storage increment for the coupling facility level (CFLEVEL) at allocation time. For example, sizes are rounded up to the nearest 1 MB for CFLEVEL 16. For more information about the storage increments for different CFLEVELs, see the information about coupling facility control code support in *System z10® Processor Resource/Systems Manager™ Planning Guide*.

For information about the CPC support for different CFLEVELs and the function in each CFLEVEL, see CF levels.

For more information about calculating storage requirements, see the following topics:

- "Storage calculations for temporary storage data sharing" on page 154
- "Storage calculations for coupling facility data tables" on page 173
- "Named counter list structure storage" on page 212

.

Managing the pool structure

It is important to watch out for signs that the pool structure is becoming full, as this might have a serious impact on all the applications that are using the pool.

Monitoring pool structure usage levels

Use the DISPLAY POOLSTAT server command to display the current usage level of the pool structure.

The DISPLAY POOLSTAT command produces messages DFHXQ0432I, DFHF0432I or DFHNC0432I. The most important information in these messages is the maximum percentage of lists, entries, and elements, used during the current statistics interval. For the named counter server, message DFHNC0432I only shows the number of entries because there is always one list and no elements.

Operator messages reporting on pool structure usage

Messages to the operator, for example DFHXQ0411I and DFHXQ0412I, are issued when threshold levels are reached for the numbers of entries or elements used.

Further messages are issued if the pool becomes full. You can set up automated operations processes to watch for these messages and alert your operators to the situation, or take corrective action, if necessary.

You can use the MVS operator command SETXCF ALTER,START with an increased SIZE option, to expand the structure, if the structure has not yet reached the maximum size that is defined in the CFRM policy.

Use of CFRM automatic ALTER to increase pool structure size

The coupling facility resource management (CFRM) policy can specify the keyword ALLOWAUTOALT(YES).

This allows the operating system to issue an ALTER command automatically when the structure is near to becoming full, to increase its size or to adjust the ratio of elements to entries. The threshold at which this happens is specified by the FULLTHRESHOLD keyword in the policy. The default threshold is 80%, which is the same as the default threshold at which the server itself issues an automatic ALTER command to optimize the ratio of entries to elements. The server's automatic ALTER process is more sophisticated, because it takes into account the peak usage of the structure within the current statistics interval, rather than just the current usage. Therefore it is best to ensure that the server's automatic ALTER process is triggered first, by making sure that the percentage values of the server's ENTRYWARN and ELEMENTWARN parameters are at least 5 percent less than the percentage value of the CFRM FULLTHRESHOLD keyword.

Using system-managed rebuild to increase pool structure size

If the structure has not many entries or elements left, but it has already reached its maximum size, you can still use system-managed rebuild to expand it dynamically without closing down the servers, if all the systems using the structure are at a level which supports this function.

First update the CFRM policy to increase the size to the required value, and then activate the updated policy using SETXCF START,POLICY. After this, you can rebuild the structure. The rebuild allocates a new instance of the structure that uses the updated policy, copies across the existing data, and then discards the old instance.

Increasing the number of data lists

If the number of data lists specified via the MAXQUEUES or MAXTABLES server parameter is too small, an attempt to allocate a new data list will fail with message DFHXQ0443 or message DFHCF0443.

There is no way to increase the number of lists without deleting and recreating the structure, which necessitates closing down all of the servers temporarily. You cannot use system-managed rebuild to increase the number of data lists because it copies this number from the existing structure.

You can preserve existing data by using the server program to unload it to a sequential file, and then using SETXCF FORCE to delete the existing structure. You can then use the server program again to reload the data, and allocate a new structure with the appropriate MAXQUEUES or MAXTABLES parameter.

Deleting or emptying the pool structure

If the pool structure is no longer required, or all of the data in the structure is to be discarded, you can delete a pool by closing down all the servers for that pool, and then using the SETXCF FORCE command to delete the structure.

If the server is subsequently started again for the same structure name, an empty structure will be created using the information in the active CFRM policy and the server initialization parameters.

Server connection management

A client CICS region establishes a cross-memory connection to a server the first time that a request refers to the pool for that server. A single connection is established to each server, regardless of the number of tables, queues, or counters that are accessed in the pool.

For coupling facility data table and temporary storage queue servers, a multi-threaded asynchronous connection is established. This connection allows requests to be overlapped up to a fixed maximum number of concurrent requests. Requests that exceed this maximum number are queued within the CICS region until a request thread becomes available.

For named counters, requests are processed synchronously, so only one request can be active at a time for a given client region TCB.

For information about managing failures in coupling facility connectivity, see the section about coupling facility connectivity recovery in *z/OS Parallel Sysplex Recovery*.

Terminating server connections

Each connection has a client side and a server side. If either side is terminated separately, the connection is no longer usable, although the other side might not be terminated until some time later.

For QUASIRENT applications, the connection is associated with the CICS quasi-reentrant (QR) TCB, and so a connection normally remains active until CICS is closed down when the QR TCB is terminated. For threadsafe applications, the connection can be associated with an open TCB and so will remain open until the open TCB is terminated. A connection is closed automatically as part of the resource management termination processing for a TCB. During resource management termination, the connection termination routine on the client side issues a cross-memory call to the server to terminate the connection on the server side as well.

If you want to close down a server after all the CICS regions that are using it have terminated, for example when you are preparing to close down the system, first quiesce the server and then terminate it using the server STOP command (or the equivalent MVS STOP command). Use of the STOP command prevents any new CICS regions from connecting to the server and terminates the server as soon as all the CICS regions that are using it have been terminated.

If you need to close down a server immediately, while CICS regions are still connected to it, use the server CANCEL command, because the normal server STOP command (or the MVS STOP command) is not effective until the connections are terminated. You can also use the MVS CANCEL command to terminate the server, but this prevents the server from going through normal closedown processing.

Note that CICS cannot terminate the client side of the connection automatically. CICS has no way of knowing that the server wants to close down, because although the connection allows CICS to make cross-memory calls to the server, it does not provide any means for the server to notify CICS of asynchronous events. Currently, CICS does not provide any means of terminating a connection on demand except in the case of the named counter server CALL interface which provides a FINISH function for this purpose, but this function is primarily for batch use.

If you terminate the server with CANCEL, the server side of each connection is terminated immediately, but the client side is not affected. The next request from CICS to use the original connection fails and CICS tries to establish a new connection instead. This succeeds if the server has been restarted. However, CICS does not close the old connection explicitly, so when it eventually terminates,

messages are produced not only for the termination of any active connection, but also for the termination of any previous connections to the same server.

Failed server connections

If CICS terminates abruptly, without going through the normal resource manager termination processing for the quasi-reentrant TCB, for example because of a FORCE command or system completion code 40D, an end-of-memory resource manager routine cleans up the client side of the connection.

Because this routine does not run in the CICS region itself, it cannot use the cross-memory connection to notify the server. Therefore, if CICS terminates without carrying out the normal resource manager termination processing for the quasi-reentrant TCB, the server side of the connection might remain active. For coupling facility data tables this might cause problems because the server does not allow the original CICS region to resynchronize after restart if its APPLID is already in use.

From time to time the server checks the client status for each connection, and cleans up the server side of the connection if the client has gone away. This check is done once a minute, and it is also triggered each time a new connection is to be established. Therefore, any connections which have failed are normally cleaned up before the new connection attempts to resynchronize. Because the clean-up processing that terminates the server side of a connection is asynchronous, and might take one or two seconds, the clean-up processing might not be finished in time for the resynchronization from the original CICS region to succeed immediately, but if the resynchronization does not succeed on the first attempt, it should succeed when it is next retried.

Restarting a server

All three types of CICS data-sharing server (temporary storage, coupling facility data tables, and named counters) support automatic restart using the services of the Automatic Restart Manager (ARM).

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

The server normally registers at start-up with ARM, so that it will be restarted automatically if it fails, subject to any rules in the installation ARM policy. If ARM registration fails for any reason except for ARM being unavailable, the server cannot be started. If ARM is unavailable, the server starts normally but has to be restarted manually if it fails.

The servers recognize the ARM return code that indicates that the ARM couple data set has not been formatted for the current MVS system, as being equivalent to ARM being unavailable.

A server does not start if registration fails with return code 8 or above.

When a server starts up, if it is unable to connect to its structure because of some environmental error such as a structure failure, it automatically waits for the structure to become available, using the Event Notification Facility (ENF) to watch for events relating to its structure. This wait occurs before the cross-memory interface is enabled, so the server is not visible to client regions at this time and will appear to be unavailable. While it is waiting, the server can be cancelled using the MVS CANCEL command if it is no longer required.

If the server is running normally, but the coupling facility interface reports a loss of connectivity or a structure failure, the server immediately terminates itself. This disconnects it from the coupling facility, and terminates the server side of any current cross-memory connections from client regions. The server will normally be restarted immediately by the ARM, but will continue to be unavailable to client regions until the coupling facility structure is available again (possibly as a new empty instance of the structure).

An abrupt coupling facility failure such as a power failure may result in a loss of connectivity indication even though the structure has failed, because the operating system cannot determine the state of the structure in that case. This could prevent a new structure from being allocated until the operating system can determine the status of the existing structure, for example after the failed coupling facility has been successfully restarted. If it is certain that the old structure has been lost, but the system has not yet recognized the fact, the operator may be able to save some time by issuing the SETXCF FORCE command to delete the old structure, allowing the system to go ahead and create a new instance of the same structure in a different coupling facility.

You can find more information about automatic restart of coupling facility servers in Automatic restart of CICS data-sharing servers

CICS server support for system-managed processes

The three servers, temporary storage data sharing, coupling facility data tables, and named counters, support system-managed processes for coupling facility list structures.

These system-managed processes are:

- "System-managed list structure rebuild"
- "System-managed list structure duplexing" on page 232

System-managed list structure rebuild

System-managed rebuild allows z/OS to manage the moving of coupling facility structures used for shared temporary storage, coupling facility data tables, and named counter server pools, without recycling the CICS system using them.

Before this, you could move a structure by using server functions to UNLOAD to a sequential data set and then RELOAD elsewhere from the data set, but the switch caused errors in CICS such that restart was recommended, which in some situations was an unacceptable outage.

System-managed rebuild rebuilds the contents of a coupling facility list structure to a new location. The only impact when rebuild occurs is that requests are temporarily suspended within MVS during the few seconds or tens of seconds needed for rebuild processing. The system-managed rebuild process rebuilds only from one structure to another. Therefore, it is not applicable when the original structure has been lost or damaged. It is very useful for planned maintenance and is used for recovery purposes where connectivity to the structure has been lost from one or more systems but at least one system still has access to the original structure. For a loss of connectivity, the system does not initiate a system-managed rebuild automatically, regardless of connectivity options in the policy, but a rebuild can be requested using an operator command.

Any pending requests are made to wait during system-managed rebuild. Apart from the time delay, the application should not notice anything unusual when a system-managed rebuild occurs. For more information about system-managed rebuild, see *z/OS MVS Programming: Sysplex Services Guide*.

CICS supports the MVS system-managed rebuild facility, provided that the SUSPEND=FAIL option of the IXLCONN macro is available. The CICS servers detect automatically whether this support is available.

- If SUSPEND=FAIL support is available, a server connects to its list structure specifying the ALLOWAUTO=YES option of the IXLCONN macro.
- If SUSPEND=FAIL is not available, a server connects with ALLOWAUTO=NO, and the system-managed rebuild facility is not used.

When a server is active, you can use the MVS operator `DISPLAY XCF,STR` command to display the connection details in message IXC360I.

For example, to look at the connection details for structure DFHXQLS_PRODTSQ1, use the following command:

```
Display  XCF,STR,STRNAME=DFHXQLS_PRODTSQ1,CONNAME=ALL
```

To look at the connection details for coupling facility data tables structure DFHCFLS_DTPOOL1, use the following command:

```
Display  XCF,STR,STRNAME=DFHCFLS_DTPOOL1,CONNAME=ALL
```

To look at the connection details for the named counter structure DFHNCLS_PRODNC1, use the following command:

```
Display  XCF,STR,STRNAME=DFHNCLS_PRODNC1,CONNAME=ALL
```

For any of these start commands, the resulting IXC360I message output contains the following lines (in the CONNECTION information section), indicating that system-managed rebuild is enabled for the connection:

```
ALLOW AUTO      : YES
 SUSPEND        : FAIL
```

TS data sharing and CFDT servers

For temporary storage data sharing and coupling facility data tables, any pending requests are made to wait during system-managed rebuild. Apart from the time delay, the application should not notice anything unusual when a system-managed rebuild occurs.

Timeout considerations:

The wait-time for a system-managed rebuild is expected to vary from a few seconds for a small structure to a few tens of seconds for a large structure. This means that transactions can be purged by DTIMOUT, or by an operator command, while waiting on the rebuild.

To minimize the risk of unnecessary problems caused by timeouts during syncpoint processing, the CICS wait exit for CFDT unit of work control functions specify that the wait is not purgeable.

Note: Making the wait for CFDT unit of work control functions non-purgeable does not apply to CICS TS 1.3 regions connected to a later level of the CFDT server that supports system-managed rebuild. In this case, if a transaction makes any recoverable changes to a coupling facility data table before being purged, it will

probably go into a wait again during syncpoint processing. If this happens, it should be left waiting until the rebuild completes, because any further attempt to purge it could result in the UOW being shunted as having failed during syncpoint. If the UOW is shunted in these circumstances, it will require manual intervention to retry syncpoint processing and complete the UOW when the structure becomes available again.

Named counter server

Requests issued using the CALL interface to the named counter server are not made to wait during rebuild, but instead the server returns a new environment error return code with value 311, under the NC_ENVIRONMENT_ERROR category.

An EXEC interface request during rebuild waits using a timer wait and retry loop which can be interrupted by an operator purge command but is not eligible for DTIMOUT, since the wait is definitely known not to be caused by any form of deadlock.

Compatibility considerations:

The named counter client region interface module DFHNCIF normally resides in a linklist library and should be at the highest level of CICS TS in use within an MVS image.

If a CICS region is running a CICS TS 2.2 or later level of DFHNCIF (from the link list) but the EXEC CICS interface is from CICS TS 2.1 or earlier, the named counter server does not wait and retry during rebuild, but instead returns INVREQ with RESP2 code 311.

If a program running in a region that is using a CICS TS 2.1 or earlier DFHNCIF issues a request to a CICS 2.2 or later level of the server during a rebuild, CICS returns error code 301, UNKNOWN_ERROR. For the EXEC interface, this is mapped to an INVREQ with RESP2 code 301.

System-managed list structure duplexing

System-managed coupling facility duplexing provides a general-purpose, hardware-assisted mechanism for duplexing coupling facility structure data.

It is a robust mechanism that provides recovery from failures such as loss of a single structure or coupling facility, or from loss of connectivity, by a rapid switch to the other structure in a duplex pair.

Unlike system-managed rebuild, which can rebuild only from one structure to another, and is not applicable when the original structure has been lost or damaged, system-managed duplexing creates and maintains a duplexed copy of a structure in advance of any failure. It is largely transparent to the users of the structure, and is available for both planned and unplanned outages of a coupling facility or structure. For example, until CICS TS 2.2, coupling facility data tables and temporary data sharing structures have contained mostly “scratch-pad” information that is not critical to recover. With duplexing, you can use the coupling facilities to store more critical information.

Transactions that are accessing a duplexed structure might experience delays, which could result in the DTIMOUT threshold being reached, when:

- A structure is quiesced by MVS while duplexing is being established.

- A structure is quiesced as a a result of an operator command to stop or start duplexing.
- A structure is switched to the secondary structure or back to the primary structure.

A newly-started data sharing server is not allowed to connect to a structure during any phase of a duplexing rebuild until the duplex established phase is reached.

System-managed duplexing is built on the system-managed rebuild function. However, whereas system-managed rebuild requires OS/390 with APAR OW39892, system-managed duplexing requires z/OS Version 1 Release 2 with an enabling APAR. You also need to specify the DUPLEX option, in the CFRM policy information for the structure, as either DUPLEX(ENABLED) or DUPLEX(ALLOWED). If you specify DUPLEX(ENABLED), MVS will initiate and attempt to maintain a user-managed duplexing operation for the structure. If you specify DUPLEX(ALLOWED), an operator can start duplexing, using the SETXCF START,REBUILD,DUPLEX command. You also need to ensure that coupling facilities that are taking part in duplexing operations communicate with one another through a “peer link”.

For more information about system-managed duplexing, see *z/OS MVS Programming: Sysplex Services Guide* and *z/OS MVS Setting Up a Sysplex*.

Chapter 4. Defining resources

After you have installed CICS, you must define and install all the resources that your CICS regions require to run user transactions.

For a list of resources, see CICS resources: listing, syntax, and attributes.

Resource definition online (RDO) transaction CEDA

Use the CEDA transaction to add, remove, or change resource definitions online.

The resource definition batch utility DFHCSDUP

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

Note that the DFHCSDUP utility opens the CSD in non-RLS mode, even when you request RLS access on your JCL. This means that, if you access the CSD from CICS in RLS mode, it cannot be open when you run DFHCSDUP. The reason for the restriction is that the DFHCSDUP utility does not have the capabilities that are needed in order to open a recoverable file in RLS mode. The restriction also applies, however, if your CSD is nonrecoverable.

How to invoke DFHCSDUP

You can invoke the DFHCSDUP program in two ways:

- As a batch program
- From a user program running either in batch mode or in a TSO environment

DFHCSDUP commands overview

You can use DFHCSDUP commands to do the following tasks:

- Add a group to the end of a named list in a CSD file (ADD command)
- Alter attributes of an existing resource definition. on the CSD (ALTER command)
- Append a group list from one CSD file to a group list in another, or in the same, CSD file (APPEND command)
- Copy all of the resource definitions in one group to another group in the same, or in a different, CSD file (COPY command)
- Copy a single resource definition from one group to another group (COPY command)
- Define a single resource, or a group of resources, on the CSD (DEFINE command)
- Delete from the CSD a single resource definition, all of the resource definitions in a group, or all of the group names in a list (DELETE command)
- Extract data from the CSD and pass it to a user program for processing (EXTRACT command)

© Copyright IBM Corp. 1989, 2017

- Initialize a new CSD file, and add to it the CICS-supplied resource definitions (INITIALIZE command)
- List selected resource definitions, groups, and lists (LIST command)
- Process an APAR—that is, apply maintenance for a specific APAR to the CSD (PROCESS command)
- Remove a single group from a list on the CSD file (REMOVE command)
- Scan all IBM-supplied and user-defined groups for a resource (SCAN command)
- Service a CSD file when necessary (SERVICE command)
- Upgrade the CICS-supplied resource definitions in a primary CSD file for a new release of CICS (UPGRADE command)
- Define resources using a set of user-defined default values (USERDEFINE command)
- Verify a CSD file by removing internal locks on groups and lists (VERIFY command)

For information about each of these commands, see Resource management utility DFHCSDUP commands.

Sharing the CSD between CICS Transaction Server for z/OS, Version 5 Release 4 and earlier releases

If you want to share the CSD between CICS regions at different release levels, to enable you to share common resource definitions, you must update the CSD from the higher level region - CICS Transaction Server for z/OS, Version 5 Release 4 .

About this task

In CICS Transaction Server for z/OS, Version 5 Release 4 , some attributes are obsolete, and are removed from the CSD definitions.

Procedure

- Use the **ALTER** command on definitions that specify obsolete attributes. This does not cause the loss of these attributes in CICS Transaction Server for z/OS, Version 5 Release 4 , so you can safely update resource definitions from a CICS TS for z/OS, Version 5.4 region.
- If you are sharing the CSD between a CICS TS for z/OS, Version 5.4 region and a region at an earlier release of CICS, you can use the CICS TS for z/OS, Version 5.4 CSD utility, DFHCSDUP, to update resources that specify obsolete attributes.
  1. Specify if you want to use the compatibility option. Use the **PARM** parameter on the EXEC PGM=DFHCSDUP statement, using the option COMPAT. The default is NOCOMPAT, which means that you cannot update obsolete attributes. (See Figure 50 on page 237.)
  2. You cannot use the **EXTRACT** command of the DFHCSDUP utility in this release when the COMPAT option is specified.

What to do next

CSD compatibility between different CICS releases in Upgrading discusses these obsolete attributes and their compatibility with earlier releases.

Invoking DFHCSDUP as a batch program

The example job shows you the job control statements that you can use to invoke DFHCSDUP as a batch program.

About this task

```
//CSDJOB  JOB  accounting info,name,MSGLEVEL=1
//STEP1   EXEC PGM=DFHCSDUP,REGION=0M,
//             PARM='CSD(READWRITE),PAGESIZE(60),NOCOMPAT'
//STEPLIB DD   DSN=CICSTS54.CICS.SDFHLOAD,DISP=SHR
//DFHCSD   DD  DISP=SHR,DSN=CICSTS54.CICS.DFHCSD
//SECNDCSD DD  UNIT=SYSDA,DISP=SHR,DSN=CICSTS54.CICS.SECNDCSD
//indd     DD  UNIT=SYSDA,DISP=SHR,DSN=extract.input.dataset
//outdd    DD  UNIT=SYSDA,DISP=SHR,DSN=extract.output.dataset
//* or
//outdd    DD  SYSOUT=A
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  *
⋮
     DFHCSDUP commands
/*
//
```

Figure 50. Sample job to run DFHCSDUP

To use this job, you need to edit it as follows:

Procedure

1. Change the EXEC statement to specify a suitable REGION size and a **PARM** parameter:

   Use the **PARM** parameter to specify any of the following options:

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

   CSD({READWRITE|READONLY})
   Specifies whether you want read and write access or read-only access to the CSD from this batch job. The default value is READWRITE.

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

   NOCOMPAT or COMPAT
   Specifies whether the DFHCSDUP utility program is to run in compatibility mode (that is, whether it can update definitions that are obsolete in CICS Transaction Server for z/OS, Version 5 Release 4). The default is NOCOMPAT, which means that you cannot update obsolete attributes. For more information about this option, see “Sharing the CSD between CICS Transaction Server for z/OS, Version 5 Release 4 and earlier releases” on page 236.

2. If you specify the **FROMCSD** parameter on an APPEND, COPY, or SERVICE command, add a SECNDCSD DD statement to specify the secondary CSD. The ddname for this DD statement is the name that you specify on the **FROMCSD** parameter. The secondary CSD must be a different data set from the primary; you must not define primary and secondary DD statements that reference the same data set.
3. If you specify the EXTRACT command, you might need to take one or both of the following actions:
   a. Concatenate with STEPLIB the libraries that contain your USERPROGRAM programs.

b. Include a DD statement for any input data set that is defined in your user program. This statement is represented in the figure with the ddname *indd*. For example, the supplied user program, DFH$CRFA, needs a DD statement with a ddname of CRFINPT.

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

   PROGRAM
   TRANSACTION
   TYPETERM
   XTPNAME
   DSNAME

   For programming information about the use of the CRFINPT file by the programs DFH$CRFx or DFH0CRFC (for COBOL), see The sample EXTRACT programs.

4. If you specify the EXTRACT command, include the DD statements for any data sets that receive output from your extract program. The figure shows two options, to write the data set to disk or to SYSOUT. The ddname is whatever ddname you define in the user program. The figure uses the ddname *outdd*. The sample programs that are supplied with CICS need DD statements for the following ddnames:

Table 13. DD statements for the supplied sample programs

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

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

   For programming information about the user programs that are supplied with CICS, see The sample EXTRACT programs.
6. In the SYSIN data set, you can code commands and keywords by using abbreviations and mixed case, as given in the syntax box in the description of

each command. If you prefer, you can use a data set or a partitioned data set member for your commands, rather than coding them in the input stream. Keyword values can be longer than one line, if you use the continuation character (an asterisk) at the end of a line (in column 72). Subsequent lines start in column 1. For example, you can use this facility to specify XTPNAME values of up to 128 hexadecimal characters. If you enter an ambiguous command or keyword, the DFHCSDUP program issues a message to indicate the ambiguity.

Invoking the DFHCSDUP program from a user program

Invoking the DFHCSDUP program from a user program enables you to create a flexible interface to the utility.

About this task

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

You can run your user program:

- In batch mode
- Under TSO.

Note:

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

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

   Your put-message exit routine can terminate the DFHCSDUP program, (You **must** supply a put-message routine if you want your operators to regain control after an ATTENTION interrupt.)
2. Suitably authorized TSO users can use the CEDA INSTALL transaction to install resources that have previously been defined with the DFHCSDUP program.

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

The following sections outline the entry parameters of the DFHCSDUP program and the responsibilities of the user program. For programming information about invoking the DFHCSDUP program from a user program, see User programs for the system definition utility program (DFHCSDUP).

Entry parameters for the DFHCSDUP program

When invoking the DFHCSDUP program, your program passes a list of up to five parameters.

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

Note: A maximum of three options may be specified:

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

CSD({READWRITE|READONLY})
specifies whether you require read/write or read-only access to the CSD. The default value is READWRITE.

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

NOCOMPAT|COMPAT
specifies whether the DFHCSDUP utility program is to run in compatibility mode (that is, whether it can update definitions that are obsolete in Version 5 Release 4). The default is NOCOMPAT, which means that you cannot update obsolete attributes. For further information about this option, see "Sharing the CSD between CICS Transaction Server for z/OS, Version 5 Release 4 and earlier releases" on page 236.

DDNAMES
A list of ddnames that, if specified, are substituted for those normally used by the DFHCSDUP program.

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

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

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

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

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

Responsibilities of the user program

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

- AMODE(24) and RMODE(24) are in force
- S/370 register conventions are obeyed
- If the **EXITS** parameter is passed, any programming environment needed by the exit routines has been initialized

- Any ACBs or DCBs passed for use by the DFHCSDUP program are OPEN.

Command processing in DFHCSDUP following internal error detection

If you have provided a put-message-exit routine for the DFHCSDUP program, it is invoked whenever a message is issued. You can use this exit to respond to error messages produced by DFHCDSUP processing, when the DFHCSDUP program is invoked from a user program.

The put-message-exit routine is not used if the DFHCSDUP program is running as a batch program. For programming information about the DFHCSDUP exits, see User programs for the system definition utility program (DFHCSDUP).

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

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

1. If the error occurs during connection of the CSD, no subsequent commands are completed.
2. If the error occurs elsewhere, no subsequent commands are executed other than LIST commands.

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

Autoinstall

Autoinstall is a method of creating and installing CICS resources dynamically as they are required. You can use autoinstall for SNA LUs, MVS consoles, APPC connections, IPIC connections, programs, map sets, partition sets, and journals.

With autoinstall, you do not have to define and install every resource that you intend to use. Instead, CICS dynamically creates and installs a definition for you when a resource is requested. CICS bases the new definition on a *model* definition that you provide.

You can use an *autoinstall control program* to control the way autoinstall operates in your CICS region.

Using autoinstall saves the time that is otherwise used to define and install every resource, and it saves storage, because each installed resource occupies storage whether the resource is being used or not.

Autoinstall models

You must provide at least one *model* resource definition for each type of resource to be autoinstalled.

When a resource is requested that does not have an installed definition, CICS creates a definition based on what you have specified in the model. You can have more than one model, depending on what properties you want the autoinstalled resources to have; for example, if you had 500 terminals all with the same

properties and another 500 with a different set of properties, you could have two model terminal definitions, one for each of the two sets of properties.

Autoinstall control program

You control autoinstall by means of an *autoinstall control program*, either user-written or supplied by CICS. This program is responsible for, among other things, providing the model name or names to CICS and, providing z/OS Communications Server information to CICS for autoinstall for terminals, and so on.

CICS provides several autoinstall control programs; one for terminals; one for MVS consoles; one for connections; and one for programs, map sets, and partition sets. You can use the CICS-supplied ones or you can customize them to suit your installation.

Autoinstalling z/OS Communications Server terminals

How to use autoinstall for z/OS Communications Server terminals and connections.

For information on autoinstalling MVS consoles, see "Autoinstalling MVS consoles" on page 253.

For information on autoinstalling connections and parallel sessions, see "Autoinstalling APPC connections" on page 257.

For programming information about the autoinstall control program, see Writing a program to control autoinstall of LUs.

Deciding which terminals to autoinstall

This section will help you to decide which terminal devices to autoinstall.

Automatic transaction initiation

If a BMS ROUTE message is sent to a terminal that has a TCT entry but is not logged on, CICS saves the message for subsequent delivery.

In addition, if the TCT entry so specifies, CICS attempts to acquire the terminal and log it on for that purpose. CICS attempts to satisfy other ATI requests (EXEC CICS START or a transient data trigger level) in the same way.

The use of autoinstall for printers is severely limited because almost all transactions for printers are initiated automatically. It is possible to autoinstall printers, however, if you arrange for them to be logged on. Entering VARY NET,...,LOGON as a console command does this.

For an autoinstalled terminal, a TCT entry **may** be available even when the terminal is logged off. This happens only if the terminal has been logged on earlier in the CICS run, and depends on the TYPETERM definition, the system initialization parameters, and the SNT entry for the last user of the terminal. For details, see "Automatic sign-off, logoff, and TCTTE deletion" on page 251. If a TCT entry exists, an autoinstalled terminal can accept an ATI request just like an individually defined terminal.

If you choose autoinstall for terminals that might receive ATI requests, make use of the AUTOCONNECT attribute on the TYPETERM definition for your models. AUTOCONNECT(YES) means that the terminal is logged on to CICS automatically at an emergency restart (see Figure 54 on page 253), by CICS requesting a z/OS Communications Server SIMLOGON (simulated logon). (Because this requires a TCT entry, it does not happen at cold or warm start.)

You may find that setting up a printer-owning region is the best approach, especially if you have distributed printing with many small printers.

Whether or not you autoinstall your printers, you can associate a printer and an alternate printer with a display device. The association is made when the display device is autoinstalled. Definitions for these printers need not have been installed at the time the display device is autoinstalled, but they must exist at the time of use.

The TCT user area (TCTUA)

The TCT user area is an optional extension to the TCT entry.

The TCTUA is available for application use (the rest of the TCT entry belongs to CICS). It has traditionally been used for two purposes:

- To pass data from one transaction of a pseudo-conversational sequence to the next.
- To maintain user profile information and statistics during a terminal session. (This is not necessarily a z/OS Communications Server session, but a period of access to a particular application, as defined by the application.)

The first use has gradually been supplanted by COMMAREA and other CICS facilities, but the second is still fairly common. An application may store statistics, such as teller totals, in the TCTUA, which is initialized by a PLTPI program at the beginning of execution and retrieved by a PLTSD program at termination (shutdown). Autoinstall does not provide the ability to save this information between logoff and logon, because the TCTUA does not exist then. In addition, the TCTUA is not available to PLTPI and PLTSD programs at system initialization and termination. A new technique must be devised to allow the initialization of TCTUA and user data when the user logs on or logs off.

As noted earlier, the autoinstall process creates the TCT entry (including the TCTUA) before the first transaction is executed at the terminal, but after the autoinstall control program has done its initial execution. Thus you cannot access the TCTUA in the autoinstall control program and so any TCTUA initialization must be done later. You could write your own 'good morning' transaction to do this, or use the first transaction of the application in question.

Furthermore, the autoinstall control program does not have access to the TCTUA at logoff either, because CICS deletes the TCT entry (including the TCTUA) before invoking this program. Therefore, if an application needs to capture statistics or other information from such a TCTUA, it must get access before CICS does this. The place to do this is in the **node error program (NEP)**, the user-written component of the terminal error processing routines, because CICS drives the NEP exit before it deletes the TCT entry.

The terminal list table (TLT)

A terminal list table is a list of terminals, defined either by four-character CICS terminal names or by three-character CICS operator identifiers.

It is used principally for routing messages to multiple destinations and for giving limited operational control of a group of terminals to a supervisor. Both of these uses must be rethought in an autoinstall environment. If a TLT lists terminals that do not have TCT entries, because they are not logged on at the time the TLT is used, supervisory operations against those terminals fails. For example, you cannot put a nonexistent TCT entry into or out of service.

Similarly, message routing works differently for individually defined terminals and for autoinstalled terminals. When a message is sent to an individually defined terminal that is not logged on, the message is saved in temporary storage, and delivered when the terminal logs on. When a message is sent to a terminal that is not defined because it is an autoinstalled terminal and is not logged on, CICS gives a route fail condition, indicating that it knows nothing of that terminal. Indeed, if terminal names are generated and assigned randomly, as they may be in an autoinstall environment, the whole TLT mechanism breaks down.

Transaction routing

An autoinstall request to a local system overrides an autoinstall request for the same definition shipped from a different system.

If transaction routing can occur between two CICS systems, any terminal that can log on to both should be installed in both in the same way. Such a terminal should be:

- Autoinstalled in both systems, or
- Defined to each system at initialization

Autoinstall and output-only devices

Most of the benefits of autoinstall apply more to display devices than to printers.

Displays initiate transactions in CICS; usually, any transaction output is returned to the input terminal, so that neither CICS nor the application needs to know any other NETNAME than that of the display, which identifies itself in the process of logging on.

On the other hand, when output is directed to output-only printers, either CICS or the application must know what NETNAME to use, and this implies that some knowledge of the network is maintained somewhere. The primary and alternate printer names in an individually-defined TCT entry constitute this kind of information, as maintained by CICS. In the case of autoinstalled terminals, corresponding information—if it is required—must be maintained in tables or files, embedded in the application, supplied by z/OS Communications Server ASLTAB and ASLENT model terminal support (MTS), or supplied dynamically by the user.

Autoinstall and z/OS Communications Server

This topic explains how autoinstall works with z/OS Communications Server. It is intended to help you understand the processing that takes place when you use autoinstall.

The process of logging on to CICS using autoinstall

(The process is illustrated in Figure 51 on page 246 and Figure 52 on page 247.) CICS supports the model terminal support (MTS) function of z/OS Communications Server. Using MTS, you can define the model name, the printer (PRINTER), and the alternate printer (ALTPRINTER) for each terminal in a z/OS Communications Server table. CICS captures this information as part of autoinstall processing at logon, and uses it to create a TCTTE for the terminal. If you are

using MTS, you must use a version of DFHZATDX that is suitable for use on CICS Transaction Server for z/OS. See Writing a program to control autoinstall of LUs for programming information about the user-replaceable autoinstall program.

1. z/OS Communications Server receives your request, determines that you want to use CICS, and passes your request to CICS.
2. CICS extracts the NETNAME name of your terminal from the logon data. CICS searches the TCT for an entry with the same NETNAME.
3. If it finds such an entry, CICS issues an OPNDST to z/OS Communications Server to establish a session between CICS and the terminal. This is the normal CICS logon process.
4. If it fails to find a matching entry, CICS checks the system initialization parameters that were specified in the SIT, or reset by using CEMT, to check whether it can allow an autoinstall.
5. If the system initialization parameters allow an autoinstall, CICS checks the terminal data that is passed by z/OS Communications Server, to check whether the terminal is eligible for autoinstall.
6. If the terminal is eligible, CICS examines the bind image to see whether it carries sufficient information.
7. If the z/OS Communications Server bind image data proves sufficient, CICS searches the autoinstall model table (AMT) in sorted ascending order, and autoinstalls the terminal in one of the following ways:
   - If z/OS Communications Server supplies CICS with a valid model name, CICS passes this name to the **autoinstall control program**. (If the logon request comes to CICS through z/OS Communications Server, and if you supply z/OS Communications Server with names of model terminals, CICS can obtain the name of the model terminal from the logon data.)
   - If z/OS Communications Server does not supply CICS with a valid model name, CICS searches the AMT for suitable autoinstall models and passes them to an **autoinstall control program**, together with z/OS Communications Server logon data. If z/OS Communications Server supplies CICS with an invalid model name, message DFHZC6936 results.
8. The autoinstall control program (either the CICS-supplied program or one written by you) selects one of the models and provides the rest of the information necessary to complete a TCT entry for the terminal.
9. When the autoinstall control program returns control, CICS builds a TCT entry by using the autoinstall model, the data returned by the autoinstall control program, and the z/OS Communications Server logon data for the terminal. CICS then adds the new entry to the TCT and issues an OPNDST to z/OS Communications Server to establish a session between CICS and the terminal.
10. If the TYPETERM definition so specifies, CICS uses the QUERY function to find out about some of the features of the terminal. (These features are listed in TYPETERM resources.)

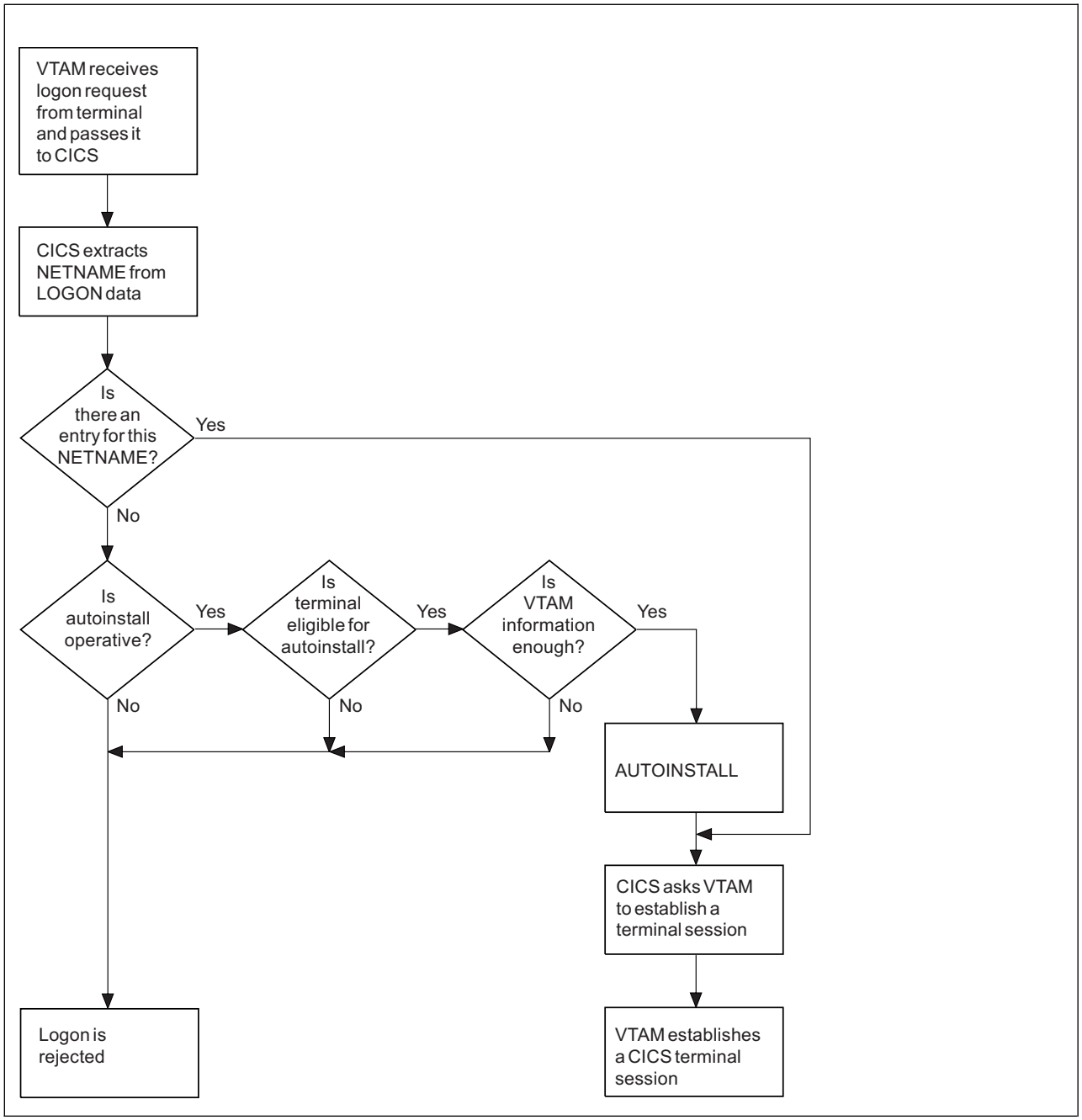


Figure 51. The process of logging on to CICS using autoinstall.

The autoinstall process itself is shown as a single box. What happens inside this box is depicted in Figure 52 on page 247

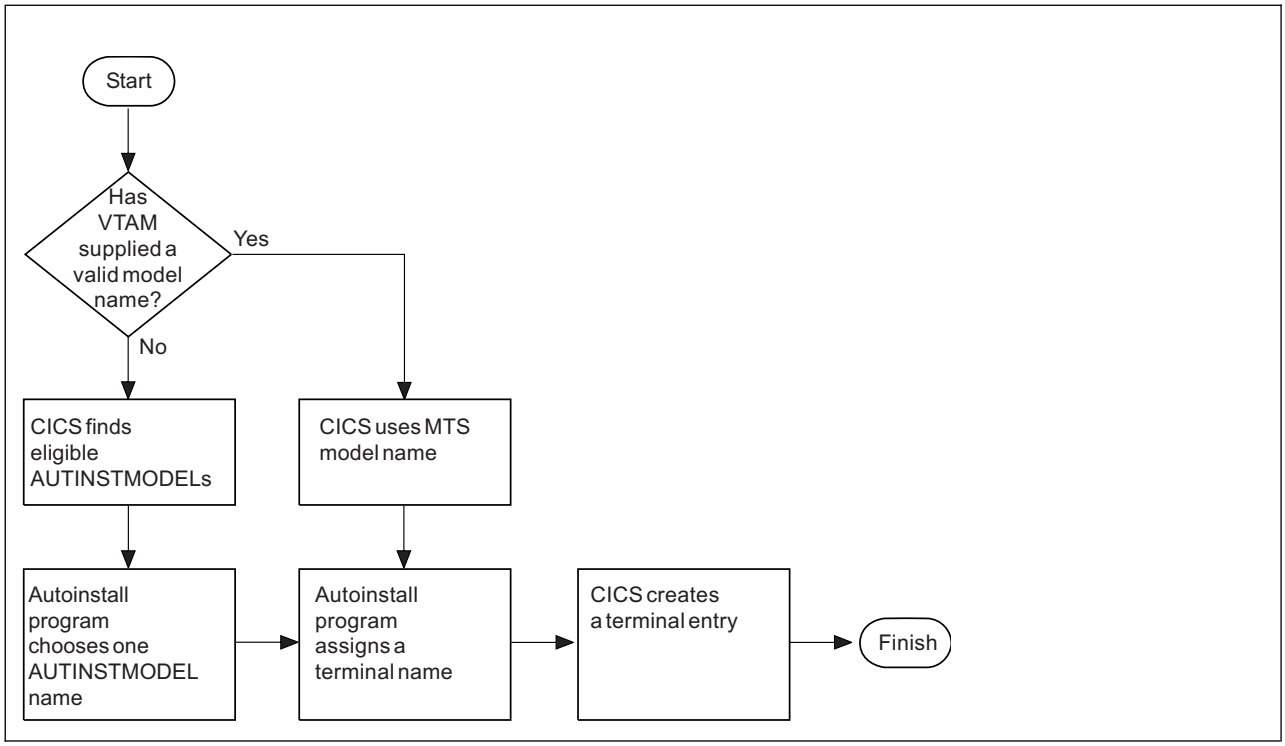


Figure 52. How CICS autoinstalls a terminal.

The overall process in which this fits is shown in Figure 51 on page 246 .

What happens when the user logs off

1. CICS issues a CLSDST to ask the z/OS Communications Server to end the session.
2. CICS attempts to delete the TCT entry after a delay specified in the AILDELAY system initialization parameter.

   The TCT entry cannot be deleted if there is a lock effective at the time. For instance, **CEMT INQUIRE TERMINAL** or an outstanding ATI request locks the TCT entry.
3. CICS gives control to the autoinstall control program in case it has any further processing to do; for example, to free the TERMINAL name it gave to the terminal.

If the TYPETERM definition of the terminal specifies SIGNOFF(LOGOFF) and the RACF segment specifies timeout, expiry of the timeout period causes the terminal to be logged off and the user to be signed off. After this automatic logoff, the delay period commences, leading to deletion of the TCT entry unless the terminal logs on again before the period ends. For details, see “Automatic sign-off, logoff, and TCTTE deletion” on page 251.

Implementing z/OS Communications Server autoinstall

You need to perform a number of steps to set up autoinstall for your z/OS Communications Server terminals.

Procedure

1. Determine whether your terminals are eligible for autoinstall.

   The following terminals can be autoinstalled:

- z/OS Communications Server locally attached 3270 terminals (non-SNA), both displays and printers
- z/OS Communications Server logical unit type 0 terminals
- z/OS Communications Server logical unit type 1 terminals, including SCS printers
- z/OS Communications Server logical unit type 2 terminals
- z/OS Communications Server logical unit type 3 terminals
- z/OS Communications Server logical unit type 4 terminals
- z/OS Communications Server logical unit type 6.2 single-session terminals
- TLX or TWX terminals using NTO

The following terminals cannot be autoinstalled:
- Pipeline terminals
- Automatic teller machines (3614 and 3624)
- Non-z/OS Communications Server resources
- z/OS Communications Server logical unit type 6.1 ISC and MRO sessions

2. Decide whether you can benefit from using autoinstall.

   You are likely to benefit from autoinstall if your system has:
   - A significant number of z/OS Communications Server terminals
   - Frequent changes to your network
   - Many z/OS Communications Server terminals logged off much of the time
   - Many z/OS Communications Server terminals using other applications much of the time
   - Many z/OS Communications Server terminals that need access to multiple, but unconnected, CICS systems

   Autoinstall might be less beneficial if you have:
   - A small, static network
   - Many terminals more or less permanently logged on to one CICS system
   - Many terminals logging on and off frequently
   - Many terminals logging on and off at the same time

3. Decide which devices to autoinstall.

   This decision depends on how you use your z/OS Communications Server terminals. For example, a terminal that is logged on all the time can be autoinstalled, but you might choose to define it individually.

   An autoinstall logon is slower than a logon to a terminal individually defined to CICS, so if you switch continually between applications and have to log on to CICS frequently, you may require individual definitions for some terminals.

   You should also consider your use of automatic transaction initiation (ATI), terminal list tables (TLTs), and the intercommunication methods in use in your installation. See “Deciding which terminals to autoinstall” on page 242.

4. Create your TYPETERM and model TERMINAL definitions.

   CICS supplies some TERMINAL and TYPETERM definitions; these are listed in TYPETERM definitions in group DFHTYPE and Model TERMINAL definitions in group DFHTERM. You can use these definitions if they are suitable; if not, create your own using CEDA or DFHCSDUP.

   Define an autoinstall model for each different kind of terminal to be autoinstalled. Try to keep the number of definitions to a minimum, so that the autoinstall control program can be as simple as possible.

   When you create your definitions, consider whether you want to use the QUERY structured field (see TYPETERM attributes). It can help the autoinstall control program to choose which model on which to base a definition, and so speed up the autoinstall process.

5. Redefine DFHZCQ. For every region using autoinstall, redefine DFHZCQ to be RESIDENT(YES). (DFHZCQ is in the CICS-supplied group DFHSPI). See

Defining programs as resident, nonresident, or transient for guidance on why you should consider making programs resident.

6. Ensure that your z/OS Communications Server LOGMODE table entries are correct. "Autoinstall and z/OS Communications Server" on page 244 explains the relationship between CICS autoinstall and z/OS Communications Server. For programming information, including a list of z/OS Communications Server LOGMODE table entries, see Coding entries in the VTAM LOGON mode table.
7. Design and write an autoinstall control program.

   The terminal autoinstall control program is invoked by CICS every time there is a valid request for a TCT entry to be autoinstalled, and every time an autoinstalled TCT entry is deleted.

   For programming information about the autoinstall control program, see Writing a program to control autoinstall of LUs.

   Before beginning your program, look at the CICS-supplied autoinstall control program DFHZATDX in group DFHSPI to see if it is suitable for what you want to do with autoinstall.
8. Enable terminal autoinstall.

   You can enable autoinstall for terminals either by specifying suitable system initialization parameters, or by using the EXEC CICS or CEMT SET and INQUIRE SYSTEM commands.

   Five system initialization parameters relate to terminal autoinstall:

   AIEXIT
   specifies the name of the autoinstall program to be used. It defaults to DFHZATDX, the name of the IBM-supplied autoinstall control program.

   AIQMAX
   specifies the maximum number of terminals that can be queued concurrently for autoinstall. When this limit is reached, CICS requests z/OS Communications Server to stop passing LOGON and BIND requests to CICS until CICS has processed one more autoinstall request.

   The purpose of the limit is to protect the system from uncontrolled consumption of operating system storage by the autoinstall process, as a result of some other abnormal event. Normally, in the process of autoinstall, the principal consumer of CICS storage is the autoinstall task (CATA) itself. The amount of CICS storage consumed by the autoinstall process during normal operation can therefore be controlled by creating an appropriate TRANCLASS definition to limit the number of autoinstall tasks that can exist concurrently.

   AILDELAY
   specifies the time interval, expressed as hours, minutes, and seconds (hhmmss), which elapses after an autoinstall terminal logs off before its TCTTE is deleted. The default value is 0, indicating that TCTTEs are deleted at logoff time and at warm shutdown as CLSDST is issued for the autoinstall terminals still in session. Specifying an AILDELAY interval permits the TCTTE to be reused should the terminal log back on before the interval has expired.

   AIRDELAY
   specifies the time interval, expressed as hours, minutes, and seconds (hhmmss), which elapses after emergency restart before terminal entries are deleted if they are not in session. The default value is 700, indicating a restart delay of 7 minutes.

GRPLIST
specifies the list or lists containing the group or groups of autoinstall models created.

For information on how to specify these system initialization parameters, see Specifying CICS system initialization parameters.

Three options relate to terminal autoinstall on the INQUIRE and SET AUTOINSTALL command:

CUR(*value***)**
specifies the number of autoinstall logon requests that are currently being processed.

MAXREQS(*value***)**
specifies the largest number of autoinstall requests that are allowed to queue at one time, in the range 0-999.

You can prevent more terminals from logging on through autoinstall by setting this value to 0. This allows autoinstalled entries for terminals currently logged on to be deleted by the autoinstall program when they log off.

PROGRAM(*pgrmid***)**
specifies the name of the user program that is controlling the autoinstall process. The default is the CICS-supplied program DFHZATDX.

Recovery and restart of autoinstalled terminal definitions

This topic explains what happens to autoinstalled terminal definitions at logoff and system restart times.

What happens at CICS restart

At an emergency restart, autoinstalled TCT entries are recovered unless you specify a restart delay period of zero in the **AIRDELAY** system initialization parameter.

Users can log on again after an emergency restart without going through the autoinstall process. Those terminals with AUTOCONNECT(YES) specified in their TYPETERM definition are automatically logged on during the restart process, without the need for operator intervention. The recovery of autoinstalled TCT entries avoids the performance impact of many concurrent autoinstall requests following a CICS restart. A terminal user logging on after restart uses the TCT entry created for that terminal's NETNAME during the previous CICS run. It is just as if that terminal had an individual TERMINAL definition installed in the TCT.

Because this could pose a threat to security, CICS checks for operator activity after recovery. After a delay, all autoinstalled TCT entries that were recovered but are not in session again are deleted. As well as improving security, this ensures that CICS storage is not wasted by unused TCT entries. You can specify the length of the delay using the system initialization parameters.

If z/OS Communications Server persistent sessions support is in use for the CICS region and AIRDELAY is not equal to zero, autoinstalled TCT entries are treated exactly like other TCT entries.

At a warm start, TCT entries that were previously autoinstalled are lost, unless you logoff and CICS is shut down before the AILDELAY time has expired.

If a TCTTE is recovered during emergency restart, a specification of AUTOCONNECT(YES) prevents deletion of the TCTTE by AIRDELAY. If you want the TCTTE storage to be deleted in this case, specify AUTOCONNECT(NO).

Automatic sign-off, logoff, and TCTTE deletion

If a session ends through expiry of the user's TIMEOUT period, the terminal entry is deleted as described in the preceding section only if SIGNOFF(LOGOFF) is specified in the TYPETERM definition of the model.

Table 14 summarizes the automatic deletion and recovery of TCTTEs for autoinstalled terminals.

Figure 53 on page 252 shows how automatic sign-off, logoff, and TCTTE deletion occur if a session is timed out. Figure 54 on page 253 shows how logon and TCTTE deletion occur if, at a warm start or emergency restart, a TCTTE exists for an autoinstalled terminal. Table 15 on page 253 shows how automatic TCTTE deletion occurs if a session ends for any reason other than timeout.

Table 14. AUTOINSTALL—summary of TCTTE recovery and deletion

CICS RUNNING:	
Restart delay = 0	TCTTE entries are not cataloged and therefore cannot be recovered in a subsequent run.
Restart delay > 0	TCTTE entries are cataloged. If they are not deleted during this run or at shutdown, they can be recovered in a subsequent emergency restart.
SHUTDOWN:	
Warm	Terminals are logged off and their TCTTEs are deleted, except for those terminals which were logged off before shutdown but whose AILDELAY has not expired.
Immediate	No TCTTEs are deleted. All can be recovered in a subsequent emergency restart.
Abnormal termination	No TCTTEs are deleted. All can be recovered in a subsequent emergency restart.
STARTUP:	
Cold	No TCTTEs are recovered.
Warm	No TCTTEs are recovered.
Emergency restart when restart delay = 0	No TCTTEs are recovered (but see note in Figure 54 on page 253). This session is unbound if it persists.
Emergency restart when restart delay > 0	TCTTEs are recovered. For details see Figure 54 on page 253. This session is recovered if it persists.

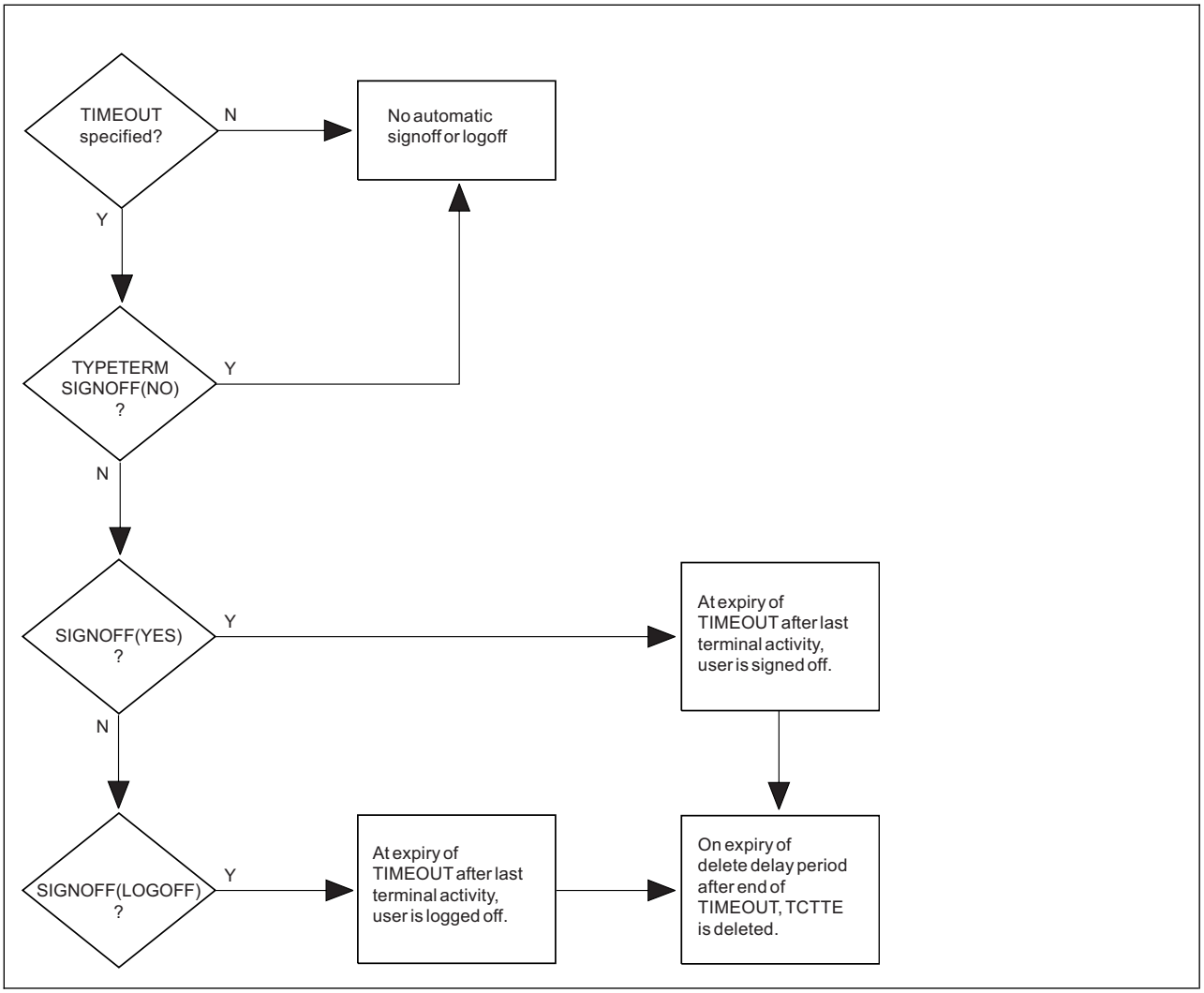


Figure 53. Automatic sign-off, logoff and deletion of autoinstalled terminals.

When a session is timed out because there is no terminal activity, CICS uses these steps to determine whether to delete an autoinstalled terminal definition:

1. If no timeout is specified in the RACF segment, the user is not automatically signed off or logged off.
2. If SIGNOFF(NO) is specified in the TYPETERM definition, the user is not automatically signed off or logged off.
3. If SIGNOFF(YES) is specified in the TYPETERM definition, and a timeout is specified in the RACF segment, the user is signed off when the timeout period has expired.

   Before the timeout period has expired, the user can resume terminal activity.
4. If SIGNOFF(LOGOFF) is specified in the TYPETERM definition, and a timeout is specified in the RACF segment, the user is signed off and the terminal is logged off. The terminal definition is deleted after a further interval, which is specified in the AILDELAY attribute of the TYPETERM definition.

   Before the timeout period has expired, the user can resume terminal activity. After the timeout has expired, but before the terminal definition is deleted, the user can log on again without the overhead of the autoinstall process.

Table 15. AUTOINSTALL

AUTOINSTALL—automatic TCTTE deletion after non-automatic logoff		
Non-automatic LOGOFF: z/OS Communications Server informs CICS of a session outage, terminal logs off, or transaction disconnects terminal	<---------**Delete delay period**---------> The terminal can log on during this period without repeating the autoinstall process.	TCTTE entry deleted.

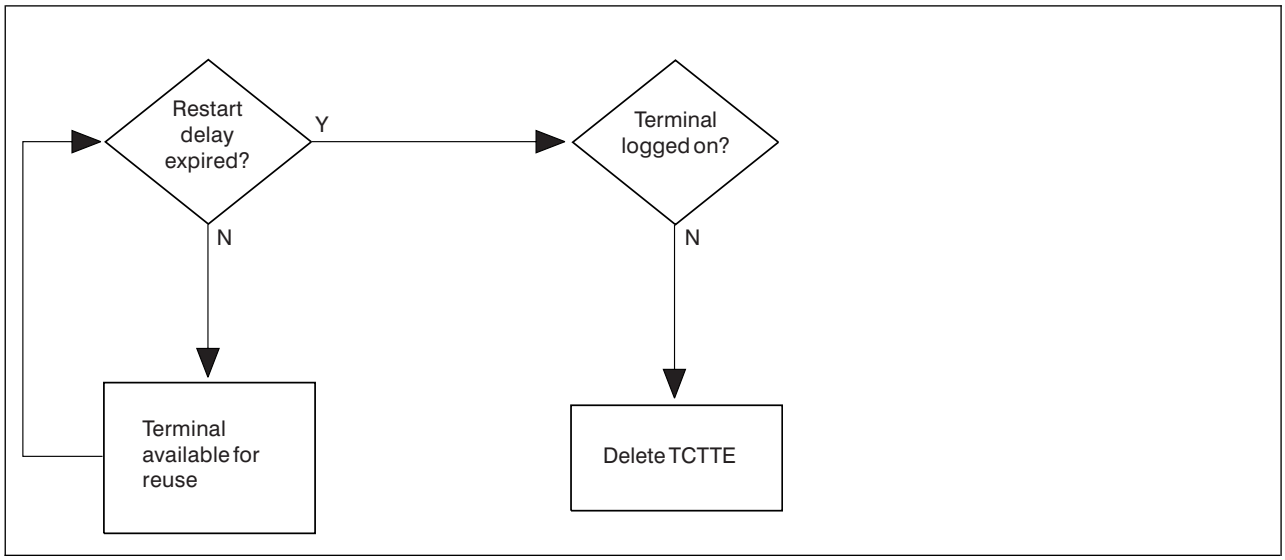


Figure 54. AUTOINSTALL—automatic logoff and TCTTE deletion after an emergency restart.

During a warm or emergency start, CICS uses these steps to determine whether an autoinstalled terminal definition that has been recovered should be deleted:

1. Before the restart delay period, specified in the AIRDELAY attribute of the TYPETERM definition, expires, the terminal definition is available for use.
2. After the restart delay period expires, the terminal definition is deleted. If a user attempts to log on at the terminal, the definition is autoinstalled.

Note:

1. Successive CICS runs are assumed to use the same restart delay period. If a run with restart delay > 0 is followed by an emergency restart with restart delay = 0, undeleted TCTTEs are recoverd. If the recovered TCTTE specifies AUTOCONNECT(YES), the associated terminal is logged on; otherwise the TCTTE is deleted after startup
2. Emergency restart: If the restart delay peiod = 0, there is no TCTTE recovery during emergeny restart.

Autoinstalling MVS consoles

Commands issued at an MVS console (or in a job stream) can be directed to a CICS region running as a started task, or job, using the MVS MODIFY command. Before CICS can accept the MVS command, it needs an entry in the terminal control table for the console issuing the command, which CICS terminal control can use to display a response.

This is how CICS handles commands (transaction invocations) it receives from an MVS operator console:

Pre-installed console definitions
When MVS receives your request, it identifies the CICS region from the task or job name, and passes your request to CICS.

CICS extracts the console's name from the MODIFY data, and searches the terminal control table (TCT) for a CONSNAME entry that matches the console name. If CICS finds a matching entry, it starts the transaction specified on the MODIFY command, and the transaction can send the results to the console using the termid of the console's entry in the terminal control table.

Autoinstalled console definitions

If CICS fails to find a matching entry, it checks the autoinstall status for consoles to determine whether it can perform an autoinstall for the console.

If autoinstall for consoles is active, CICS searches the autoinstall model table (AMT) and initiates the autoinstall process for the console. CICS either:

- Passes a list of autoinstall model definitions to the autoinstall control program, together with information about the console, or
- Automatically uses the first console model definition it finds, or a console model with the same name as the console, and autoinstalls the console using a CICS-generated termid, without calling your autoinstall control program.

Which of these options CICS takes is determined by the autoinstall status for consoles. The autoinstall status for consoles is either set at startup by the AICONS system initialization parameter, or dynamically by a CEMT (or EXEC CICS) SET AUTOINSTALL CONSOLES command.

The terminal autoinstall control program

You use the same autoinstall control program for console autoinstall as for z/OS Communications Server terminals and APPC connections, specifying the name of the control program on the AIEXIT system initialization parameter.

If the autoinstall control program is invoked (either the CICS-supplied program or your own) it selects one of the models and provides the rest of the information necessary to complete a TCT terminal entry for the console. When the autoinstall control program returns control, CICS builds a terminal control table terminal entry (TCTTE) for the console using the autoinstall model, the termid, and other data returned by the autoinstall control program, and MVS console data. CICS then adds the new entry to the TCT and starts the transaction specified on the MODIFY command.

Preset security for autoinstalled consoles

If the model terminal specifies USERID(*FIRST) or USERID(*EVERY), CICS uses the user ID passed by MVS on the MODIFY command to sign on the console, in effect using the MVS-passed user ID as the preset userid for the new console.

Automatic deletion of autoinstalled consoles

CICS automatically deletes autoinstalled consoles if they are unused for a specified delay period (the default is 60 minutes). As part of the installation function, the autoinstall control program can set a 'delete-delay' value for the console. The delete-delay period is the length of time (in minutes) that an autoinstalled console can remain installed without being used before CICS deletes it. Setting this value to 0 inhibits automatic deletion. Autoinstalled consoles are not recorded on the catalog and not recovered at restart. Note that a console is deleted even if there is a currently signed-on user.

Implementing autoinstall for MVS consoles

To implement autoinstall for MVS consoles, define appropriate resource definitions and configure system initialization parameters.

About this task

To autoinstall MVS consoles, you must ensure that the required model resource definitions are installed. The autoinstall models supplied in DFHLIST do not contain models suitable for console autoinstallation. The supplied group DFHTERMC contains an autoinstall model definition for a console, but it is not included in DFHLIST.

You can optionally write an autoinstall control program if you want to have additional control over the autoinstall process; for example, you might want to restrict the consoles that can be autoinstalled in a production environment. The autoinstall program must be able to handle console installation and delete requests. For information about writing an autoinstall program for consoles, see Writing a program to control autoinstall of consoles.

Procedure

1. Define a console terminal definition with the following attributes:
   - Specify AUTINSTMODEL(YES), or AUTINSTMODEL(ONLY)
   - Specify the CONSNAME(name)
   - Reference a TYPETERM that specifies DEVICE(CONSOLE)

   You can use the model console definition defined in the group DFHTERMC, which is added to your CSD by the **INITIALIZE** and **UPGRADE** commands.

   Note: When a model terminal definition is used by the console autoinstall function, CICS ignores the console name specified on the model definition.
2. Install the model console definition. You can either add its group to a group list and perform a cold start or install the resource by using CICS Explorer.
3. If you provide your own program to control the autoinstall of MVS consoles, set the AICONS system initialization parameter to YES. You can define this value in the SIT or override it dynamically using the **SET AUTOINSTALL CONSOLES(PROGAUTO)** command.
4. To use the autoinstall control program that is supplied with CICS, set the AICONS system initialization parameter to AUTO. You can define this value in the SIT or override it dynamically using the **SET AUTOINSTALL CONSOLES(FULLAUTO)** command. With the AUTO option, CICS allocates the TERMID automatically.
5. Ensure that the required autoinstall programs and transactions are installed. These resources are your autoinstall control program, the transactions CATA and CATD, and the programs DFHZATD and DFHZATA. You must have the following resources installed:
   - PROGRAM resource for your autoinstall control program
   - TRANSACTION resources for CATA and CATD
   - PROGRAM resources for DFHZATD and DFHZATA.

   The supplied autoinstall control program, DFHZATDX or DFHZATDY, accepts a request from any console, provided an autoinstall model for a console is found in the AMT. Use the model definition supplied in group DFHTERMC, or alternatively create your own autoinstall console models (see "The autoinstall control program for MVS consoles" on page 256)

Defining model TERMINAL definitions for consoles

Whenever CICS receives a command from an unknown console, and autoinstall for consoles is active, CICS searches the AMT for autoinstall models that describe a console.

These models must specify a CONSNAME, and reference a TYPETERM definition that specifies DEVICE(CONSOLE). The console name in a model definition is a dummy value in the case of autoinstall, and is ignored by CICS, which passes a list of AUTINSTNAME attribute values to the autoinstall control program, so that it can choose one of them.

Specifying automatic preset security

One attribute that your autoinstall control program may require in a model definition is a USERID that provides the correct preset security.

Selecting a model that has preset security defined ensures that the operator's security authority is predetermined. This avoids an operator having to logon to CICS, using the CESN signon transaction, in order to issue a MODIFY command. Two special operands of the USERID parameter provide for an automatic signon of consoles:

- USERID(*EVERY) means that CICS is to use the userid passed on the MVS MODIFY command every time a MODIFY command is received. The console is signed on using the MVS userid as the preset userid for the console being autoinstalled. The console remains signed on with this userid until the console is deleted or another MODIFY command is received with another userid. If a MODIFY command is received without a userid, CICS signs on the default CICS userid until a MODIFY command is received that has a valid userid. For non-console terminals, or if security is not enabled, this value is ignored.
- USERID(*FIRST) means that CICS is to use the userid passed on the first MVS MODIFY command that requires the console to be autoinstalled. The console is signed on with the MVS userid as the preset userid. The console remains signed on with this userid until the console is deleted. If a MODIFY command is received without a userid, CICS signs on the default CICS userid. For non-console terminals, or if security is not enabled, this value is ignored.

The autoinstall control program for MVS consoles

The console name and other MVS data about the console are not sufficient to build a terminal control table terminal entry (TCTTE) for the console. The autoinstall control program must create a CICS terminal identifier (termid) for the console, and choose a suitable model from the list of models passed by CICS in the communications area.

IBM supplies two assembler versions of an autoinstall control program (DFHZATDX and DFHZATDY in SDFHLOAD) that perform the basic functions, but it these may not perform all the functions that you require. For example, you may have your own conventions for terminal names and their relationship to console names. Terminal names are up to four characters long, and console names are up to eight characters long, hence it is often not possible to derive one from the other.

In addition to providing the termid, you can code your program to perform other functions associated with console definition. For example, your program could:

- Perform security checks
- Monitor the number of autoinstalled consoles currently logged on.

The autoinstall control program runs in a transaction environment as a user-replaceable program, rather than as a CICS exit. This means that you can read files, and issue other CICS commands, to help you determine the terminal name. The TCTTE entry for the console, however, does not exist at either of the times that the program is invoked, because (1) for the installation function it has not yet been created, and (2) for the delete function it has already been deleted. The program therefore runs in transaction-without-terminal mode.

You can write an autoinstall control program in the following languages: assembler language, C, COBOL, or PL/I. The CICS-supplied autoinstall program source is available in all four languages. The assembler version, which is used by default, is also shipped in executable form in SDFHLOAD. If you decide to write your own program, you can use one of the IBM-supplied programs as a pattern.

You specify the name of your autoinstall control program on the AIEXIT system initialization parameter. CICS supports only one autoinstall control program at a time, and therefore your program must handle console and terminal autoinstall requests if support for both is required.

When you test your autoinstall control program, you will find that CICS writes install and delete messages to the transient data destination, CADL, each time CICS installs and deletes a TCT entry. If there are no autoinstall models installed for consoles, CICS does not call the autoinstall control program, and fails the autoinstall request.

For information on implementing the CICS-supplied autoinstall control program, or designing and writing your own program, see Writing a program to control autoinstall of consoles

Autoinstalling APPC connections

When you decide whether to use autoinstall for connections, there are several factors to consider.

Security

Before using autoinstall for connections, consider whether the security considerations are suitable for your purposes.

The autoinstalled connection inherits the security attributes specified in the model. The security attributes from the CONNECTION definition are:

- SECURITYNAME
- ATTACHSEC
- BINDSECURITY

If you are using compatibility mode to share a CSD file with an earlier release of CICS, the BINDPASSWORD attribute is also inherited. See SESSIONS attributes for information on these attributes.

From the SESSIONS definition, the autoinstalled connection inherits the preset security attribute USERID. If you are attempting to autoinstall an attachsec verify connection from a non-EBCDIC based system, then you should refer to Bind-time security with LU6.2

Model terminal support (MTS)

MTS is not supported for connection autoinstall. This means that you must code the routines yourself to select the model definition name; you cannot get z/OS Communications Server to do it for you. MTS is described in “Autoinstall and z/OS Communications Server” on page 244.

Deletion of autoinstalled connections

Unlike autoinstalled terminal definitions, autoinstalled connection definitions are not deleted when they are no longer in use. This means that they continue to occupy storage.

The rules for cataloging and deletion of autoinstalled APPC connections have changed in CICS Transaction Server for z/OS. All autoinstalled connections are now cataloged, depending on the **AIRDELAY** system initialization parameter. If AIRDELAY=0, synclevel 1 connections are not cataloged.

The rules for deletion are:

- Autoinstalled synclevel 1 connections are deleted when they are released.
- If CICS is not registered as a generic resource, autoinstalled synclevel 2 connections are not deleted.
- If CICS is registered as a generic resource, autoinstalled synclevel 2 and limited resources connections are deleted when the affinity is ended.

If autoinstall is enabled, and an APPC BIND request is received for an APPC service manager (SNASVCMG) session that does not have a matching CICS CONNECTION definition, or a BIND is received for a single session, a new connection is created and installed automatically.

Implementing APPC connection autoinstall

You are most likely to benefit from autoinstall for connections if you have large numbers of workstations all with the same characteristics.

Procedure

1. Decide whether to use autoinstall for connections

   The main **benefits** of using autoinstall for connections are that cold, warm, and emergency restarts are faster, you have fewer CSD file definitions to manage, and less storage is taken up by unused definitions.

   However, some possible restrictions are discussed in "Autoinstalling APPC connections" on page 257 and "Recovery and restart for connection autoinstall" on page 260.
2. Decide which sessions to autoinstall

   You can use autoinstall for CICS-to-CICS connections, but it is intended primarily for workstations.
3. Create your model connection definitions

   A model definition provides CICS with one definition that can be used for all connections with the same properties. See "Model definitions for connection autoinstall" on page 259.
4. Design and write an autoinstall control program

   The autoinstall control program provides CICS with the extra information it needs to complete an autoinstall request, such as the autoinstall model name.

   For programming information about the autoinstall control program, see Writing a program to control autoinstall of LUs.
5. Enable autoinstall for connections

   Autoinstall for connections is enabled when you enable autoinstall for terminals; see "Implementing z/OS Communications Server autoinstall" on page 247 for information about the system initialization parameters and the CEMT and EXEC CICS INQUIRE and SET options used.

If terminal autoinstall has been enabled but you want to prevent autoinstall for connections, set the model connection out of service by using the **SET CONNECTION**(*connection-name)* **OUTSERVICE** command. For details, see and SET CONNECTION. When the model connection is out of service, the autoinstall control program cannot access it and copy it, so the autoinstall function for connections is effectively disabled.

Model definitions for connection autoinstall

Model definitions for connection autoinstall are different from those for terminal autoinstall in that they do not have to be defined explicitly as models. Any installed connection definition can be used as a "template" for an autoinstalled connection.

For performance reasons, use an installed connection definition that is not otherwise in use. The definition is locked while CICS copies it and, if you have a very large number of sessions autoinstalling, the delay may be noticeable.

You can set the model connection definition out of service by using the CEMT or EXEC CICS SET CONNECTION OUTSERVICE command. This effectively disables autoinstall for connections, because the autoinstall control program cannot access and copy the model definition.

For CICS-to-CICS connection autoinstall, the MAXIMUM attribute in the SESSIONS definition of the model connection should be large enough to accommodate the largest device using the model.

When creating model connections, you must ensure that the usergroup modenames specified in the session's MODENAME field match the usergroup modenames in the connection to be autoinstalled.

The autoinstall control program for connections

The autoinstall control program is invoked at installation for APPC single- and parallel-session connections initiated by a BIND. The autoinstall control program provides CICS with any extra information it needs to complete an autoinstall request. For APPC parallel sessions, the control program provides a SYSID for the new definition.

When an APPC BIND request is received by CICS, CICS receives the partner's z/OS Communications Server NETNAME and passes it to the autoinstall control program. The control program uses the information contained in the partner's NETNAME and in the z/OS Communications Server BIND to select the most appropriate model on which to base a new connection. In order to return the name of the most suitable model to CICS, the control program must know the NETNAME or SYSID of every model.

CICS supplies a sample control program, DFHZATDY, for connection autoinstall. You can use DFHZATDY unchanged if both of the following conditions are met:

- Your model connections are called CCPS, CBPS, or CBSS
- You use the last four characters of the NETNAME as the SYSID or terminal name

If these conditions are not met, you have to change DFHZATDY to suit your installation. Its source is supplied in CICSTS54.SDFHSAMP.DFHZATDY is defined as follows:

```
DEFINE PROGRAM(DFHZATDY)  GROUP(DFHAI62)  LANGUAGE(ASSEMBLER)
       RELOAD(NO)  RESIDENT(NO)  STATUS(ENABLED)  CEDF(NO)
       DATALOCATION(ANY)  EXECKEY(CICS)
```

The definitions for the supplied model connections and sessions are:

```
DEFINE CONNECTION(CBPS)  GROUP(DFHAI62)  NETNAME(TMPLATE1)
       ACCESSMETHOD(VTAM)  PROTOCOL(APPC)  SINGLESESS(NO)
DEFINE SESSION(CBPS)  GROUP(DFHAI62)  CONNECTION(CBPS)
       MODENAME(LU62PS)  PROTOCOL(APPC)  MAXIMUM(10,5)

DEFINE CONNECTION(CBSS)  GROUP(DFHAI62)  NETNAME(TMPLATE2)
       ACCESSMETHOD(VTAM)  PROTOCOL(APPC)  SINGLESESS(YES)
DEFINE SESSION(CBSS)  GROUP(DFHAI62)  CONNECTION(CBSS)
       MODENAME(LU62SS)  PROTOCOL(APPC)  MAXIMUM(1,0)

DEFINE CONNECTION(CCPS)  GROUP(DFHAI62)  NETNAME(TMPLATE3)
       ACCESSMETHOD(VTAM)  PROTOCOL(APPC)  SINGLESESS(NO)
DEFINE SESSION(CCPS)  GROUP(DFHAI62)  CONNECTION(CCPS)
       MODENAME(LU62PS)  PROTOCOL(APPC)  MAXIMUM(10,5)
```

If you want to use these definitions, you must add group DFHAI62 to your group list. Do not try to use the terminal autoinstall exit DFHZATDX to autoinstall connections; any sessions installed by DFHZATDX are terminated and message DFHZC6921 is issued.

Note: VTAM is now the z/OS Communications Server.

For programming information on customizing the autoinstall control program, see Writing a program to control autoinstall of LUs.

Recovery and restart for connection autoinstall

Information about autoinstalled connections are recovered in some, but not all, restart situations.

Because autoinstalled connections are not cataloged, they cannot benefit from persistent sessions support. This means that when a session does persist, any autoinstalled connections relating to it are unbound.

Autoinstalled connections are recovered only at a cold start of CICS; they are not recovered at warm and emergency restarts.

Unit-of-recovery descriptors (URDs) for autoinstalled connections are recovered after cold, warm, and emergency restarts, as long as the SYSIDs of the connections are consistent.

The SYSIDs of the autoinstalled connections must be consistent if you are using recoverable resources.

Autoinstalling IPIC connections

Unlike autoinstalling other resources, autoinstalling an IPCONN does not require a model definition (although this is recommended).

Unlike the model definitions used to autoinstall terminals, the definitions used to autoinstall IPCONNs do not need to be defined explicitly as models. Instead, CICS can use any previously-installed IPCONN definition as a template for a new definition.

For more information on autoinstalling IPCONNs, see Autoinstalling IPIC connections; preliminary considerations.

Autoinstalling programs, map sets, and partition sets

If autoinstall for programs is active, and an implicit or explicit load request is issued for a previously undefined program, map set, or partition set, CICS dynamically creates a definition, and installs it and catalogs it as appropriate.

An implicit or explicit load occurs in the following situations:

- CICS starts a transaction.
- An application program issues one of the following commands:
  EXEC CICS LINK (without a SYSID)
  EXEC CICS XCTL
  EXEC CICS LOAD
  EXEC CICS ENABLE (for global user exit, or task-related user exit, program)
- After an EXEC CICS HANDLE ABEND PROGRAM(...), a condition is raised and CICS transfers control to the named program.
- CICS calls a user-replaceable program for the first time.
- An application program issues one of the following commands:
  EXEC CICS RECEIVE or SEND MAP
  EXEC CICS SEND PARTNSET
  EXEC CICS RECEIVE PARTN

Applications deployed on platforms can make an implicit or explicit load request that causes a program, map set, or partition set to be auto-installed. A program that is auto-installed by a task for an application that is deployed on a platform is private to that version of the application. Private resources are not available to any other application or version installed on the platform, and they are not available to other tasks in the CICS region. For more information about private resources for applications, see Private resources for application versions.

Implementing program autoinstall

The only program that cannot be autoinstalled is the program autoinstall control program.

Procedure

1. Decide whether your programs are eligible for autoinstall

   All other programs can be autoinstalled, including:
   - All other user-replaceable programs
   - Global user exits (GLUEs) and task-related user exits (TRUEs)
   - PLT programs

   User-replaceable programs and PLT programs are autoinstalled on first reference. GLUEs and TRUEs are autoinstalled when enabled.
2. Decide whether to use autoinstall for programs

   Using autoinstall for programs can save time spent on defining individual programs, mapsets, and partitionsets. Savings can also be made on storage, because the definitions of autoinstalled resources do not occupy space until they are referenced.

   Use of autoinstall for programs can reduce the number of definitions to be installed on a COLD start, thereby reducing the time taken.
3. Decide which programs to autoinstall

Depending on your programs, you can choose to use a mixture of RDO and autoinstall. A suggested way to manage program autoinstall is to continue to use RDO for existing program definitions and for installing groups containing related programs. Use autoinstall as you develop and install new applications, and in test environments, where you might otherwise install large numbers of programs at CICS startup.

4. Enable autoinstall for programs

   You can enable autoinstall for programs either by specifying system initialization parameters by using the **SET SYSTEM** command.

   Three system initialization parameters relate to program autoinstall:

   PGAICTLG
   The PGAICTLG parameter specifies whether autoinstalled program definitions are cataloged. See "Cataloging for program autoinstall."

   PGAIPGM
   The PGAIPGM parameter specifies whether the program autoinstall function is active or inactive.

   PGAIEXIT
   The PGAIEXIT parameter specifies the name of the program autoinstall exit.

   Three options relate to program autoinstall on the **SET SYSTEM** command:

   PROGAUTOCTLG
   The PROGAUTOCTLG option specifies whether autoinstalled program definitions are to be cataloged. See "Cataloging for program autoinstall."

   PROGAUTOEXIT
   The PROGAUTOEXIT option specifies the name of the program autoinstall exit.

   PROGAUTOINST
   The PROGAUTOINST option specifies whether the program autoinstall function is active or inactive.

   For programming information on how to specify EXEC CICS commands, see Introduction to System programming commands in Developing system programs, and for information on how to use CEMT, see CEMT - master terminal.

5. Create your model program definitions

   A model definition provides CICS with one definition that can be used for all programs with the same properties. See "Model definitions for program autoinstall" on page 263

6. Design and write an autoinstall control program

   The autoinstall control program provides CICS with the extra information it needs to complete an autoinstall request, such as the autoinstall model name. You can write your autoinstall program in any language supported by CICS, with full access to the CICS application programming interface. See "The autoinstall control program for programs" on page 263.

Cataloging for program autoinstall

You can specify whether an autoinstalled program definition is cataloged or not (that is, whether the definition is retained over a warm or emergency start) by using either the PGAICTLG system initialization parameter or the PROGAUTOCTLG option on the CEMT INQUIREvSET SYSTEM command.

The values you can specify are:

ALL
Autoinstalled program definitions are written to the global catalog at the time of the autoinstall, and following any subsequent modification.

MODIFY
Autoinstalled program definitions are cataloged only if the program definition is modified by a SET PROGRAM command subsequent to the autoinstall.

NONE
Autoinstalled program definitions are not cataloged. This gives a faster CICS restart (warm and emergency) compared with the MODIFY or ALL options, because CICS does not reinstall definitions from the global catalog. Definitions are autoinstalled on first reference.

The setting that you specify for cataloging of autoinstalled programs has no effect on programs that are autoinstalled by a task for an application that is deployed on a platform. These programs are never cataloged.

The effects of specifying cataloging for program autoinstall apply mainly to recovery and restart. See "Program autoinstall and recovery and restart" on page 264.

Model definitions for program autoinstall

Model definitions for program autoinstall are different from those for terminal autoinstall in that they do not have to be defined explicitly as models. Any installed program, mapset, or partitionset definition can be used as a "template" for program autoinstall.

If you do not want to use your own definitions, you can use the CICS-supplied model definitions. These are:
- DFHPGAPG for programs
- DFHPGAMP for mapsets
- DFHPGAPT for partitionsets

They are listed in Supplied resource definitions, groups, and lists.

Note: Although the DFHPGAPG definition does not specify a program language, the CICS autoinstall routine detects the program language from the program being loaded.

The autoinstall control program for programs

You specify the name of the control program you want to use in the PGAIEXIT system initialization parameter, or use the CEMT or EXEC CICS INQUIREvSET SYSTEM command.

For detailed programming information about the autoinstall control program for programs, see Writing a program to control autoinstall of LUs. This topic is a summary of that information.

When the autoinstall control program is invoked for program autoinstall

The autoinstall program is invoked in a number of different situations for programs, mapsets, and partitionsets.

For programs, the autoinstall control program is invoked when:
- Any of these commands references a previously undefined program:

  - EXEC CICS LINK
  - EXEC CICS XCTL
  - EXEC CICS LOAD
- The program is the first program in a transaction.
- An EXEC CICS ENABLE is issued for a GLUE or a TRUE.
- An abend occurs after an EXEC CICS HANDLE ABEND PROGRAM command is issued and CICS invokes the named program.
- CICS calls a user-replaceable program.

For mapsets, the autoinstall control program is invoked when an EXEC CICS SEND MAP or EXEC CICS RECEIVE MAP refers to a previously undefined mapset.

For partitionsets, the autoinstall control program is invoked when an EXEC CICS SEND PARTNSET or EXEC CICS RECEIVE PARTN command refers to a previously undefined partitionset.

Sample programs

CICS supplies a number of sample programs for the program autoinstall exit.

The following sample programs are supplied by CICS:
- DFHPGADX—assembler program for program autoinstall exit
- DFHPGAHX—C program for program autoinstall exit
- DFHPGALX—PL/I program for program autoinstall exit
- DFHPGAOX—COBOL definition for program autoinstall exit

The source for these programs is supplied in the CICSTS54.SDFHSAMP sample library, but only the assembler version is supplied in executable form, in the CICSTS54.SDFHLOAD load library.

Program autoinstall functions

The program autoinstall facility uses model definitions, together with a user-replaceable program, to create explicit definitions for programs, mapsets, and partitionsets that need to be autoinstalled.

CICS calls the user-replaceable program with a parameter list that gives the name of the appropriate model definition. On return from the program (depending on the return code), CICS creates a resource definition from information in the model and from parameters which the program returns.

Note: CICS does not call the user-replaceable program for any CICS programs, mapsets, or partitionsets (that is, any objects beginning with the letters DFH).

Program autoinstall and recovery and restart

There is a difference in performance between warm and emergency restarts using program autoinstall without cataloging, and warm and emergency restarts using autoinstall with cataloging.

If you are using autoinstall **with** cataloging, restart times are similar to those of restarting a CICS region that is not using program autoinstall. This is because, in both cases, resource definitions are reinstalled from the catalog during the restart. The definitions after the restart are those that existed before the system was terminated.

If you are using autoinstall **without** cataloging, CICS restart times are improved because CICS does not install definitions from the CICS global catalog. Instead, definitions are autoinstalled as required whenever programs, mapsets, and partitionsets are referenced following the restart.

Autoinstalling model terminal definitions

If you define a model terminal for autoinstall, you install it just as you would an ordinary resource definition; that is, by installing the group containing it.

However, terminal definitions specified as AUTINSTMODEL(ONLY) are stored only in the autoinstall model table (AMT) at this time; they do not result in a TCTTE on the active CICS system. These model terminal definitions are stored in the AMT until needed by CICS to create a definition for an actual terminal logging on through the z/OS Communications Server using autoinstall. The resulting terminal definition is "automatically installed" at this time.

This is what happens when you install a TERMINAL definition, either at system initialization or using INSTALL:

AUTINSTMODEL(NO)
A TCT entry is created for the terminal.

AUTINSTMODEL(YES)
A TCT entry is created for the terminal, and the model definition is stored for later use by the autoinstall process.

AUTINSTMODEL(ONLY)
The model definition is stored for later use by the autoinstall process.

If you install two model TERMINAL definitions with the same AUTINSTNAME, the second one replaces the first.

For further information about autoinstall and model TERMINAL definitions, see "Autoinstalling z/OS Communications Server terminals" on page 242.

Autoinstalling journals

CICS writes journal data to journals or logs on log streams managed by the MVS system logger, or to SMF.

You must define JOURNALMODELs so that the connection between the journal name, or identifier, and the log stream, or SMF log, can be made. You can use RDO, DFHCSDUP, or an EXEC CICS CREATE JOURNALMODEL command to define a journalmodel.

CICS resolves the link between the log stream and the journal request by using user-defined JOURNALMODELs or, in the situation where an appropriate JOURNALMODEL does not exist, by using default names created by resolving symbolic names. See JOURNALMODEL attributes for more information about this.

Journals are not user-definable resources.

Macro resource definition

Reference information for resource definition macros used to create CICS tables.

Introduction to CICS control tables and macros

You can define most CICS resources by using resource definition online (RDO), but for some resources you must use CICS macros.

You use macros to define:
- Non-SNA networks
- Non-SNA LUs
- Non-VSAM files
- Monitoring resources
- System recovery resources

You must use *resource definition online (RDO)* for VSAM files, and to define programs, map sets, partition sets, queues, transactions, and profiles. You must also use RDO to define SNA LUs, and links and sessions with MRO (multiregion operation) and ISC (intersystem communication) systems. RDO is described in CICS resources.

CICS is configured under your control during system initialization. You select a system initialization table (SIT) and, through it, CICS selects other control tables. Each control table is created separately and can be re-created at any time before system initialization. You prepare the required control tables by coding the appropriate macros. For each table, the macros automatically generate the necessary linkage editor control statements.

You might need to read about the following areas related to control tables:
- The *system initialization table (SIT)*, required for the system to be operational. Other tables are required only if you are using the corresponding CICS facilities. For details of the SIT, see Specifying CICS system initialization parameters.
- The *job control language (JCL)* required for the control tables and how to link-edit and assemble the macro statements that you specify. See "Defining resources in CICS control tables" on page 270.
- Whether CICS loads a table above or below 16 MB (also known as above the line or below the line). The CICS table macros contain linkage editor control statements that determines this. Table 16 shows whether specific tables are loaded above or below 16 MB.

The control tables that can be defined by macros are shown in Table 16.

Table 16. Control tables that can be defined by macros. The third column shows whether the table is loaded above or below the 16 MB line.

Control table	Contents	Above the line?	Reference
Command list table (CLT)	Sets of commands and messages for an XRF takeover. The command list table (CLT) is used for XRF (extended recovery facility). If you are using XRF, you must have a CLT; it is used only by the alternate CICS system. The CLT contains a list of commands that are passed to JES or MVS for execution. It also provides the authorization for canceling the active CICS system.	Yes	Command list table (CLT)

Table 16. Control tables that can be defined by macros (continued). The third column shows whether the table is loaded above or below the 16 MB line.

Control table	Contents	Above the line?	Reference
Data conversion table	A data conversion table might be needed if the CICS system is using ISC to communicate with a member of the CICS family that runs on a hardware platform that does not use EBCDIC. The conversion table defines how data is to be changed from ASCII format at the workstation to EBCDIC format at the CICS host.		The DFHCNV macros used to create the table are described in Defining the conversion table.
DL/I directories (PDIR)	Databases and program specification blocks. If you use CICS-IMS DBCTL (database control) exclusively to manage your CICS system's use of DL/I, you need not define the DL/I directory (PDIR) using CICS. The PDIR is a directory of all the remote program specification blocks (PSBs) that are accessed by the CICS system. If you function-ship requests to a remote database manager (remote DL/I), you need only one directory, the PDIR.	No	PDIR—DL/I directory
File control table (FCT)	BDAM file definitions. The file control table (FCT) is retained to allow you to define BDAM files.	No	File control table (FCT)
Monitoring control table (MCT)	Monitoring actions (data collection) to be taken at each user event monitoring point (EMP). Different actions can be specified for each monitoring class at each EMP.	Yes	Monitoring control table (MCT)
Program list table (PLT)	A list of related programs. You may want to generate several PLTs to specify a list of programs that are to be executed in the initialization programs phase of CICS startup; executed during the first or second quiesce stages of controlled shutdown; or both, or enabled or disabled as a group by a CEMT ENABLE or DISABLE command.	Yes	Program list table (PLT)
Recoverable service table (RST)	Sets of recoverable service elements. The recoverable service table (RST) is used for IBM CICS IMS DBCTL (database control) support. If you are using XRF and DBCTL, you must have an RST: it is used by the active CICS system. The RST contains a list of recoverable service elements that define the DBCTL configuration. It defines which DBCTL CICS connects to.	Yes	Recoverable service table (RST)
System initialization table (SIT)	Parameters used by the system initialization process. In particular, the SIT identifies (by suffix characters) the versions of CICS system control programs and CICS tables that you have specified are to be loaded.		See Specifying CICS system initialization parameters
System recovery table (SRT)	A list of codes for abends that CICS intercepts.		System recovery table (SRT).

Table 16. Control tables that can be defined by macros (continued). The third column shows whether the table is loaded above or below the 16 MB line.

Control table	Contents	Above the line?	Reference
Temporary storage table (TST)	Generic names (or prefixes) for temporary storage queues. CICS still supports the use of a TST in combination with or in place of TSMODEL resource definitions. You must use a TST if you have application programs that reference temporary storage data sharing queues by specifying an explicit SYSID on EXEC CICS temporary storage commands, or if a SYSID is added by an XTSEREQ global user exit program. You must also use a TST if you require the TSAGE attribute. For temporary storage queues where you do not require these functions, you can use TSMODEL resource definitions, which provide all other functions of the TST and some additional functions.	Yes	Temporary storage table (TST)
Terminal control table (TCT)	Retained to define non- SNA LU networks.	No	Terminal control table (TCT)
Terminal list table (TLT)	Sets of related terminals. The terminal list table (TLT) allows terminal or operator identifications, or both, to be grouped logically. A TLT is required by the supervisory terminal operation (CEST), to define and limit the effective range of the operation. It can also be used by a supervisory or master terminal operation (CEMT) to apply a function to a predetermined group of terminals. A TLT can be used, singly or in combination with other TLTs, to provide predefined destinations for message switching.	No	Terminal list table (TLT)
Transaction list table (XLT)	Sets of logically related transaction identifications. A list of identifications that can be initiated from terminals during the first quiesce stage of system termination, or a group of identifications that can be disabled or enabled through the master terminal.	Yes	Transaction list table (XLT)

Format of macros

The CICS macros are written in assembler language and, like all assembler language instructions, must be written in a standard format.

Name	Operation	Operand	Comments
blank or symbol	DFHxxxxx	One or more operands separated by commas	

TYPE=INITIAL (control section):

Most of the tables must start with a TYPE=INITIAL macro.

For some tables you can provide information that applies to the whole table, on the TYPE=INITIAL macro.

The TYPE=INITIAL macro establishes the control section (CSECT) for the CICS system table, and produces the necessary linkage editor control statements. CICS automatically generates the address of the entry point of each table through the DFHVM macro that is generated from each TYPE=INITIAL macro. The entry point label of the table is DFHxxxBA. Only the END statement need be specified.

Naming and suffixing the tables:

You can have more than one version of a table. Each table has a name consisting of six fixed characters followed by a two character suffix.

The tables are named as follows:

Table 17. Names of the control tables

Table	Name
Command list table	DFHCLT*xx*
File control table	DFHFCT*xx*
Monitoring control table	DFHMCT*xx*
Program list table	DFHPLT*xx*
Recoverable service table	DFHRST*xx*
Resource control table	DFHRCT*xx*
System recovery table	DFHSRT*xx*
Terminal control table	DFHTCT*xx*
Terminal list table	DFHTLT*xx*
Temporary storage table	DFHTST*xx*
Transaction list table	DFHXLT*xx*

The first six characters of the name of each table are fixed. You can specify the last two characters of the name, using the SUFFIX operand. The SUFFIX operand is specified on the TYPE=INITIAL macro for each table.

Suffixes allow you to have more than one version of a table. A suffix may consist of one or two characters. The acceptable characters are: A-Z 0-9 @. (Do not use **NO** or **DY**.) Select the version of the table to be loaded into the system during system initialization, by specifying the suffix in the appropriate system initialization parameter operand.

For example:

```
DFHSIT...,FCT=MY,...
```

Note: The TYPE=INITIAL macros have a STARTER operand that is not listed in the descriptions of the individual macros in the main body of this book. Coding STARTER=YES enables you to use the $ and # characters in your table suffixes. The default is STARTER=NO. This operand should be used only with starter system modules.

TYPE=FINAL (end of table):

Most of the tables, again with the single exception of the SIT, must end with a TYPE=FINAL macro.

The TYPE=FINAL macro creates a dummy entry to signal the end of the table. It must be the last statement before the assembler END statement. The format is always like this:

```
         DFHxxT   TYPE=FINAL
```

Defining resources in CICS control tables

Some CICS resource are defined in CICS control tables.

The tables and their resource definitions are created by macros. You must use macros to define non-z/OS Communications Server networks and terminals, non-VSAM files, databases, and resources for monitoring and system recovery. You must use RDO for VSAM files, programs, map sets, partition sets, queues, transactions, and profiles.

For each of the CICS tables listed in Table 16 on page 266, complete the following steps:

1. Code the resource definitions you require.
2. Assemble and link-edit these definitions, using the CICS-supplied procedure DFHAUPLE, to create a load module in the required CICS load library. The load library is either CICSTS54.SDFHLOAD or CICSTS54.SDFHAUTH, which you must specify by the NAME parameter of the DFHAUPLE procedure. The CICS-supplied macros used to create the CICS tables determine whether tables are loaded above the 16MB line. All tables, other than the 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:
   - CLT: specifying CLT=NO causes CICS to try and load DFHCLTNO. The CLT is used only in the alternate CICS, when you are running CICS with XRF, and is always required in that case.
   - 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.
   - TCT: specifying TCT=NO causes CICS to load a dummy TCT, DFHTCTDY.
   - TLT: terminal list tables are specified by program entries in the CSD, and do not have a system initialization parameter.
   - MCT: specifying MCT=NO causes the CICS monitoring domain to dynamically build 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 could run your active and alternate CICS regions from separate sets of load libraries—in which case, you should make the separate copies **after** generating your control tables.

Table 18 lists all the CICS tables that can be assembled, link-edited, and installed in your CICS libraries.

Table 18. CICS tables that you can assemble, link-edit, and install in CICS libraries

Table	Module Name	Abbreviation	Required load library
Command list table	DFHCLTxx	CLT	SDFHAUTH
Data conversion table	DFHCNV	CNV	SDFHLOAD
File control table	DFHFCTxx	FCT	SDFHLOAD
Monitor control table	DFHMCTxx	MCT	SDFHLOAD
Program list table	DFHPLTxx	PLT	SDFHLOAD
DL/I program specification block	DFHPSBxx	PSB	SDFHLOAD
Recoverable service element table	DFHRSTxx	RST	SDFHAUTH
System initialization table	DFHSITxx	SIT	SDFHAUTH
System recovery table	DFHSRTxx	SRT	SDFHLOAD
Terminal control table	DFHTCTxx	TCT	SDFHLOAD
Terminal list table	DFHTLTxx	TLT	SDFHLOAD
Temporary storage table	DFHTSTxx	TST	SDFHLOAD
Transaction list table	DFHXLTxx	XLT	SDFHLOAD

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 19 in the CICSTS54.SDFHSAMP library:

Table 19. Sample CICS system tables in the CICSTS54.SDFHSAMP library

Table	Suffix	Notes®
Command list table (CLT)	1$	XRF regions only
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$	
System initialization table (SIT)	$$	Default system initialization parameters
System initialization table (SIT)	6$	
System recovery table (SRT)	1$	
Terminal control table (TCT)	5$	Non-z/OS Communications Server terminals only

Although you can modify and reassemble the 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 exceptional in that a new table can be brought into use without shutting down either the active CICS region or the alternate CICS region.)

Defining control tables to CICS

You can assemble and link-edit more than one version of a table, and (except for the CNV) use a suffix to distinguish them. To specify which version you want CICS to use, code a system initialization parameter of the form *tablename=xx* as a startup override.

About this task

Other tables that have special requirements are program list tables (PLTs), terminal list tables (TLTs), and transaction list tables (XLTs). For each TLT, autoinstall for programs must be active or you must specify a program resource definition in the CSD, using the table name (including the suffix) as the program name. PLTs or XLTs are autoinstalled if there is no program resource definition in the CSD. 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)
```

See "Naming and suffixing the tables" on page 269 for information about single and two-character suffixes.

For information about program and terminal list tables, see Program list table (PLT) and Terminal list table (TLT).

The DFHCNV conversion table is required when communicating with CICS on a non-z Systems platform. For information about the data conversion process, see Defining the conversion table.

Assembling and link-editing control tables: the DFHAUPLE procedure

To assemble and link-edit your tables, write a job that calls the CICS-supplied sample procedure DFHAUPLE.

About this task

The DFHAUPLE procedure needs the following libraries to be online:

Table 20. Library requirements for the DFHAUPLE procedure

Library name	Tables required for
SYS1.MACLIB	All
CICSTS54.SDFHMAC	All
CICSTS54.SDFHLOAD	All except the CLT, RST, and SIT
CICSTS54.SDFHAUTH	CLT, RST, and SIT
SMP/E global zone	All
SMP/E target zone	All

When you have assembled and link-edited your tables, define them to CICS by system initialization parameters.

Steps in the DFHAUPLE procedure:

The DFHAUPLE procedure is tailored to your CICS environment and stored in the CICSTS54.XDFHINST library when you run the DFHISTAR job.

About this task

The DFHAUPLE procedure contains the following steps:

1. ASSEM: This step puts your table definition macros into a temporary partitioned data set (PDS) member that is used as input to the ASM and SMP steps. The BLDMBR step subsequently adds further members to this temporary PDS.
2. ASM: In this assembly step, SYSPUNCH output goes to a temporary sequential data set. This output consists of IEBUPDTE control statements, lin-edit control statements, SMP control statements, and the object deck.
3. BLDMBR: In this step, the IEBUPDTE utility adds further members to the temporary PDS created in the ASSEM step. These members contain link-edit control statements and SMP control statements, and the object deck from the assembly step.
4. LNKEDT: The link-edit step uses the contents of the PDS member LNKCTL as control statements. The object code produced in step 2 comes from the temporary PDS. The output goes to the load library that is specified by the NAME parameter on the procedure. You must specify NAME=SDFHAUTH for the CLT, RST, and SIT, and NAME=SDFHLOAD for all the others.
5. ZNAME: This step creates a temporary data set that passes to the SMP/E JCLIN job step; it contains a SET BDY command that defines a target zone name. This tells SMP/E which target zone to update.
6. SMP: The SMP step uses the temporary PDS members MACROS, SMPCNTL, SMPJCL1, SMPJCL2, LNKCTL, SMPEOF, and the object deck to update the control data set (CDS).
7. DELTEMP: This final step deletes the temporary partitioned data set, &&TEMPPDS.

   This step must run successfully if you want SMP to reassemble CICS tables automatically when applying later maintenance.

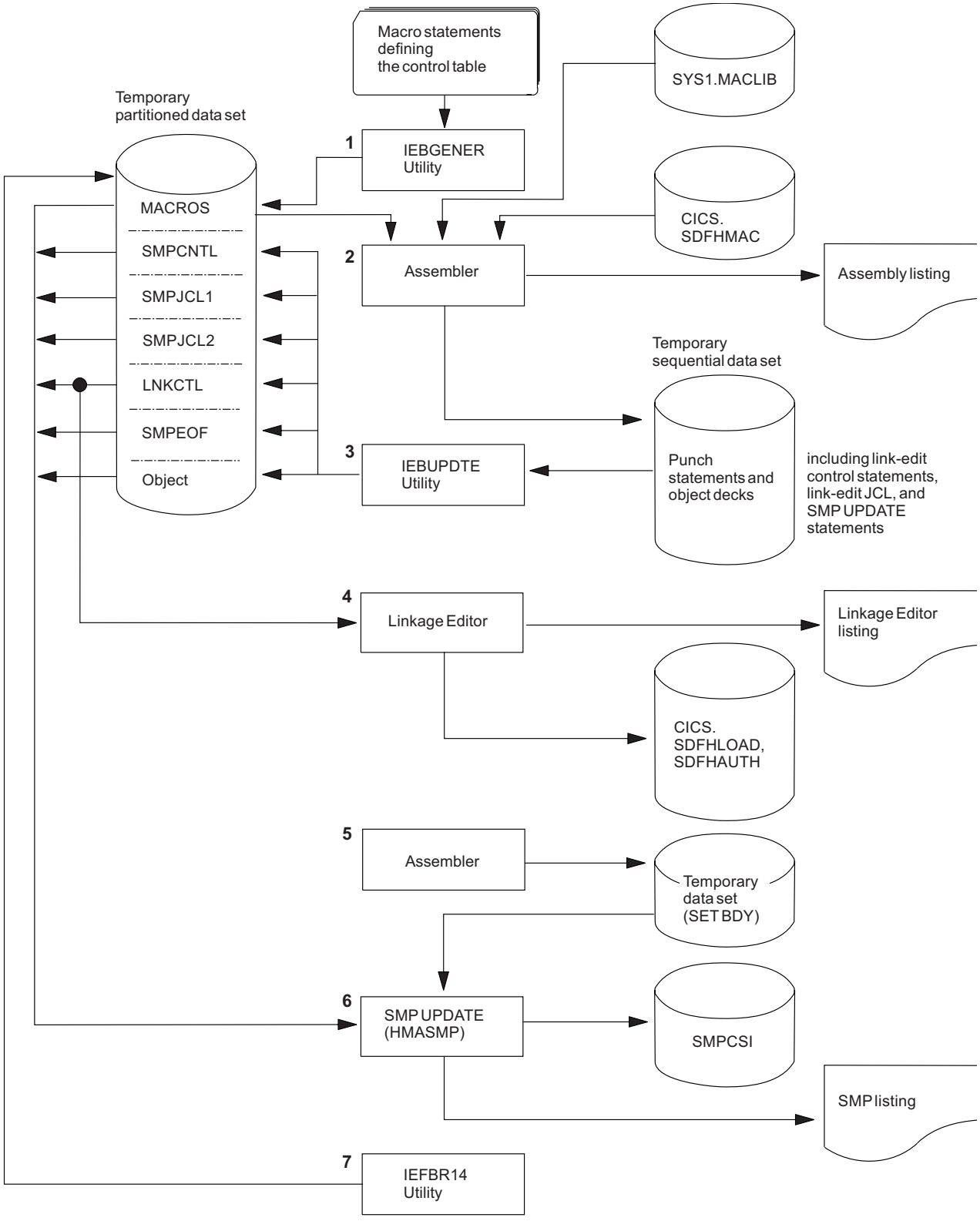


Figure 55. Assembling and link-editing the control tables using the DFHAUPLE procedure

Sample job stream for control tables:

You can use a single job stream to assemble and link-edit all of the CICS control tables except for the CLT and the RST

Use the following job stream:

```
//jobname      JOB     (accounting information),
//             CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
//ASMTAB       EXEC    PROC=DFHAUPLE[,NAME={SDFHLOAD|SDFHAUTH}]
//*
//ASSEM.SYSUT1 DD   *
         .
      Control table macro statements
         .
/*
```

Figure 56. Job stream for control tables

Specify NAME=SDFHAUTH on this job for the system initialization table only; link-edit all other tables (except the CLT and RST) into SDFHLOAD.

Defining CICS bundles

A CICS bundle is a directory that contains artifacts and a manifest that describes the bundle and its dependencies. CICS bundles provide a way of grouping and managing related resources. CICS bundles also provide versioning for the management of resource updates, and can declare dependencies on other resources outside the bundle. Application developers can use CICS bundles for application packaging and deployment, business events, and services. System programmers can use CICS bundles for system events and policies.

About this task

You create CICS bundles using the IBM CICS Explorer or IBM Developer for z System z®. The bundle specifies a set of resources that CICS creates dynamically when the bundle is deployed. The bundle can also specify any prerequisite system resources for the application. CICS does not dynamically create prerequisite system resources but can check that they exist in the CICS region.

Some characteristics of CICS resources change because they are defined in a CICS bundle and dynamically created as part of a bundle deployment. The life cycles of CICS resources that are created in this way cannot be managed independently of the BUNDLE resource. They must be managed through the CICS bundle as a unit. Because of bundle and resource lifecycle interdependency, application architects must carefully consider which set of resources for an application should have their lifecycle tied to the lifecycle of a CICS bundle. To understand more about the potential implications of defining resources in CICS bundles, see “Characteristics of resources in CICS bundles” on page 281.

You can deploy an individual CICS bundle in a CICS region, or you can use it as part of a packaged application deployed on a platform, or deploy it directly on a platform. If you are deploying the CICS bundle in an application or to a platform, you create the CICS bundle using the normal process, but then you package it as part of the application, or add it directly to the platform.

Applications deployed on platforms that use the CICS resources that are supported as private resources, in combination with other resources designed for applications and with imported resources, are eligible for multi-versioning. The following resources are supported as part of multi-versioned applications:

- PROGRAM resources defined in CICS bundles that are part of the application
- LIBRARY resources defined in CICS bundles that are part of the application
- PACKAGESET resources defined in CICS bundles that are part of the application
- Policies
- Statements of application entry points
- Any resource that is defined as a dependency, or import, for the application

A CICS bundle can be installed in CICS regions by more than one application if it contains only resources that are eligible for multi-versioning.

A CICS bundle that contains other resource types cannot be included in other applications that are installed in the same CICS regions in the platform, and you cannot add it to the platform if it has already been installed for an application. If your application requires a resource that is not supported for multi-versioning and has already been used in a CICS bundle that has been installed in the target region type in the platform, create a new CICS bundle project. Declare the resource as a dependency for the new CICS bundle, by specifying an import in the manifest file. Then include the new CICS bundle project in your application project.

Procedure

1. Use the IBM CICS Explorer to create a CICS bundle and define the artifacts that you want to include in the bundle. Follow the instructions in Working with bundles in the CICS Explorer product documentation.
2. If you are deploying the CICS bundle on its own, deploy it to a suitable directory on z/OS UNIX. CICS must have permission to read this directory. If you are deploying the CICS bundle as part of an application or a platform, follow the instructions in Working with platforms, applications, and policies in the CICS Explorer product documentation to include the CICS bundle.
3. If you are deploying the CICS bundle on its own, define, enable, and make available a BUNDLE resource for the CICS bundle. See BUNDLE resources for details of the attributes to specify.
4. If you are deploying the CICS bundle with a platform, follow the instructions in Working with bundles in the CICS Explorer product documentation to remove the previous version of the CICS bundle then add the new version of the CICS bundle to the platform.
5. If you are deploying the CICS bundle as part of an application, follow the instructions in Working with applications in the CICS Explorer product documentation to deploy the application.

Results

CICS reads the manifest in the bundle directory and dynamically creates the CICS resources. CICS also checks that any required dependencies, for example programs or files, outside the bundle are present in the CICS region.

What to do next

You can inquire on the installed BUNDLE resource and its associated resources using the CICS Explorer. The resource signature of every dynamically created resource indicates that it was created by a BUNDLE resource.

You cannot directly enable, disable, or discard resources that are created dynamically by a BUNDLE resource. When you enable, disable, or discard a CICS bundle, or make it available or unavailable, the actions are also applied to the dynamically created resources for the bundle. If the CICS bundle is part of an application in a platform, and you enable, disable, or discard the platform or application, or make the application available or unavailable, the actions are applied to the CICS bundle and the dynamically created resources for the bundle.

You can use the editor in the CICS Explorer to modify the artifacts that are specified in a CICS bundle, and the definitions for resources that CICS creates dynamically when the bundle is deployed. To apply your changes, deploy your modified CICS bundle on the directory on z/OS UNIX, make unavailable then disable and discard the BUNDLE resource, and then install the BUNDLE resource again and make it available.

Artifacts that can be deployed in bundles

The artifacts that you can define and deploy in CICS bundles include application and system events, Atom feeds, channel-based services, CICS policies, CICS programs, OSGi bundles, XML-based services, and transactions. Each of these artifacts is represented by one or more CICS resources and these resources are dynamically created as part of the bundle deployment.

The resource signature for dynamically created resources indicates that they were created during the bundle deployment, and contains the name of the BUNDLE resource.

When you define and deploy CICS resources in a CICS bundle, the management and lifecycle of the resources is delegated to the CICS bundles that dynamically created them. The dynamically created resources are in effect part of the bundle or the application, rather than existing independently.

- You can inquire on the dynamically created resources, but you cannot enable, disable, or discard them directly.
- When you enable, disable, or discard the CICS bundle, the action is also applied to the dynamically created resources for the bundle.
- If the CICS bundle is part of an application on a platform, and you enable, disable, or discard the platform or application, the actions are applied to the CICS bundle and the dynamically created resources for the bundle.
- You use the editors in CICS Explorer to modify the artifacts specified in a CICS bundle, and the definitions for resources that are defined in the bundle.

For more information about the implications of managing resources in CICS bundles, see “Characteristics of resources in CICS bundles” on page 281.

When you create the definition for a LIBRARY, PACKAGESET, or PROGRAM resource in a CICS bundle, and deploy the CICS bundle as part of an application on a platform, the dynamically created LIBRARY, PACKAGESET, or PROGRAM resource is private to that version of that application. Private resources are not available to any other application or version installed on the platform, and they are

not available to other tasks in the CICS region. The names of private resources therefore do not have to be unique in your installation. Applications that use the CICS resources that are supported as private resources, together with other resources such as policies and application entry points, are eligible for multi-versioning. For more information about private resources, see "Private resources for application versions" on page 289.

The following artifacts can be deployed using a CICS bundle. The resource type is defined as a URI in the bundle manifest.

Application event or system event
For CICS event processing, you can use CICS Explorer to define and deploy event bindings, capture specifications, and EP adapters in CICS bundles. When you install a BUNDLE resource for a business event, CICS dynamically creates suitable EVENTBINDING, CAPTURESPEC, and EPADAPTER resources in the CICS region. For instructions to define business events using the CICS event binding editor in CICS Explorer, and deploy them in CICS bundles, see Creating an event binding.

URI: http://www.ibm.com/xmlns/prod/cics/bundle/EPADAPTER

http://www.ibm.com/xmlns/prod/cics/bundle/EPADAPTERSET

http://www.ibm.com/xmlns/prod/cics/bundle/EVENTBINDING

Atom feed
To serve an Atom feed from CICS, you can use CICS Explorer to create and deploy an Atom configuration file in a CICS bundle. When you install a BUNDLE resource for an Atom feed, CICS dynamically creates suitable ATOMSERVICE, XMLTRANSFORM, and URIMAP resources in the CICS region. For instructions to define Atom feeds using the Atom configuration wizard and the Atom configuration editor in CICS Explorer, and deploy them in CICS bundles, see Setting up CICS definitions for an Atom feed.

URI: http://www.ibm.com/xmlns/prod/cics/bundle/ATOMSERVICE

Channel-based service
Channel-based services are CICS applications that are described as components and assembled together using the Service Component Architecture (SCA) tooling in IBM Developer for z Systems. The SCA tooling deploys the composite application to CICS as a CICS bundle. These services are available only to other CICS applications that use the **INVOKE SERVICE** API command and pass binary data in containers on a channel. For instructions to use IBM Developer for z Systems to create a channel-based service, see Creating a channel-based service.

URI: http://www.ibm.com/xmlns/prod/cics/bundle/SCACOMPOSITE

File You can create a definition for a FILE resource in a CICS bundle. The following file types are supported for definition in CICS bundles:

- VSAM files (including files that refer to CICS-maintained, user-maintained, and coupling facility data tables, as well as files that refer to VSAM data sets)
- Remote VSAM files
- Remote BDAM files

When you install a BUNDLE resource that contains a file definition, CICS dynamically creates the FILE resource in the CICS region.

URI: http://www.ibm.com/xmlns/prod/cics/bundle/FILE

JSON transform

A JSON transform is used by an application with the linkable interface for transforming JSON. The JSON assistant uses a language structure or a JSON schema to generate the JSON binding, and also creates a bundle. When you install the BUNDLE resource, CICS dynamically creates an JSONTRANSFRM bundle part that defines where the JSON binding and schema are located.

URI: http://www.ibm.com/xmlns/prod/cics/bundle/JSONTRANSFRM

JVM server

You can create a definition for a JVMSERVER resource in a CICS bundle. The JVM profile for the JVM server is packaged in the CICS bundle along with the resource definition, so that the JVM profile is available in any CICS region where you install the CICS bundle. You can create the JVM profile using sample templates for an OSGi JVM server, an Axis2 JVM server, or a Liberty JVM server, or import an existing JVM profile to the CICS bundle from elsewhere in the workspace or from the local file system. You can customize the JVM profile to fit your system's requirements. When you install a BUNDLE resource that contains a JVM server definition, CICS dynamically creates the JVMSERVER resource in the CICS region.

URI: http://www.ibm.com/xmlns/prod/cics/bundle/JVMSERVER

Library

You can create a definition for a LIBRARY resource in a CICS bundle. When you install a BUNDLE resource that contains a definition for a dynamic program LIBRARY concatenation, CICS dynamically creates the LIBRARY resource in the CICS region.

URI: http://www.ibm.com/xmlns/prod/cics/bundle/LIBRARY

OSGi bundle

Java applications that are packaged as OSGi bundles can be deployed in CICS bundles to run in a JVM server. When you install a BUNDLE resource that contains OSGi bundles and services, CICS dynamically creates the OSGIBUNDLE and OSGISERVICE resources that represent the OSGi bundles and services in the OSGi framework. CICS uses the resources to manage the life cycle of the OSGi bundles and OSGi services. For instructions to use the CICS Explorer Software Development Kit (SDK) to create CICS bundles that contains OSGi bundles, see Developing applications using the CICS Explorer SDK.

URI: http://www.ibm.com/xmlns/prod/cics/bundle/OSGIBUNDLE

Package set

You can create a definition for a PACKAGESET resource in any CICS bundle that is deployed with a CICS platform, application, or application binding project. Use the PACKAGESET resource to define which DB2 collection to use to qualify any SQL requests made by application tasks running on a CICS platform.

When you install a BUNDLE resource that contains a PACKAGESET resource with a CICS application or application binding, it is private to the application. CICS defines that DB2 collection to be used by all the application tasks.

When you install a BUNDLE resource that contains a PACKAGSET resource with a CICS platform, CICS defines the DB2 collection that will be

used by all application tasks that run on the platform where no specific DB2 collection is defined for the CICS application.

URI: http://www.ibm.com/xmlns/prod/cics/bundle/PACKAGESET

Pipeline
You can create a definition for a PIPELINE resource in a CICS bundle. The pipeline configuration file for the pipeline is packaged in the CICS bundle along with the resource definition. You can create the pipeline configuration file using one of the CICS-supplied sample configuration files for service providers and service requesters, or import an existing configuration file from the local file system. When you install a BUNDLE resource that contains a pipeline definition, CICS dynamically creates the PIPELINE resource in the CICS region. PIPELINE resources that are defined in CICS bundles can only be used with WEBSERVICE resources that are defined in CICS bundles or created dynamically by a pipeline scan.

URI: http://www.ibm.com/xmlns/prod/cics/bundle/PIPELINE

Policy You can use CICS Explorer to define and deploy policies in a CICS bundle. Policies can be deployed to manage the behavior of user tasks within CICS regions. When you install a BUNDLE resource for a policy, CICS dynamically adds the policy rules to the set of rules used to manage user tasks in the CICS region.

URI: http://www.ibm.com/xmlns/prod/cics/bundle/POLICY

Program
You can create a definition for a PROGRAM resource in a CICS bundle for any high-level programming language. When you install a BUNDLE resource that contains a program definition, CICS dynamically creates the PROGRAM resource in the CICS region.

URI: http://www.ibm.com/xmlns/prod/cics/bundle/PROGRAM

TCP/IP service
You can create a definition for a TCPIPSERVICE resource in a CICS bundle. When you install a BUNDLE resource that contains a TCP/IP service definition, CICS dynamically creates the TCPIPSERVICE resource in the CICS region.

URI: http://www.ibm.com/xmlns/prod/cics/bundle/TCPIPSERVICE

Transaction
You can create a definition for a TRANSACTION resource in a CICS bundle. When you install a BUNDLE resource that contains a transaction definition, CICS dynamically creates the TRANSACTION resource in the CICS region.

URI: http://www.ibm.com/xmlns/prod/cics/bundle/TRANSACTION

URI map
You can create a definition for a URIMAP resource in a CICS bundle. When you install a BUNDLE resource that contains a URI map definition, CICS dynamically creates the URIMAP resource in the CICS region.

URI: http://www.ibm.com/xmlns/prod/cics/bundle/URIMAP

Web application
Web applications that are packaged as WAR files, an EBA file, or EAR files can be deployed in a CICS bundle to run in a Liberty JVM server. When you install a BUNDLE resource that contains a web application, CICS

dynamically creates suitable Liberty application definitions in the `installedApps.xml` file. The Liberty JVM server uses these definitions to dynamically create the web application, and the lifecycle of the web application can be controlled using the BUNDLE resource. For instructions to use the CICS Explorer SDK to create CICS bundles that contain web applications, see Deploying applications in a JVM server in Deploying.

URI: http://www.ibm.com/xmlns/prod/cics/bundle/WARBUNDLE

http://www.ibm.com/xmlns/prod/cics/bundle/EBABUNDLE

http://www.ibm.com/xmlns/prod/cics/bundle/EARBUNDLE

Web service

You can create a definition for a WEBSERVICE resource in a CICS bundle. When you install a BUNDLE resource that contains a web service definition, CICS dynamically creates the WEBSERVICE resource in the CICS region. Along with the WEBSERVICE resource definition, you can import a web service binding file and a WSDL document or WSDL archive file to be packaged in the bundle. You can also generate URIMAP resources and alias transactions to support the web service. For a web service provider, you can optionally include a PROGRAM resource definition in the bundle. The definition for the PROGRAM resource can be edited using the resource editor in the CICS Explorer.

URI: http://www.ibm.com/xmlns/prod/cics/bundle/WEBSERVICE

XML-based service

XML-based services are typically web service provider or requester applications that use XML to interface with other applications and use a binding to transform the data. XML-based services are available to CICS applications that use the **INVOKE SERVICE** API command or to business services that are on an external network. If you create a web service using the SCA tooling in IBM Developer for z Systems, you can deploy the web service as a CICS bundle. When you install a BUNDLE resource that contains a web service, CICS dynamically creates a number of resources, including URIMAP and WEBSERVICE resources. For instructions to use IBM Developer for z Systems to create an XML-based service, see Creating an XML-based service.

An XML-based service can also be an application that uses the **TRANSFORM** API commands to map application data to and from XML. The XML assistant uses a language structure or an XML schema to generate the XML binding, and can also create a bundle. When you install the BUNDLE resource, CICS dynamically creates an XMLTRANSFORM resource that defines where the XML binding and schema are located.

URI: http://www.ibm.com/xmlns/prod/cics/bundle/XMLTRANSFORM

You can extend the list of supported resource types by using the callback interface in the resource lifecycle manager domain. With this interface, vendors can create new user resource types and manage them in BUNDLE resources.

Characteristics of resources in CICS bundles

Some characteristics of CICS resources change because they are defined in a CICS bundle and dynamically created as part of a bundle deployment. When you design the architecture of an application, or select resources from an existing application to define in CICS bundles, be aware of these important considerations.

The full list of CICS resource types for which you can create resource definitions in CICS bundles is as follows:

- FILE
- JVMSERVER
- LIBRARY
- PIPELINE
- PROGRAM
- TCPIPSERVICE
- TRANSACTION
- URIMAP
- WEBSERVICE

For the list of all artifacts that you can define and deploy in CICS bundles, including OSGi bundles and events, see Artifacts that can be deployed in bundles in Administering.

Bundle and resource interdependency

When you use CICS bundles, the lifecycle of the installed resources follows the lifecycle of the bundle. When you make available or unavailable, enable or disable, or discard a CICS bundle, the same action is applied to all the resources that were dynamically created by the CICS bundle. In addition, if your CICS bundle is included in an application bundle, you manage the CICS bundle as part of the application.

When a CICS bundle dynamically creates and installs a resource into a CICS region, the state of the resource is managed through the CICS bundle by which the resource was installed. You enable or disable the dynamically created resource by performing that action on the CICS bundle. For resources in CICS bundles that are managed as part of an application, you also control users' access to the resources by declaring application entry points. PROGRAM, TRANSACTION, and URIMAP resources can be declared as application entry points, and can be set as available or unavailable to users. With application entry points, you can install the application including the CICS bundle and its resources, then enable them to verify the installation. When you choose to provide the service to users, you make the application containing the CICS bundle available using the CICS Explorer. This action makes the application entry point, and the resources in the CICS bundle, available to callers. An individual CICS bundle that contains application entry points can also be made available by using the **EXEC CICS SET BUNDLE** command.

In general, you do not directly modify the attributes of a resource that was dynamically created and installed by a CICS bundle. Instead, you install a new version of the CICS bundle with the required changes. However, some long-lived resources, such as the FILE, TCPIPSERVICE, and JVMSERVER resources, that have been installed by a CICS bundle, can be modified directly using CICS system commands, although the changes are not recovered across a warm restart of CICS.

An exception is also made for JVMSERVER and TCPIPSERVICE resources to the rule about changing the state of resources that are defined in CICS bundles. By default, these resources are not disabled until all the current tasks have finished using them. In a situation that requires immediate removal of these resources, you can use the SET JVMSERVER PURGE, FORCEPURGE, or KILL command, or the SET TCPIPSERVICE IMMCLOSE command, on the dynamically created resource in the CICS region, to force completion of the disable process when the resources are

still in use. You can only issue these commands against a dynamically created resource after you have attempted to disable the CICS bundle or the application with which it is deployed.

Because of bundle and resource lifecycle interdependency, application architects must consider which set of resources for an application should have their lifecycle tied to the lifecycle of a CICS bundle. In the following two scenarios, the organization of the bundled resources determines how the application is administered:

- You want to modify the URI map associated with a program, without having to disable the program. To enable modification, the URI map and program resources can be packaged into different CICS bundles.
- You want to reload a program that was installed in multiple CICS regions by a CICS bundle, without having to disable the program across an entire CICSplex. To reload, you can apply the program change individually to each CICS region, one region at a time. Work can be configured so that it bypasses the region that is currently being updated, while you are applying the changes.

If an application or a CICS bundle requires a CICS resource to be available in the CICS region, but you do not want the lifecycle of the resource to be tied to the lifecycle of the CICS bundle, you can add the resource as a dependency for the CICS bundle. Dependencies, or imports, are specified in <import> elements in the bundle manifest. You can specify the action that CICS takes at bundle installation time if one of the required imports is not present in the CICS region; for example, the CICS bundle can remain disabled, or be enabled but with a warning message. For more information about dependencies, see "Manifest contents for a CICS bundle" on page 295.

Installed resources that are defined in CICS bundles cannot be overwritten by resource definitions with duplicate names. When the bundle part containing a resource has been installed, if you try to install a duplicate resource using another CICS bundle, or using RDO, EXEC CICS CREATE commands or other methods, the request is rejected. You also cannot overwrite installed resources with a duplicate resource that is defined in a CICS bundle. When you attempt to install the CICS bundle, if a resource with the same name and resource type exists, the bundle part containing the resource is not installed, and the CICS bundle is placed in an unusable state. Resources that are defined in CICS bundles are therefore protected against accidental overwriting by duplicate resource names.

Private resources in bundles

When you create the definition for certain resource types in a CICS bundle, and deploy the CICS bundle as part of an application on a platform, the dynamically created resource is private to that version of that application. Private resources are not available to any other application or version installed on the platform, and they are not available to other tasks in the CICS region. Resources deployed in this way do not require unique names, and multiple versions of the resources can be deployed as part of application versions, without the need to disable the previous version of the resources. For more information about private resources, and the resource types that are supported, see Private resources for applications.

Resources to deploy with platform bundles

TCPIPSERVICE resources, JVM servers, and pipelines are resources that are likely to be required in multiple CICS regions in a platform, and to be used to run more

than one application. To enable convenient scaling of a platform, it is good practice to deploy the CICS bundles for TCPIPSERVICE, JVMSERVER, and PIPELINE resources as part of the platform bundle, rather than as part of an application. You can specify the CICS regions in the platform where the resource is to be installed. For new resources, you can add new CICS bundles to a running platform after the platform is installed.

Certain CICS resources, including PIPELINE and JVMSERVER resources, must be installed and enabled before you install other resources that depend on them. With these resources, you can use platform architecture to manage the dependency. Deploy the CICS bundles containing the definitions for the required resources as part of a platform bundle, and then deploy the CICS bundles containing the dependent resources as part of applications that are deployed on the platform. This architecture ensures that when the resources are first installed in a CICS region or if the CICS region is restarted, the required resources and the dependent resources are installed and enabled in the correct order.

Mixed case resource names

CICS bundles support mixed case names for resources where these are supported in CICS. However, your file system might not allow files with the same name but different cases to reside in the same CICS bundle project. If you require duplicate resource names that use different cases, define these resources in separate CICS bundle projects.

Escaped characters in resource names

When you save a CICS bundle in the CICS Explorer, certain characters in the resource names are escaped so that the resource can be used in different file systems. When you expand the CICS bundle in the CICS Explorer and view the resources defined in the CICS bundle, the resource names are displayed with the escape sequences. When you open the resources for editing, the resource names are displayed with the original unescaped characters.

The following characters in the names of resources are escaped when you save a CICS bundle in the CICS Explorer:

Table 21. Escape sequences in resource names in CICS bundles

Character	Escape sequence
. (period)	%2E
/ (forward slash)	%2F
& (ampersand)	%26
? (question mark)	%3F
: (colon)	%3A
| (vertical bar)	%7C
" (double quotation mark)	%22
; (semicolon)	%3B
< (less than)	%3C
> (greater than)	%3E
¬ (logical not)	%AC

FILE resources

The following file types are supported for definition in CICS bundles:

- VSAM files (including files that refer to CICS-maintained, user-maintained, and coupling facility data tables, as well as files that refer to VSAM data sets)
- Remote VSAM files
- Remote BDAM files

The initial status of a FILE resource that is dynamically created from a CICS bundle is derived from the initial status of the CICS bundle that defines the resource. As a result, it is not possible to define a FILE resource with a STATUS of UNENABLED to inhibit the implicit opening of files by applications.

The PASSWORD attribute is not supported for dynamically created FILE resources.

When a file that is defined in a CICS bundle is installed, it is added to the catalog. CICS recovers the bundle installed files from the catalog during a warm or emergency restart. These recovered files are used for the CICS recovery only. After CICS completes the recovery, all these files are deleted. When the bundle is re-created, CICS picks up the definition from the bundle directory, and creates a new file. If a file recovered from the catalog has a retained lock, it cannot be deleted on completion of recovery, and as a result, the file becomes orphaned (does not belong to any bundle) and has to be deleted manually after CICS has completed restart.

JVMSERVER resources

For JVM servers that are packaged in CICS bundles, the JVM profiles are packaged with the resource definitions in the CICS bundles. You can therefore install the JVM server in any CICS region without needing to set up a JVM profile in the local JVM profile directory for the CICS region.

When creating a JVM server, you can create the JVM profile using sample templates for an OSGi JVM server, an Axis2 JVM server, or a Liberty JVM server, or import an existing JVM profile to the CICS bundle from elsewhere in the workspace or from the local file system.

The set of acceptable characters in the JVM profile name is more restricted when the JVMSERVER resource is defined in a CICS bundle. For details, see JVMSERVER attributes. The directory name for the directory where the JVM profile is stored is limited to 240 characters, which is the same limit that applies to JVM profiles that are not defined in CICS bundles.

A JVM server that is defined in a CICS bundle must be installed and enabled before you install OSGi bundles or other application artifacts for Java applications that run in it. It is good practice to deploy a CICS bundle containing the definition for a JVMSERVER resource as part of a platform bundle, and to then deploy the CICS bundles containing OSGi bundles or other Java application artifacts as part of applications that are deployed on the platform. This architecture ensures that when the resources are first installed in a CICS region or if the CICS region is restarted, the JVM server and the Java application resources are installed and enabled in the correct order.

LIBRARY resources

If a LIBRARY resource is included in a CICS bundle, it should only be used for loading the modules used by that CICS bundle. Although advisable, this requirement is not enforced for a stand-alone CICS bundle, but it is enforced for a CICS bundle that is packaged and installed as part of an application on a platform.

The BUNDLEPART resource associated with the LIBRARY resource does not report a status of DISABLED until the LIBRARY concatenation and all programs loaded from it have a use-count of zero. It might be necessary to purge work from CICS so that the bundle disable process can complete.

If a program is loaded from a LIBRARY concatenation that was defined and installed by a CICS bundle, that program is treated as though it was defined in the CICS bundle, even though the program might have been defined independently. The lifecycle of these programs conforms to the lifecycle of the CICS bundle in which the LIBRARY resource is defined. If a CICS bundle that contains a LIBRARY resource is disabled, all programs loaded from that LIBRARY concatenation are disabled.

PIPELINE resources

For pipelines that are packaged in CICS bundles, the pipeline configuration files are also packaged with the resource definitions in the CICS bundles. You can create a pipeline configuration file using one of the CICS-supplied sample pipeline configuration files, or import an existing pipeline configuration file from the local file system.

If a PIPELINE resource is packaged in a CICS bundle, it should be deployed as part of a platform for hosting WEBSERVICE resources that are defined using CICS bundles. PIPELINE resources that are defined in CICS bundles can only be used with WEBSERVICE resources that are defined in CICS bundles or created dynamically by a pipeline scan. WEBSERVICE resources defined using the CICS CSD or BAS are not compatible with PIPELINE resources that are defined in CICS bundles.

A PIPELINE resource that is packaged in a CICS bundle must be installed and enabled before you install the WEBSERVICE resources that require it. It is good practice to deploy a CICS bundle containing the definition for a PIPELINE resource as part of a platform bundle, and to then deploy the CICS bundles containing WEBSERVICE resources as part of applications that are deployed on the platform. This architecture ensures that when the resources are first installed in a CICS region or if the CICS region is restarted, the PIPELINE resource and the WEBSERVICE resources are installed and enabled in the correct order.

A PIPELINE resource that is defined in a CICS bundle can specify a WSDIR attribute, but this is discouraged. The SHELF attribute is not used for PIPELINE resources that are defined in CICS bundles.

PROGRAM resources

If a program is in use and the CICS bundle from which it was installed is disabled, CICS disables the program, and no new work can be started by that program. However, the associated BUNDLEPART resource remains enabled until the use-count for the program reaches zero. As a result, during this period the BUNDLEPART resource, and the program both report different states. If the

use-count for the program does not reach zero after an acceptable interval, it might be necessary to purge work from CICS so that the bundle disable process can complete.

When the bundle state has changed from DISABLING to DISABLED, the CICS bundle can be discarded. However, if you discard the CICS bundle, the program that was installed by the bundle is discarded and becomes unavailable until the CICS bundle is reinstalled.

The PROGRAM NEWCOPY and PROGRAM PHASEIN processes are not available for programs installed from CICS bundles. The process for a program defined in a CICS bundle is to install a new version of the CICS bundle, or of the application with which the CICS bundle was deployed. For PROGRAM resources defined in stand-alone CICS bundles, you must uninstall the previous version of the CICS bundle before you install the new version. When you deploy PROGRAM resources defined in CICS bundles as part of an application on a platform, you can install new versions of the PROGRAM resource without uninstalling the previous versions, and manage the availability of the new versions for users. For more information about multi-versioning for applications, see Multi-versioning for applications deployed on platforms.

Programs that are installed by CICS bundles are not suitable for invocation from the PLT.

PROGRAM resources can be declared as application entry points, although the PROGRAM resource does not have to be defined in a CICS bundle to be declared as an application entry point. Programs that are declared as an application entry point must have a unique PROGRAM resource name in your environment. For more information about programs as application entry points, see Application entry points in Product overview.

TRANSACTION resources

A transaction might be allocated work that is due to be started, by an EXEC CICS START command, at a scheduled time. If that work is scheduled to start after the CICS bundle is disabled, the scheduled work is canceled, and no new work is allowed to start for that transaction. If the CICS bundle is then re-enabled, the canceled work is not rescheduled, and remains canceled.

TRANSACTION resources can be declared as application entry points, although the TRANSACTION resource does not have to be defined in a CICS bundle to be declared as an application entry point. A TRANSACTION that is declared as an application entry point must have a unique TRANSACTION resource name in your environment. For more information about TRANSACTION's as application entry points, see Application entry points in Product overview.

URIMAP resources

A WEBSERVICE resource is normally associated with at least one URIMAP resource. The URIMAP resources can be packaged in the same bundle as the WEBSERVICE resource, but it can be preferable to package them in a separate bundle that can be customized for each platform to which the WEBSERVICE resource is deployed. For example, you might want to use a different URI in a quality assurance platform than is used in a production platform, or define a mirror transaction for the WEBSERVICE resource in the production platform but not in the unit testing platform.

When you define a URIMAP resource in a CICS application bundle, you can use an application entry point declaration to control users' access to the service provided by the URIMAP resource. In this situation, when you install and enable the application, the service provided by the URIMAP resource is not yet available to callers. When you choose to provide the service to users, use the CICS Explorer to make the application containing the CICS bundle available. This makes the application entry point, and therefore the service provided by the URIMAP resource, available to callers.

You can also declare as an application entry point a URIMAP resource that is defined outside the application. In this situation, the service becomes available to users as soon as you install and enable the URIMAP resource.

When you are using a URIMAP resource to provide a static response from CICS as an HTTP server, the HFSFILE attribute specifies the name of a z/OS UNIX System Services zFS file that forms the body of the response. For a URIMAP resource that is defined in a CICS bundle, the zFS file must be packaged in the CICS bundle with the URIMAP resource. The HFSFILE attribute should specify a relative file path, which is relative to the root directory of the CICS bundle. For a relative file path, do not use a slash at the beginning of the path. For more information, see "Referencing zFS artifacts in a bundle." Absolute references to zFS locations outside of the bundle are possible, but strongly discouraged.

WEBSERVICE resources

For web services that are packaged in CICS bundles, you can import a web service binding file and a WSDL document or WSDL archive file to be packaged in the bundle along with the resource definition. To support the web service, you can use a WEBSERVICE definition packaged in a CICS bundle to generate a URIMAP definition in a separate bundle. You can also create an alias transaction for the URI map, and an optional URIMAP definition for WSDL discovery.

Web services that are packaged in CICS bundles have additional states of DISABLING and DISABLED, which do not apply to web services created using other methods. When disabling is in progress for a CICS bundle, WEBSERVICE resources defined in the bundle enter DISABLING state. Work that is currently executing is allowed to complete, but the web service does not accept new work. When the web service is no longer in use, the WEBSERVICE resource enters DISABLED state. Requests to a web service in DISABLING or DISABLED state are rejected and a SOAP fault, "Web service is not in service", is sent. If CICS is the web service requester, the INVOKE SERVICE command returns a RESP code of INVREQ and a RESP2 value of 8.

If the CICS bundle is enabled again, the WEBSERVICE resource returns to INSERVICE state. Otherwise, the WEBSERVICE resource can be discarded by discarding the CICS bundle. You can inquire on the state of a WEBSERVICE resource using the EXEC CICS or CEMT **INQUIRE WEBSERVICE** command, the CICSPlex SM web user interface, or the CICS Explorer, but you cannot set it manually.

Referencing zFS artifacts in a bundle

A number of CICS resources reference external zFS artifacts for further configuration information. For example, JVMSERVER resources require a JVM profile, and PIPELINE resources require a pipeline configuration file. If these

resources are defined in a CICS bundle, the zFS artifacts that they require must also be stored in the CICS bundle, and referenced using a relative zFS name.

Storing the zFS configuration files in the CICS bundle ensures that the resource definition and its configuration files are deployed as an atomic unit. If you update the configuration files after deployment, make the change in the CICS bundle project in the CICS Explorer, and the bundle should be redeployed. In most cases the CICS Explorer contains rich editors for working with the configuration files.

When the CICS bundle containing the resource definition is installed, CICS resolves the reference to the zFS configuration file relative to the CICS bundle's root directory. The configuration file can be stored in the root directory or in a subdirectory in the CICS bundle project.

For applications that are deployed in a JVM server, the **&CONFIGROOT;** symbol can be used in the JVM profile, which is substituted at runtime with the root directory for the CICS bundle. Applications can use the value of the **com.ibm.cics.jvmserver.configroot** system property to locate the configuration files for the JVM server.

Private resources for application versions

When you define certain CICS resources in CICS bundles as part of an application installed on a platform, the resources are private to that version of that application. You can therefore install more than one resource of those types with the same name, at the same time, on the same platform instance.

For supported resource types, a CICS resource is private if the resource is defined in a CICS bundle that is packaged and installed as part of an application, either as part of the application bundle, or as part of the application binding. When you create a CICS resource in this way, the resource is not available to any other application or version installed on the platform, and it is not available to other applications in the CICS region. It can be used only by the version of the application where the resource is defined. These resources are known as private resources.

The following CICS resources are supported as private resources for applications:

- LIBRARY resources, which represent one or more data sets, known as dynamic program LIBRARY concatenations, from which program load modules can be loaded.
- PACKAGESET resources, which represent DB2 collections and are used to qualify which table in a DB2 database an unqualified EXEC SQL request refers to.
- POLICY resources, which represent one or more rules that manage the behavior of user tasks in CICS regions.
- PROGRAM resources, which represent an application program. A program that is auto-installed by a task for an application that is deployed on a platform is also private to that version of the application.

CICS resources of other resource types that are defined as part of applications, and CICS resources that are defined by any other methods, are publicly available for all tasks. These resources are known as public resources. In applications that have only a single version, private resources that are declared as application entry points become public resources when the application entry point is made available. For multi-versioned applications, if the application is the highest available version,

a program that is declared as an application entry point is public. The programs that are declared as application entry points for the other versions of the same application are private.

If you do not want a resource of the supported resource types to be private, do not package the resource definition as part of an application. Instead, define the CICS resource using a standalone CICS bundle, a CICS bundle that is installed at the level of a platform, the CICS CSD, or the CICSPlex SM data repository. If an application requires the CICS resource to be available in the CICS region, add the resource as a dependency for the application or the application binding, in an <import> element of the bundle manifest.

POLICY and PACKAGESET resources, which can be defined only in CICS bundles, have the same support as private resources when they are defined in a CICS bundle that is deployed as part of an application. You can therefore use POLICY and PACKAGESET resources with the same name in different applications and application versions.

Private resources relate to a specific version of the application, not just a specific application. They therefore enable the installation of multiple versions of applications on the same platform instance, at the same time. For more information about multi-versioning for applications, see Multi-versioning for applications deployed on platforms.

Managing private resources for applications

When you define a private resource in a CICS bundle that is packaged and installed as part of an application, the resource name does not have to be unique in your installation. You can use this facility to avoid resource name clashes between applications that were developed independently, but used the same resource names. The requirement for unique resource names for the supported CICS resources can be removed by managing the resources as part of applications deployed on a platform. You can use this process to assist with server consolidation.

When you change an application, you use CICS Explorer to modify the relevant elements of the application that are packaged in CICS bundles, apply a new version number to those CICS bundles to identify the change, and then reversion and reinstall the application. You leave any unmodified CICS bundles at the same version number as before, because CICS manages the process of multiple installations of CICS bundles with the same ID and version number.

Reinstallation of unmodified CICS bundles is only available for CICS bundles that are installed as part of an application deployed on a platform. Standalone CICS bundles cannot be reinstalled if they are already installed with the same ID and version, or in the case of CICS bundles created in releases before CICS TS Version 5.1, installed with no ID and version. However, the same CICS bundle can be installed in CICS regions as a standalone CICS bundle, and also installed and reinstalled as part of one or more applications deployed on a platform.

Special CICS messages are issued when actions such as installing and discarding are performed on private resources. The messages provide the same information as for the corresponding actions on public resources of that type, but they also state the platform, application, and application version to which the private resource applies, so that you can audit or troubleshoot the actions in the relevant context.

To view the private resources for each installed version of an application, use the CICS Explorer. In the application descriptor editor, you can view the private resources and the application entry points for the application by resource type, and filter them by CICS region or by CICS bundle to help locate particular resources. You can also view the DD names that z/OS has generated for the LIBRARY concatenation of data sets for private LIBRARY resources.

You can inquire on or browse private resources using the **EXEC CICS INQUIRE** system programming command for the resource type. By default, CICS searches for the resources that are available to the program where the **EXEC CICS INQUIRE** command is issued. You can also choose to browse private resources for a specified application.

- When you issue an **EXEC CICS INQUIRE** command from a public program, information is returned about the named public resource. If the resource is not available as a public resource, a "not found" response is returned.
- When you issue an **EXEC CICS INQUIRE** command from a program that is running under a task for an application deployed on a platform, information is returned about the named private resource for that application, if it exists. If the application does not have a private resource with that name, information is returned about a public resource with the specified name. If the resource is not available as a private resource for that application or as a public resource, a "not found" response is returned.
- When you use an **EXEC CICS INQUIRE** command in browse mode from a public program, if you do not specify any other input parameters, the set of public resources of the specified type is returned. If the same browse command is issued from a program that is running under a task for an application deployed on a platform, the browse returns a set of resources consisting of any private resources of the specified type for the application, and the public resources of the specified type.
- To browse the private resources for an application, from either a public program or a private program, issue the **EXEC CICS INQUIRE** command with the START option and specify as input the application context, consisting of the platform, application, and application version. The browse returns a set of resources consisting of only the private resources of the specified type for the application. If no application is found with the specified application context, the APPNOTFOUND condition is returned.

For more information about browsing private resources, including examples of browsing resources for a different application from the application where you issue the command, see Browsing resource definitions.

To support private resources in the CICSPlex SM real-time analysis (RTA) function, you must specify the application context parameters PLATFORM, APPLICATION, APPLMAJORVER, APPLMINORVER and APPLMICROVER in the evaluation definition (EVALDEF) for the resource. For instructions to create and maintain real-time analysis definitions using user-defined CICSPlex SM WUI views and resource objects, see Real-time analysis.

CICS produces separate statistics records for private resources. A statistics record for a private resource has information about the application for which the resource was defined. The statistics DSECTs and DFHSTUP reports for public program and library resources have corresponding DSECTs and DFHSTUP reports for private resources. Programs that are declared as application entry points are identified and reported in both the public and private statistics, because, while the entry point is publicly accessible, it is also part of the application.

Private resources as dependencies

In a CICS bundle that is packaged as part of an application deployed on a platform, public or private resources can be defined as dependencies, including private programs that are autoinstalled by an application. CICS checks first for a private resource with the specified type and name in the same application as the CICS bundle that contains the <import> element in the bundle manifest file. If more than one version of the application is installed, the current application context determines the version of the private resource that is imported. If the resource is not found as a private resource for the application, CICS checks for a public resource in the CICS region that matches the specified type and name, and imports that resource. You cannot import private resources from other applications.

Private LIBRARY resources

LIBRARY resources represent one or more data sets, known as dynamic program LIBRARY concatenations, from which program load modules can be loaded. The LIBRARY resource is supported as a private resource for an application version. Each version of an application should include at least one private LIBRARY resource representing the version-specific data sets containing the load modules for the application.

- When a task for an application deployed on a platform requires a program load, the private LIBRARY concatenations defined for that version of the application are searched first, so that the correct version of the program is loaded. If multiple LIBRARY concatenations are defined for the same application version, they are searched in order of their ranking.
- If there are no LIBRARY concatenations defined in the application, or the program is not found in any of the private LIBRARY concatenations, the public LIBRARY concatenations defined for the whole CICS region (including DFHRPL) are searched.
- For a task that is not associated with an application deployed on a platform, only the public LIBRARY concatenations are searched, so all program load modules that are resident in private LIBRARY concatenations are not available.

If any of the private LIBRARY concatenations for an application are disabled, because the CICS bundle that defines the relevant LIBRARY resource is disabled, CICS does not search any other private LIBRARY concatenations, or any public LIBRARY concatenations that are defined for the whole CICS region. All subsequent program loads by the application therefore fail until the CICS bundle that defines the LIBRARY resource is enabled.

For a private LIBRARY resource that is defined in a CICS bundle that is packaged and installed as part of an application bundle or application binding bundle, the name of the LIBRARY resource is not used as the DD name for the LIBRARY concatenation of data sets. Instead, CICS requests a unique DD name for the LIBRARY concatenation of data sets when the application is installed on the platform. The resource name can therefore be the same as LIBRARY names used elsewhere in the installation, or by different versions of the application. CICS issues message DFHLD0518 to state the DD name that z/OS has generated for the LIBRARY concatenation. You can also view the data set names for an installed application in the CICS Explorer.

Private PACKAGESET resources

The private resource PACKAGESET represents a DB2 collection. A PACKAGESET resource has a single attribute, name, which is the name of a DB2 collection identifier and is up to 128 bytes in length. Use a PACKAGESET to specify different collection identifiers across different environments by enabling CICS to issue the EXEC SQL SET CURRENT PACKAGESET command on behalf of the application. This resource makes the handling of data in a cloud environment easier and more flexible.

An application can have a maximum of two PACKAGESET resources associated with it. If more than one PACKAGESET resource is installed, the associated resource is selected in the following order:

1. The PACKAGESET resource that is installed as part of the application or the application binding. Installing a PACKAGESET resource as part of the application binding enables the use of different collection identifiers in different environments without changing the application bundle. For example, test, quality assurance, and production environments can each associate the application with a different PACKAGESET resource, and therefore a different collection identifier. If a PACKAGESET resource is installed as part of an application rather than its application binding, the same PACKAGESET resource and therefore the same collection identifier will be used in all environments.
2. The PACKAGESET resource that is installed as part of the platform. With this PACKAGESET resource associated, the same PACKAGESET resource and therefore the same collection identifier will be used by all applications running on the platform.

When the CICS-DB2 attachment facility processes the first SQL request for a unit of work, the facility checks whether a PACKAGESET resource exists that applies to the task. First, it checks whether a PACKAGESET resource is defined whose scope matches the platform, application and version in the application context of the task. If it is not, the facility checks whether a PACKAGESET resource is defined for all applications on the platform. If a PACKAGESET resource exists, the CICS-DB2 attachment facility links to a CICS-supplied program, DFHD2SPS, to issue an **EXEC SQL SET CURRENT PACKAGESET** command before it processes the first SQL request from the application.

For subsequent SQL requests, no additional processing is required unless the application context changes. If the application context has changed, the CICS-DB2 attachment facility links to DFHD2SPS to either set the new package set name, or to set a blank package set name if the new context does not have a PACKAGESET resource defined. Examples of when application context can change are when an **EXEC CICS LINK** command has been issued into a new application, or when an **EXEC CICS LINK** call returns to the calling application.

At syncpoint time, the CICS-DB2 attachment facility links to DFHD2SPS to reset the PACKAGESET resource to blank.

DFHD2SPS runs as an concurrency(REQUIRED), API(CICSAPI), EXECKEY(CICS) program on an L8 open TCB. It is linked to from the CICS-DB2 TRUE DFHD2EX1, which typically is running on an L8 TCB, so no TCB switching overhead is incurred. However, for Java programs and for XPLINK C programs, DFHD2EX1 runs on a T8 or X8 TCB respectively, and so TCB switches will occur to the L8 TCB and back again when invoking DFHD2SPS.

The PACKAGESET resource is optional, and existing mechanisms to manage different collections across different environments remain available, for example, multiple plans, dynamic plan exits, or setting the package set yourself in the application.

Attention: Existing programs that use **SET CURRENT PACKAGESET** commands must reestablish their package sets after calling applications that use a PACKAGESET resource. To reestablish package sets, issue another **SET CURRENT PACKAGESET** command. CICS has no knowledge of package sets other than those defined by a PACKAGESET resource.

Only use the bundle-defined PACKAGESET resource as part of an application, typically through the application binding. Any attempt to install a stand-alone bundle that contains a PACKAGESET resource that is not part of an application or platform will fail.

Private POLICY resources

POLICY resources represent one or more rules that manage the behavior of user tasks in CICS regions. A policy is an XML definition that contains one or more rule types with associated thresholds and actions. The policy rules describe the controls or actions that can be applied to one or more tasks. A condition and action pair make up a policy rule, and one or more policy rules can be defined within a policy. A policy is defined in a CICS bundle and a CICS bundle can consist of one or more policies.

CICS performs the action that is specified in the policy when tasks that are running exceed defined thresholds for resource usage. You can deploy policies to monitor the resource utilization of a user task, and to automatically respond when resource usage exceeds the thresholds you define. As long as a task is relinquishing control to CICS by issuing **EXEC CICS** commands, excessive resource usage, and looping and runaway transactions, can be detected and dealt with appropriately.

Policies are always defined in CICS bundles, which are always deployed to a specific scope. The scope can be an application, an operation within the application, or a platform.

Policies define a contract about how you want aspects of an application or system to behave. Policies apply to instances of user tasks, and different groups of tasks can share a common set of policies, thus allowing sections of the workload to be controlled in different ways. The collection of policy rules that are associated with tasks that are running in an application or platform define the boundaries within which a task can safely execute. They allow systems administrators to set, in advance, limits on the behavior of user applications. In this way, policies enable CICS to measure, react to, and enforce the behavior of tasks that transgress any of the rules that are defined in the policies that apply to them, as they execute.

For more information about polices, see CICS Policies.

Private PROGRAM resources

A PROGRAM resource represents a program load module that is stored in the program library. The PROGRAM resource is supported as a private resource for an application version. A program that is auto-installed by a task for an application that is deployed on a platform is also private to that version of the application.

Only one copy of each version of each program is loaded in CICS storage. Before loading a private program, CICS checks whether that version of the program has already been loaded from the same data set (PDS or PDSE) with a matching PROGRAM resource definition. If so, CICS uses the existing copy. The following rules therefore apply when you are reusing PROGRAM resource names:

- Multiple applications that are intended to share the same program loaded from the same PDS or PDSE must use the same attributes in the PROGRAM resource definition.
- If multiple applications use the same name for different program resources, each application must load the programs from a different data set (PDS or PDSE).

If you specify **RELOAD=YES** in the PROGRAM resource definition for a private program, its behavior for program loading changes to be the same as for a public program. A program control link, load, or XCTL request brings a fresh copy of the program into storage. RELOAD(YES) programs cannot be reused, and they cannot be shared by multiple applications. Each of the program copies must be removed from storage explicitly, using a storage control FREEMAIN request, when it is no longer required and before the task terminates.

If a program that is required by an application is not found in the private program directory for the application, CICS searches the public program directory.

When the **EXEC CICS LINK**, **XCTL**, **LOAD**, and **RELEASE** commands are issued by a program that is running under a task for an application deployed on a platform, CICS searches first for the named program in the private program directory for the application. If the named program is not found there, CICS then searches the public program directory.

Programs that are declared as an application entry point must have a unique PROGRAM resource name in your environment. To enable these programs to be called from outside the application, they must be public resources. When you make an application available that contains an application entry point for a private PROGRAM resource, the PROGRAM resource that is named as the application entry point changes from a private resource to a public resource. Only one instance of a public resource with a particular name can exist in a CICS region. The PROGRAM resource therefore cannot have the same name as a public program that is installed in the CICS region, or the same name as a public program that is defined as an application entry point by a different installed application. However, multiple versions of the same PROGRAM resource defined as an application entry point can be installed for multiple versions of the same application, because CICS manages the promotion of PROGRAM resources to public status for the versions of an application.

Manifest contents for a CICS bundle

Each CICS bundle contains a bundle manifest that describes the contents of the bundle. A bundle manifest describes which resources to create in the CICS region and the location of the supporting artifacts, which prerequisites are required for the CICS bundle to run successfully, and any services that the CICS bundle can offer to other bundles.

The manifest for a CICS bundle is called `cics.xml` and is in the `META-INF` directory of the deployed bundle on z/OS UNIX. The bundle manifest is written in XML and conforms to a schema. For more information, see . The bundle manifest is encoded in UTF-8 and contains information about the bundle itself, as well as definitions, exports, imports, application entry points, and policy scopes.

Bundle information

The manifest can contain an optional <meta-directives> element that contains information about the bundle; for example, it can contain a time stamp of the time when the bundle was created.

Definitions

Every resource in the bundle is defined by its name, resource type, and location in a <define> element:

- The name of the artifact is defined in the tooling.
- The resource type is defined as a URI.
- The location is defined as a relative file path, including the file name, to the artifact that provides the definition and metadata for the resource.

Some definitions lead to CICS creating one or more resources dynamically; for example a `http://www.ibm.com/xmlns/prod/cics/bundle/EVENTBINDING` resource type creates an EVENTBINDING resource and one or more CAPTURESPEC resources. Other definitions, such as the `http://www.ibm.com/xmlns/prod/cics/bundle/SCACOMPOSITE` resource type, are used by CICS only in runtime processing and have no equivalent CICS resource definition.

For information about the types of resources that you can define and deploy in CICS bundles, see Artifacts that can be deployed in bundles in Administering.

A resource is installed only if its type has been registered with CICS. For vendor resource types, CICS provides a registration and callback interface. For more information, see .

Exports

The <export> element provides additional information about the resources or services that a bundle can provide. CICS does not use export statements in its processing of bundles. You can view the list of exports in the CICS Explorer.

Imports

The <import> element defines references to resources that are required by the bundle. Each reference has a name and a type, but no path attributes. The reference also has an attribute that describes how CICS handles the BUNDLE resource installation if one of the required imports is not present in the CICS region. The BUNDLE resource can remain disabled, enable with warning messages, or enable with no warnings. The default behavior is that all imports are required; the BUNDLE resource can be installed but not enabled.

In a CICS bundle that is packaged as part of an application deployed on a platform, public or private resources can be defined as dependencies, including private programs that are autoinstalled by an application. CICS checks first for a private resource with the specified type and name in the same application as the CICS bundle that contains the <import> element in the bundle manifest file. If more than one version of the application is installed, the current application context determines the version of the private resource that is imported. If the resource is not found as a private resource for the application, CICS checks for a public resource in the CICS region that matches the specified type and name, and imports that resource. You cannot import private resources from other applications.

The <import> element can contain the following CICS resource types:

- `http://www.ibm.com/xmlns/prod/cics/bundle/ATOMSERVICE`

- http://www.ibm.com/xmlns/prod/cics/bundle/DB2CONN
- http://www.ibm.com/xmlns/prod/cics/bundle/DB2ENTRY
- http://www.ibm.com/xmlns/prod/cics/bundle/DB2TRAN
- http://www.ibm.com/xmlns/prod/cics/bundle/DOCTEMPLATE
- http://www.ibm.com/xmlns/prod/cics/bundle/ENQMODEL
- http://www.ibm.com/xmlns/prod/cics/bundle/EPADAPTER
- http://www.ibm.com/xmlns/prod/cics/bundle/EPADAPTERSET
- http://www.ibm.com/xmlns/prod/cics/bundle/EVENTBINDING
- http://www.ibm.com/xmlns/prod/cics/bundle/FILE
- http://www.ibm.com/xmlns/prod/cics/bundle/JOURNALMODEL
- http://www.ibm.com/xmlns/prod/cics/bundle/JSONTRANSFRM
- http://www.ibm.com/xmlns/prod/cics/bundle/JVMSERVER
- http://www.ibm.com/xmlns/prod/cics/bundle/LIBRARY
- http://www.ibm.com/xmlns/prod/cics/bundle/MAPSET
- http://www.ibm.com/xmlns/prod/cics/bundle/MQCONN
- http://www.ibm.com/xmlns/prod/cics/bundle/PARTITIONSET
- http://www.ibm.com/xmlns/prod/cics/bundle/PIPELINE
- http://www.ibm.com/xmlns/prod/cics/bundle/PROCESSTYPE
- http://www.ibm.com/xmlns/prod/cics/bundle/PROGRAM
- http://www.ibm.com/xmlns/prod/cics/bundle/SCACOMPOSITE
- http://www.ibm.com/xmlns/prod/cics/bundle/TCPIPSERVICE
- http://www.ibm.com/xmlns/prod/cics/bundle/TDQUEUE
- http://www.ibm.com/xmlns/prod/cics/bundle/TRANSACTION
- http://www.ibm.com/xmlns/prod/cics/bundle/TSQMODEL
- http://www.ibm.com/xmlns/prod/cics/bundle/URIMAP
- http://www.ibm.com/xmlns/prod/cics/bundle/WEBSERVICE
- http://www.ibm.com/xmlns/prod/cics/bundle/XMLTRANSFORM

Vendors can extend this list to include their own user resource types.

Application entry points

If your CICS bundle is part of a CICS application, you can define one or more application entry points in the bundle manifest. An application entry point identifies a resource that is an access point to an application. Application entry points are used to control users' access to different versions of an application that is deployed on a platform. They are also used to create an application context to monitor the resource usage for applications and to identify an application being run.

The <entrypoint> element in the <modify> element defines an application entry point. The bundle manifest can contain 0 to many <modify> elements for application entry points. The <entrypoint> element specifies the name of the CICS resource, the type of resource, and the application operation. Resources that are specified as application entry points do not have to be defined in the bundle or specified as imports.

Policy scopes

A policy scope is used to limit a policy to a specific application operation. When a policy is deployed with an application scope, the policy applies to all user tasks within the platform that have the matching platform,

application and application version information in their application context. A policy scope is used to further limit a policy to only those user tasks that also match the operation.

The <policyscope> element in the <modify> element defines a policy scope for a policy. The bundle manifest can contain 0 to many <modify> elements for policy scopes. The <policyscope> element specifies the name of the application operation and the name of the policy to apply to user tasks for that operation. The name of the operation must be defined by a separate <entrypoint> element and the policy must be defined by a <define> element either in the same bundle or in a separate bundle which is deployed with the bundle defining the policy scope.

Example of a manifest

The following example shows a manifest for a CICS bundle that contains a program definition that is also an application entry point, together with a policy and policy scope for user tasks for that application entry point.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<manifest xmlns="http://www.ibm.com/xmlns/prod/cics/bundle"
     build="IntB-201402121955" bundleRelease="0" bundleVersion="1"
     bundleMicroVer="0" bundleMinorVer="0" bundleMajorVer="1" id="EXAMPLE">
    <meta_directives>
        <timestamp>2014-03-12T13:17:23.988Z</timestamp>
    </meta_directives>
    <import
           warn="true"
           optional="false"
           type="http://www.ibm.com/xmlns/prod/cics/bundle/PROGRAM"
           name="BASESCOP"/>  1
    <import
           warn="true"
           optional="true"
           type="http://www.ibm.com/xmlns/prod/cics/bundle/FILE"
           name="BNDRES07"/>  2
    <import
           warn="false"
           optional="true"
           type="http://www.ibm.com/xmlns/prod/cics/bundle/TRANSACTION"
           name="BR14"/>  3
    <define
           path="MYPROG.program"
           type="http://www.ibm.com/xmlns/prod/cics/bundle/PROGRAM"
           name="MYPROG"/>  4
    <define
           path="MYPOL.policy"
           type="http://www.ibm.com/xmlns/prod/cics/bundle/POLICY"
           name="MYPOL"/>  5
    <modify>
        <entrypoint
           operation="READ"
           type="http://www.ibm.com/xmlns/prod/cics/bundle/PROGRAM"
           name="MYPROG"/>  6
    </modify>
    <modify>
        <policyscope
           operation="READ"
           type="http://www.ibm.com/xmlns/prod/cics/bundle/POLICY"
           name="MYPOL"/>  7
    </modify>
</manifest>
```

1. The bundle imports a program that is required in the CICS region. If this PROGRAM resource is not present in the CICS region, the bundle installs successfully but is placed in a disabled state. CICS also issues a warning message.
2. The bundle imports a file that is optional. If this FILE resource is not present in the CICS region, the bundle installs successfully but is placed in a disabled state. CICS also issues a warning message.
3. The bundle imports a transaction that is optional. If this TRANSACTION resource is not present in the CICS region, the bundle installs successfully but no warning message is issued by CICS.
4. The bundle defines a PROGRAM resource, including the name of the resource and the path to the XML file that defines the resource attributes.
5. The bundle defines a POLICY resource, including the name of the policy and the path to the XML file that defines the policy attributes.
6. The bundle declares a PROGRAM resource as an application entry point for a particular operation.
7. The bundle declares that the POLICY is to be applied only to tasks which match the specified operation.

Scoping of bundles

The BUNDLE resource definition provides the BASESCOPE attribute as a way of applying a scope to related BUNDLE resources. You can use this attribute for bundles that are deployed on a platform, or to set a Service Component Architecture (SCA) domain for a bundle that contains SCA composite applications.

BASESCOPE is an optional attribute on the BUNDLE resource definition that you can use in different ways. You can use the IBM CICS Explorer to view all of the BUNDLE resources that are defined in a CICS region and order them by the value of the BASESCOPE attribute.

Scoping of bundles for a platform

CICS bundles that are deployed on a platform automatically have a BASESCOPE applied when the BUNDLE resource is created during an application deployment. A BUNDLE resource that is deployed on a platform as part of an application bundle or an application binding bundle has a URI in the BASESCOPE attribute with the following format:

`cicsapplication://`*`Platform/ApplicationID/MajorVersion/MinorVersion/MicroVersion`*

Platform is the name of the platform, *ApplicationID* is the ID of the application bundle, followed by the major, minor, and micro version of the application.

CICS uses the BASESCOPE attribute to identify the relevant application and version, so that multiple versions of the same application can be installed on a platform. The BASESCOPE attribute can be used to restrict resources to the appropriate version of the application. For example, certain supported resource types that are defined in a CICS bundle, then packaged and installed as part of an application bundle or application binding bundle, are private to that version of that application. For more information about private resources for applications, see Private resources for application versions.

The same cicsapplication format basescope can be added to CICS bundles that are outside of a platform environment. This is useful in a stand-alone CICS region (SMSS) where you do not have a platform and applications. In the SMSS case, it is

the responsibility of the user to ensure that the basescope is properly specified, whereas in a platform CICS automatically generates the correct basescope attribute.

Scoping of bundles for a policy

Policies are defined in CICS bundles. In a platform, policies are deployed to a specific scope. The scope can be an application scope, an operation (within an application) scope, or a platform scope. For more information, see Policy scopes.

Scoping of bundles into SCA domains

The BASESCOPE attribute has a specific use for bundles that contain SCA composite applications. A composite application is deployed on an SCA domain. An *SCA domain* typically represents a set of services that provide an area of business function that is controlled by a single organization; for example, the SCA domain for an accounts department in a business might cover all financial related functions and contain a series of composite applications that deal with specific areas of accounting.

In a CICS region, by default there is one SCA domain. Every bundle that is deployed on the CICS region has the same default SCA domain, although the value is empty. You can use the BASESCOPE attribute on the BUNDLE resource definition to set a value for the SCA domain.

You can also deploy the same bundle multiple times into the CICS region by specifying different SCA domains for the BASESCOPE attribute. CICS uses the SCA domain and the composite together to identify the service during runtime processing. The scope of the service is available to the task that is processing the request.

Use a unique URI for the BASESCOPE attribute value; for example, `http://mycompany/HR` or `http://mycompany/warehouse`. CICS creates the names of services, composites, and references by extending the value of the BASESCOPE attribute; for example, installing a service with a local name of `location/taxService` into the HR SCA domain would create a scoped name of `http://mycompany/HR/location/taxService`.

Do not extend the same URI to create a new SCA domain. If you extend the same URI, you might get unexpected service or reference name clashes; for example, if you used `http://mycompany/HR` and `http://mycompany/HR/location` as different SCA domains and had a service with a local name of `location/taxService` and another service called `taxService`, installing these services into both the `http://mycompany/HR` and `http://mycompany/HR/location` SCA domains create clashes with the service names. Although you can install and enable BUNDLE resources successfully with these values, you might get unexpected results and errors when the services are called by other applications.

OSGi bundle recovery on a CICS restart

When you restart a CICS region that contains OSGi bundles, CICS recovers the BUNDLE resources and installs the OSGi bundles into the framework of the JVM server.

OSGi bundles that are packaged in CICS bundles are not stored in the CSD. The BUNDLE resource itself is stored in the catalog, so that on a restart of the CICS region, the OSGi bundles are dynamically re-created when the BUNDLE resource is restored.

On a cold, warm, or emergency restart of CICS, the JVM server is started asynchronously to BUNDLE resource recovery. The JVM server must be fully available to successfully restore an OSGi bundle on a CICS restart. Therefore, although the BUNDLE resources are recovered during the last phase of CICS startup, the OSGi bundles are installed only when the JVM server has completed its startup.

BUNDLE resources and the OSGi bundles that they contain are installed in the correct order to ensure that the dependencies between both CICS bundles and OSGi bundles are resolved in the framework. If CICS fails to install an OSGi bundle, the BUNDLE resource installs in a disabled state. You can use the IBM CICS Explorer to view the state of BUNDLE resources, OSGi bundles, and OSGi services.

Security for bundles

Different security profiles and checks apply to BUNDLE resources created from a definition, and to BUNDLE resources that are created when you install an application or platform.

Security for BUNDLE definitions

For CICS bundles that are defined in individual CICS regions, CICS command and resource security checks apply when you perform actions on the BUNDLE resource, in the same way as they do for other CICS resources. The XRES system initialization parameter for the CICS region specifies whether or not security checking is carried out for the BUNDLE resource type, among others. Resource security for bundles is controlled by a BUNDLE security profile. The BUNDLE security profile also applies to BUNDLEPART, OSGIBUNDLE, and OSGISERVICE resources.

Stand-alone CICS bundles need to be made available or unavailable only if they contain application entry points. Operators with UPDATE access for the security profile for a stand-alone CICS bundle, which specifies the BUNDLE resource type and the name of the BUNDLE resource, can make the resource available or unavailable.

For resources that are dynamically created by CICS bundles, no additional CICS command security checks and resource security checks take place for those resource types, either when the resources are dynamically created at bundle install time, or when you manipulate the resources by making changes to the CICS bundles. However, CICS command security and resource security for those resource types do apply when you inquire on the dynamically created resources, or if you manipulate the dynamically created resources directly.

Security for BUNDLE resources generated by applications and platforms

When an application or platform is installed in a CICSplex, any CICS bundles that are part of the deployment are dynamically created in the appropriate CICS

regions by CICSPlex SM. Each BUNDLE resource is dynamically created, and is given a unique generated name beginning with the $ character.

You give users authority to install a platform or an application by giving them the appropriate access for the CLOUD.DEF, CLOUD.PLATFORM, and CLOUD.APPLICATION security profiles in CICSPlex SM. When you give users this authority, you also give them authority to install the dynamically created BUNDLE resources in the CICS regions. CICS command security checks and resource security checks are not made when CICS bundles are installed as part of an application or platform. Simulated CICS security checking in CICSPlex SM is also not done when CICS bundles are installed as part of an application or platform.

When you make available or unavailable, enable or disable, or inquire on, a BUNDLE resource that was dynamically created when you installed an application or platform, CICS command and resource security checks and simulated CICS security checking in CICSPlex SM apply only if you perform the action directly on the individual CICS bundle. If the CICS bundle is made available or unavailable, enabled or disabled, or inquired on, by an action that you perform on the application or platform, security checking for the application or platform applies instead. You cannot discard an individual CICS bundle directly if it was created when you installed an application or platform.

Tip: To provide security for actions on individual CICS bundles that were dynamically created when you installed an application or platform, you can set up a security profile specifying the BUNDLE resource type and the resource name `$*`. Users with UPDATE access for `BUNDLE.$*` can make available or unavailable, or enable or disable, BUNDLE resources created for platforms and applications, and users with READ access can inquire on those BUNDLE resources.

The resources that are defined inside each CICS bundle installed for an application or platform are dynamically created in the CICS regions during the installation of the dynamically created BUNDLE resource. CICS command security checks and resource security checks for the individual resource types do not take place when these resources are dynamically created in the CICS regions. However, CICS command security and resource security for the individual resource types do apply when you inquire on the dynamically created resources. You cannot directly enable, disable, or discard the dynamically created resources in the CICS regions.

If you apply security measures to individual PROGRAM resources, for applications that are deployed on platforms, secure the programs that are declared as application entry points, but do not secure other programs in the applications. The security settings that you specify for a program that is part of an application deployed on a platform apply to both public and private programs, and do not take into account the version of the application. Programs that are declared as an application entry point must have a unique PROGRAM resource name in your environment. However, if you secure programs that run at a lower level in the application, programs with the same names might be running in different applications, which can lead to unforeseen consequences. In this situation, a user might have permission to access a program that is declared as an application entry point, but not have permission to access a program that runs at a lower level in the application, because the security settings from another instance of the program name are in effect. Consider the security measures that you apply to a program that is declared as an application entry point program, as applying to the whole application.

If you used CICS bundles in earlier CICS releases, check the security permissions that you gave to users for those bundles. Depending on the way in which you set up security for CICS bundles, users with authority to take actions on individual CICS bundles might now be able to act on resources that are dynamically created as part of the installation of a bundle. Ensure that the levels of authority for BUNDLE resources are still appropriate.

For more information on security for applications and platforms and the CLOUD.DEF, CLOUD.PLATFORM, and CLOUD.APPLICATION security profiles, see Security for platforms and applications in Administering.

Variables in a CICS project

You can use variables to alter attribute values quickly and easily, which simplifies deployment to multiple environments because you can resolve the variables by using a properties file that is specific to each environment.

Using variables

Typically, attribute values in resource definitions need to be changed before being installed in to different environments. For example, a data set might have a different high-level qualifier for development, test, and production environments. You can use variables to change parts of an attribute value depending on the environment it is being deployed to, by using a `variables.properties` file that is specific to each environment.

Variables are resolved during deployment by running the CICS build toolkit with the *--resolve* option, before resources are installed in CICS. The properties file that is used to resolve a variable differs depending on whether the variable is in a stand-alone bundle or is included as part of an application.

The most reliable method of creating variables is to use the **Insert variable** or **Extract value to variable** wizard in CICS Explorer. For details, see Creating variables in the CICS Explorer product documentation.

Example usage of variables

The following examples are scenarios for using variables effectively:

- JVM server names in the CICS bundle parts `.osgibundle` `.warbundle` `.ebabundle` and `.earbundle` for Java applications, where the JVM server name is different in each environment.
- Data set names in a `FILE` and `LIBRARY` definition, where the data set name includes a qualifier for the environment.
- Specifying CEDF in a `PROGRAM` definition, where debugging is appropriate for the development environment but not for production.
- Strings in a VSAM file, where a development environment may only need a small number of strings, but production will need more to improve access times.

Variables and properties files definition

When you define variables and their associated properties files, use these rules to ensure that variable substitution can occur successfully.

Defining variables in a properties file

The following rules apply to the definition of variables in a properties file:

- Valid characters in a variable name are uppercase and lowercase letters, numbers 0 - 9, periods, underscores, and hyphens. Variable names are limited to 255 characters.
- Properties files can exist in a CICS bundle or an application binding. They must be in the root of the project, and must be named `variables.properties`.
- You can supply a separate properties file to resolve stand-alone bundles, but this option is not supported for applications. There are no restrictions on the name of a properties file when used with a stand-alone bundle.
- Properties files follow the standard format for Java properties files, and must be encoded in ISO-8859-1. For more information, see Java properties files in Java Platform, Standard Edition 7 API Specification.
- Any variable that is used in a bundle must be defined in the properties file for that bundle. When substituted into the resource definition, the value that is supplied in the bundle must result in a valid value for that attribute.
- Variable values must not refer to other variables. For example, `hlq=${prod.hlq}` is not valid.
- If a variable is defined more than once in a properties file, the last instance is used.

Referencing variables in bundle parts

The following rules apply to the reference of variables in bundle parts:

- Variable names are delimited by `${` and `}` characters.
- Variables are allowed in any CICS bundle part, including in attributes or elements of a bundle part.
- There is no limit to the number of variables that can be placed in an attribute or tag.
- Variables cannot be nested, and they must not be used as part of a CICS resource definition name.

Defining terminal resources

You can define terminal resources in two different ways, depending on the type of terminal access method that you want to use.

1. ACF/SNA LUs 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 entry from the TERMINAL and TYPETERM definitions recorded in the CSD. For guidance information about this process, see "Autoinstall" on page 241.

   You can add z/OS Communications Server 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 more information, see GRPLIST system initialization parameter.

   Each LU must also be defined to ACF/SNA in a z/OS Communications Server definition statement.

2. Non-SNA LUs 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 LUs defined in the TCT are installed as CICS resources. You must also make these LUs known to the operating system, and include a DD statement in the CICS startup job stream for each LU.

Defining z/OS Communications Server terminals

A CICS region can communicate with terminals or other regions using z/OS Communications Server services.

To use z/OS Communications Server services, you must do the following:

1. Define CICS to ACF/Communications Server with an APPL statement in SYS1.VTAMLST. For more information about defining an APPL statement for CICS, see Defining CICS regions as applications to SNA.
2. Define to z/OS Communications Server the terminal resources that CICS is to use.
3. Define to CICS the terminal resources that it is to use.

Defining CICS terminal resources to z/OS Communications Server

Each terminal, or each logical unit (LU) in the case of SNA terminals, that CICS is to use must be defined to z/OS Communications Server.

The terminals can be defined as local or remote.

Local z/OS Communications Server terminals
: can be SNA terminals connected to a channel-attached cluster controller, or they can be non-SNA 3270 terminals connected through a local control unit.

Remote z/OS Communications Server terminals
: are attached to an SNA cluster controller, which is connected through 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 z/OS Communications Server tables. z/OS Communications Server has tables describing the network of terminals with which it communicates. z/OS Communications Server uses these tables to manage the flow of data between CICS and the terminals. as nodes in the network. Each terminal, or each logical unit (LU) in the case of SNA terminals, must be defined in the z/OS Communications Server tables with a z/OS Communications Server node name that is unique throughout the z/OS Communications Server domain.

If you are using z/OS Communications Server 3.3 or later, you can define the AUTINSTMODEL name, printer, and alternate printer to z/OS Communications Server by using z/OS Communications Server MDLTAB and ASLTAB macros. These definitions are passed to CICS to select autoinstall models and printers.

For information about defining resources to z/OS Communications Server, see the *z/OS Communications Server: SNA Resource Definition Reference*.

Defining terminal resources to CICS

A given z/OS Communications Server terminal, or logical unit, can be defined explicitly in the CICS system definition file (CSD). If a terminal does not have an

explicit definition in the CSD, CICS can create and install a definition dynamically for the terminal when the terminal logs on, using the CICS autoinstall facility.

About this task

Defining terminal resources explicitly in the CICS CSD

If a z/OS Communications Server terminal, or logical unit is defined explicitly in the CSD, the terminal has a TERMINAL name and a NETNAME, which is the same as the z/OS Communications Server node name. Terminals defined in this way have terminal entries installed at CICS startup.

Using the CICS autoinstall facility to dynamically create and install definitions for terminals

CICS can autoinstall terminals by reference to TYPETERM and model TERMINAL definitions created with the **AUTINSTMODEL** and **AUTINSTNAME** attributes. For information about TYPETERM and TERMINAL definitions, see "Autoinstalling model terminal definitions" on page 265 and Model TERMINAL definitions in group DFHTERM.

If you use autoinstall, you must ensure that the CICS resource definitions correctly match the z/OS Communications Server resource definitions. For programming information about z/OS Communications Server logmode definitions and their matching CICS autoinstall model definitions, see Coding entries in the VTAM LOGON mode table.

If you specify the system initialization parameter TCTUALOC=ANY, CICS stores the terminal control table user area (TCTUA) for z/OS Communications Server terminals above the 16 MB line if possible.

Defining the terminal shutdown time limit

You can specify a time limit within which all SNA LUs used by 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 topics.

Choosing an appropriate value for TCSWAIT

The value that you specify on the **TCSWAIT** system initialization parameter must be large enough so that under normal circumstances all SNA LUs and connections shut down 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
```

If you do not want a time limit (that is, you assume that all terminals never hang), specify NO for the **TCSWAIT** system initialization parameter. For more information about this parameter, see TCSWAIT system initialization parameter.

Note: VTAM is now z/OS Communications Server for SNA.

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.

Defining sequential (BSAM) devices

You can use a pair of input and output sequential data sets to simulate a terminal to CICS. For example, you might do this to test an application program before the intended terminal becomes available.

To do so, code the following DFHTCT TYPE= macros:

```
DFHTCT TYPE=INITIAL,
       ACCMETH=(NONVTAM)    defining the access
                            method
```

Define the following macro instructions contiguously:

```
DFHTCT TYPE=SDSCI,
       DSCNAME=isadscn,     defining the input
       DDNAME=indd, ...      data set
DFHTCT TYPE=SDSCI,
       DSCNAME=osadscn,     defining the output
       DDNAME=outdd, ...     data set
DFHTCT TYPE=LINE,
       ISADSCN=isadscn,
       OSADSCN=osadscn, ...
DFHTCT TYPE=TERMINAL,
       TRMIDNT=name, ...
```

The two data sets defined by the DFHTCT TYPE=SDSCI macros simulate a CICS terminal known by the name specified in the TRMIDNT operand of the DFHTCT TYPE=TERMINAL macro. The DSCNAMEs of the input and output data sets must be specified in the ISADSCN and OSADSCN operands of the DFHTCT TYPE=LINE macro respectively.

You must code a DD statement for each sequential data set defined by an SDSCI macro. The DD name on the DD statement must be the same as the name coded on the DDNAME parameter (or, by default, on the DSCNAME parameter) of the SDSCI macro. For example, you could use the following DD statements for sequential input and output:

```
//CARDIN  DD  *,DCB=BLKSIZE=80
      .
    Statements containing valid transactions
      .
/*
//PRINTER DD  SYSOUT=A,DCB=BLKSIZE=132
```

This example of an I/O combination simulates a terminal to a CICS application program. There is an example of the SDSCI statements supporting this CARDIN/PRINTER combination in the copybook DFH$TCTS, which is defined in the sample TCT, DFHTCT5$. DFH$TCTS is supplied in CICSTS54.CICS.SDFHSAMP. Input to the application program is submitted

through the input stream (CARDIN), and output to the terminal is sent to the output stream (PRINTER). If the BLKSIZE parameter is defined in the TCT for the data set, you can omit it from the JCL. However, if it is not defined in the TCT, the block size defaults to 0, and if you also omit it from the DD statement for the data set, you get message IEC141I 013-34. There are other examples of DD statements for I/O sequential data sets in some of the CICS-supplied installation verification procedures. You can find them in CICSTS54.CICS.SDFHINST after installation.

You can also use two DASD data sets to simulate a terminal. You must code a DD statement for each data set defined by an SDSCI macro, and the DD name on the DD statement must be the name coded on the DDNAME (or DSCNAME) parameter of the SDSCI macro.

For example, you might code:

```
//DISKIN1  DD DSN=SIMULATD.TERMINAL.IN,
//         UNIT=3380,DISP=OLD,VOL=SER=volid
//DISKOT1  DD DSN=SIMULATD.TERMINAL.OUT,
//         UNIT=3380,DISP=OLD,VOL=SER=volid
```

Input from this simulated terminal is read from the DISKIN1 data set. Output to the terminal is written to the DISKOT1 data set.

Each statement in the input file (from CARDIN or DISKIN1 in the examples used earlier), 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 in the pregenerated system. You can redefine this for your installation on the EODI system initialization parameter; see Specifying CICS system initialization parameters for details.

Terminating

End-of-file does not terminate sequential input. Use **CESF GOODNIGHT** as the last transaction, to close the device and stop reading from the device.

Otherwise, CICS invokes the terminal error program (DFHTEP), and issues the messages in Table 22 at end-of-file on the sequential device.

Table 22. 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

Using **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 messages DFHTC2507 and DFHTC2500 at end-of-file. However, the device remains 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**. This puts the sequential terminal into RECEIVE status and terminates reading from the terminal. In this case, it is recommended that you code a PLT program to change the status of the terminal to TRANSCEIVE.
- 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 System commands. For programming information about writing post initialization-phase programs, see Writing initialization programs.

If you use BSAM devices for testing purposes, the final transaction to close down CICS could be **CEMT PERFORM SHUT.** The receive-only status is recorded in the warm keypoint at CICS shutdown. This means that on a subsequent warm start of CICS, the terminal is still in RECEIVE status and CICS does not then read the input file.

If you use the **CEMT PERFORM SHUT**, perform a cold or initial start of the CICS region to read the input file. This assumes that recoverability of resources is not important to you.

Defining console devices

You can operate CICS from a *console device*. A console device can be a locally-attached system console, a TSO user defined as a console, or an automated process such as NetView®.

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. (You cannot define consoles in the TCT.)

Suitably authorized TSO users can enter MODIFY commands from terminals connected to TSO. To enable this, define the TSO user as a console device in the CSD.

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.

Defining console devices to CICS

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

Each console can be defined explicitly, or you can define autoinstall model definitions, and use the CICS terminal autoinstall facility for consoles to install consoles automatically.

If want to use the console autoinstall facility, specify `AICONS=YES|AUTO` as a system initialization parameter, and define TERMINAL model definitions that specify AUTINSTMODEL(YES) and the AUTINSTNAME attribute.

System consoles:

System consoles are defined to MVS in the SYS1.PARMLIB library, in a CONSOL*nn* member, that defines attributes such as NAME, UNIT, and SYSTEM.

The name is the most significant attribute, because it is the name that CICS uses to identify the console. The name is passed to CICS on an MVS **MODIFY** command. Although consoles also have a numeric identifier, this is allocated by MVS dynamically during IPL, and its use is not recommended for defining consoles to CICS.

For information about defining console devices to MVS, see the *z/OS MVS Initialization and Tuning Reference*.

For information about defining MVS consoles to CICS, see "Defining MVS consoles to CICS."

TSO users as consoles:

TSO users that issue commands to CICS, using either the TSO **CONSOLE** command or SDSF, do not require MVS definitions as consoles in the CONSOL*nn* member. MVS activates a console automatically using the user's TSO/E user ID as the console name

The TSO user issuing the **CONSOLE** command can use the NAME option to specify a console name different from the TSO user ID.

To communicate with a CICS region from TSO or SDSF, you must install a CICS console definition that specifies the TSO user ID (or the name specified on the console command) as the console name.

For information about the TSO **CONSOLE** command, see the *z/OS TSO/E System Programming Command Reference*.

For information about defining TSO users to CICS, see "Defining TSO users as console devices" on page 311.

Defining MVS consoles to CICS:

To use an MVS console as a CICS master terminal, you either define it to CICS explicitly by a terminal definition entry in the CSD, or use the CICS console autoinstall facility.

Each console you define is identified on the TERMINAL definition by the CONSNAME(*name*) attribute. CICS no longer supports the CONSOLE(*number*) attribute. Identify a console device attached to an MVS in a sysplex by its name, using the CONSNAME attribute.

For an example of the DEFINE command required to define a console, see Figure 57.

```
//DEFTERM  JOB (accounting information),MSGCLASS=A,
//         MSGLEVEL=(1,1),CLASS=A,NOTIFY=userid
//CONSDEF  EXEC PGM=DFHCSDUP
//STEPLIB  DD DSN=CICSTS54.CICS.SDFHLOAD,DISP=SHR
//DFHCSD   DD DSN=CICSTS54.CICS.DFHCSD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
*
*  Define a console for CICS
DEFINE TERMINAL(trmidnt)  GROUP(grpname)  TYPETERM(DFHCONS)
       CONSNAME(consname)  DESCRIPTION(MVS CONSOLE consname)
*
*  Define a TSO user as a console device for CICS
DEFINE TERMINAL(trmidnt)   GROUP(grpname)     TYPETERM(DFHCONS)
       CONSNAME(tsouser)   DESCRIPTION(TSO USER tsouser)
       USERID(tsouser)
*
*  Define an AUTOINSTALL model definition for a console device
DEFINE TERMINAL(autc)     GROUP(grpname)     TYPETERM(DFHCONS)
       CONSNAME(console)  DESCRIPTION(Autoinstall model for a console)
       USERID(*FIRST)     AUTINSTNAME(name) AUTINSTMODEL(YES)
*
ADD GROUP(grpname) LIST(yourlist)
*
LIST   LIST(yourlist)  OBJECTS
/*
//
```

Figure 57. Defining consoles and a TSO user in the CSD using DFHCSDUP

Defining TSO users as console devices:

You can define TSO users as console devices to CICS using either an explicitly-defined TERMINAL definition for each TSO user, or use the console autoinstall facility.

To define a TSO user as a console, specify the console name used by the TSO user on the CONSNAME attribute of the DEFINE TERMINAL command. By default, the console name is the user's TSO user ID. You are recommended to define consoles to CICS with preset security by using the USERID operand, so that the TSO user does not have to sign on using the CESN transaction. Otherwise, the TSO user's CICS signon password is displayed when entered for the CESN transaction.

For an example of the DEFINE command required to define a TSO user, see Figure 57.

For information about defining consoles (or terminals) with preset security, see the Security facilities in CICS.

Note: Substitute your own values for the operands that are shown in italics in the DEFTERM job shown in Figure 57.

AUTINSTMODEL(YES)
Specifies whether this TERMINAL definition can be used as a model for autoinstall purposes.

AUTINSTNAME(*name*)
The name assigned to this autoinstall model definition, and by which the model is known to the autoinstall control program.

CONSNAME(*consname*)
A unique 8-character console name, which corresponds to the NAME parameter in the CONSOL*nn* PARMLIB member that defines the console, or matches the console name used by a TSO user.

You must define CONSNAME even for an autoinstall model.

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

LIST(*yourlist*)
The startup group list containing the group in which you have defined the console definitions. If your new group list does not include the required CICS-supplied resources as well as your own, specify DFHLIST and *yourlist* on the GRPLIST system initialization parameter of your CICS startup job.

TERMINAL(*trmidnt*|*autc*)
A unique 4-character terminal identifier *trmidnt* as the name by which CICS is to identify the console in the TCT terminal entry (TCTTE), or a dummy name *autc* in the case of an autoinstall model definition.

USERID(*tsouser*)
The CICS preset security userid to be used to sign on this console device.

If you have defined a console device in your CSD as CONSNAME(INTERNAL), you can use it to issue commands using MVS job control language. It is also used by authorized programs that use the MGCR macro to issue MVS commands.

Having defined the console devices in the CSD, 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. Include the group list that contains the resource definitions on the **GRPLIST** system initialization parameter in the CICS startup job.
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 initialize the CSD with the DFHCSDUP INITIALIZE command, does not include any resource definitions for console devices. However, the CSD is initialized with 2 groups that contain console definitions:

DFH$CNSL
This group contains definitions for three consoles. The group is intended for use with the installation verification procedures and the CICS-supplied sample programs. You can add this to your own group list, and alter the definitions to define your own console devices.

DFHTERMC
This group contains a single definition of an autoinstall model definition for an MVS console.

If you decide to create new terminal definitions for your console devices, you can 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.

Defining z/OS Communications Server persistent sessions support

If you choose to run a CICS region with z/OS Communications Server persistent sessions support, terminal users can have their sessions recovered and continue working in the event of a CICS or z/OS Communications Server failure.

About this task

Use the CICS system initialization parameters **PSTYPE** and **PSDINT** to set up persistent sessions support for a CICS region, and use options on the CONNECTION, TYPETERM, and SESSIONS resource definitions to customize the user experience for terminal users in the event of a recovery.

The default settings for the **PSTYPE** and **PSDINT** system initialization parameters mean that persistent sessions support is available to the CICS region, but that it is not being exploited. Recovery with z/OS Communications Server persistent sessions explains what happens when you exploit persistent sessions support, and why you might want to run a CICS region without persistent sessions support.

The default values for **RECOVOPTION** and **RECOVNOTIFY** in TYPETERM definitions are SYSDEFAULT and NONE respectively but when used in this combination with persistent sessions can, in some circumstances, result in a recovered terminal becoming unusable. This condition can continue until a task is automatically initiated at the terminal, for example using **EXEC CICS START TRANSID(xxx) TERMID(yyy)**. While use of RECOVNOTIFY(NONE) may be appropriate for devices such as printers different values should be considered for user terminals.

If you want to change the type of persistent sessions support for an existing CICS region, you need to shut down the region and perform a cold start with the new settings. You cannot change between single-node persistent sessions support, multinode persistent sessions support, and no persistent sessions support while CICS is running. Because the persistence characteristics of a bound session are established when the session is created, you need to force an unbind and to rebind the sessions to acquire the correct persistence characteristics for the new level of persistent sessions support requested by CICS. The most straightforward way to achieve this is to carry out a cold or initial start of the CICS region.

You can change the persistent sessions delay interval while CICS is running, but the changed interval is not stored in the CICS global catalog, and therefore is not restored in an emergency restart.

If you do want to use persistent sessions support for a CICS region, follow these steps:

Procedure

1. Use the **PSTYPE** system initialization parameter to specify an appropriate level of persistent sessions support.
   a. If you have z/OS Communications Server V4R4 or later, in a Parallel Sysplex with a coupling facility, specify MNPS for multinode persistent

sessions support. This level of support enables recovery in the event of a z/OS Communications Server, z/OS, or CICS failure.

b. If you do not have suitable facilities for multinode persistent sessions support, specify SNPS for single-node persistent sessions support. This level of support enables recovery in the event of a CICS failure, but not in the event of a z/OS Communications Server or z/OS failure.

2. Use the **PSDINT** system initialization parameter to specify a suitable persistent sessions delay interval. This value determines for how long z/OS Communications Server holds sessions in a recovery-pending state. The interval you specify must be able to cover the time from a CICS failure to the time when the z/OS Communications Server ACB is opened by CICS during a subsequent emergency restart.
3. Choose and set an appropriate value for the RECOVOPTION option of the SESSIONS and TYPETERM resource definitions used in the CICS region. Typically, this value can be the default SYSDEFAULT, which makes CICS select the optimum procedure for recovering sessions. This setting means that in the event of a recovery, the terminal user can clear the screen and continue to enter CICS transids.
4. Choose and set an appropriate value for the RECOVNOTIFY option of the TYPETERM resource definitions used in the CICS region.
   a. If you want a successful recovery to be transparent to terminal users, specify NONE.
   b. If you want to display a message on the screen to say that the system has recovered, specify MESSAGE.
   c. If you want to start a transaction at the terminal, such as the good-morning transaction, specify TRANSACTION.
5. If you are working with an existing CICS region, perform a cold start of the CICS region to implement the changes to persistent sessions support.

Resource definition installation

When a resource definition is installed, information about the resources is used to construct the data structures that represent the resource in the CICS address space.

What happens when CICS is initialized

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

Initial or cold start

During an initial or cold start, CICS creates system tables by installing groups named in the list or lists named by the GRPLIST system initialization parameter.

If you installed a group with the INSTALL command during the previous CICS execution, you must add its name to a list if you want it to be installed during a cold start.

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

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

Warm or emergency start

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

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

- "What happens at CICS restart" on page 250
- "Recovery and restart for connection autoinstall" on page 260
- "Program autoinstall and recovery and restart" on page 264

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

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

What happens when you use the INSTALL command

Most resource definitions can be installed in groups or individually, and are committed at the individual resource level. However, some SNA LU control resource definitions must be installed in groups and are committed in *installable sets*.

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

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

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

All other resource types are committed individually, and not in installable sets. For these resources, the effect of a partially successful group INSTALL is to leave the resources that were added in a committed state.

For the installation of installable sets and for individual definition, an INSTALL may not be successful for one of two reasons:

1. A resource definition could not be installed because it is currently in use.
2. A system failure occurred during installation.

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

How to install a limited number of data definitions

If you want to install only a few new or changed definitions, install single resources. Use of the single-resource INSTALL eliminates the problems of a partial INSTALL caused by a failure.

Note that the single-resource INSTALL of some CONNECTIONs and SESSIONS is not possible.

However, if you want to change or add a larger number of definitions, you might prefer to install a new group. In that case, the following considerations apply:

- When you install a group containing an updated definition of an existing resource, the installation fails if the resource is being used at the time. Make sure that none of the resources in a group is in use before trying to install the group.
- Installation is a two-stage process: any existing definition for the resource must be "deleted" from the system tables before a definition can be installed. This can result in more than one message if the "deletion" fails and causes the installation to fail.
- If you have several CICS systems that share the same CSD file, you must be careful not to install a group of resources in the wrong system.

Duplicate resource definition names

For all resource types except TDQUEUE and FILE resources, if two groups in a list contain resource definitions of the same name and of the same resource type, CICS uses the definition in the group that is later in the list.

- For TDQUEUE resource definitions, the first definition in the list is used.
- For FILE resource definitions for local files, if the file is defined as ENABLED, the later installation of a duplicate fails. However, if the file is defined as DISABLED, the later installation of a duplicate succeeds. For FILE resource definitions for remote files, which are not defined as ENABLED or DISABLED, the installation of a duplicate always succeeds.

If you INSTALL a group while CICS is active, the resource definitions in the group override any of the same type and name already installed. When an existing resource definition is replaced in this way, the statistics associated with the old resource definition are transferred to the new definition. If a PROGRAM definition is replaced, the program is relocated on the library and loaded when the new definition is referenced for the first time. In effect, the new definition implies a NEWCOPY operation. The same rules apply to map sets and partition sets.

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

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

Installed resources that are defined in CICS bundles cannot be overwritten by resource definitions with duplicate names. When the bundle part containing a resource has been installed, if you try to install a duplicate resource using another CICS bundle, or using RDO, EXEC CICS CREATE commands or other methods, the request is rejected. You also cannot overwrite installed resources with a duplicate resource that is defined in a CICS bundle. When you attempt to install the CICS bundle, if a resource with the same name and resource type exists, the bundle part containing the resource is not installed, and the CICS bundle is placed in an unusable state. Resources that are defined in CICS bundles are therefore protected against accidental overwriting by duplicate resource names.

When you define a private resource in a CICS bundle that is packaged and installed as part of an application, the resource name does not have to be unique in your installation. You can use this facility to avoid resource name clashes between applications that were developed independently, but used the same resource names. The requirement for unique resource names for the supported CICS resources can be removed by managing the resources as part of applications deployed on a platform. You can use this process to assist with server consolidation.

For more information about private resources for applications, see .

Use the resource signature to identify the source of duplicate resource definitions that replaced existing installed resources. The resource signature shows when, how, and by whom each resource was defined, changed, and installed. The resource signature is displayed in the CICS Explorer views, on CEDA and CEMT panels, in CICSPlex SM BAS and Operations views, by **EXEC CICS INQUIRE** commands, and in DFHCSDUP reports.

Chapter 5. Setting up a platform

A platform provides a simple layer of abstraction to decouple applications from the underlying complexities of a CICS topology. You can set up a platform in a CICSplex to provide a managed environment for application deployment.

Before you begin

You must use CICSPlex SM to provide a single system image of your CICS regions. Other features of CICSPlex SM, such as Business Application Services (BAS) and workload management, are not required.

About this task

To create a platform, you use the CICS Cloud perspective in CICS Explorer. This perspective provides the views to manage the lifecycle of platforms, from creating the description in XML, to deploying and managing the resources in CICS regions.

Because a platform is described in XML, you create it in a platform project and deploy it to zFS. It is therefore important to set up the directory structure in zFS for each platform that you want to create in your CICSplex. For more information, see “Preparing zFS for platforms” on page 324.

To set up a platform, follow these steps:

Results

You have a platform running in a CICSplex.

What to do next

After you set up a platform in your CICSplex, you can deploy packaged applications on the platform. You can also add further quality of service by deploying policies to control the environment. These policies can apply to every region in the platform, or to particular applications.

Platform design

Before you create a platform, review your environment and identify how you might map regions to a platform. Alternatively, identify what new CICS regions you might create specifically to use in a platform.

Consider the following items in the design for your platform:

- Applications that you want to deploy on your platform.
- Policies that you want to deploy on your platform.
- Individual resources that you want to deploy as CICS bundles on your platform.
- Existing CICS regions and system group definitions (CSYSGRPs) that you want to adopt as part of the platform.
- New CICS regions and region types that you want to create in the platform.

© Copyright IBM Corp. 1989, 2017

If your capacity requirements increase or decrease after you have designed and installed your platform, you can use the CICS Explorer to add further CICS regions to your active platform or remove CICS regions from it.

Designing region types

A region type is a logical grouping that collects together a number of CICS regions that share common characteristics, and enables them to be managed as a unit in a platform. Look for common characteristics of CICS regions that you want to manage as a group, and use these characteristics to divide the CICS regions in the platform into suitable region types. You could group your CICS regions by using region types to meet functional, geographical, or legal requirements, as in these examples:

- Terminal owning region (TOR), File owning region (FOR), Application owning region (AOR): CICS function type
- Production, Test, Development: business function type
- Payroll, Personnel, Accounts: business processing function type
- United Kingdom, Asia, Europe: geographical areas by continent, country, state, county

For some examples of platform architecture, see Platform examples.

You can design and create new region types to use in the platform. Alternatively, you can choose to adopt existing system group definitions (CSYSGRPs) as region types in the platform. A single platform can include both created region types and adopted region types. When you use created region types, the CICSPlex SM topology for those CICS regions is created by the platform and is associated with the lifecycle of the platform. When you use adopted region types, you set up and maintain the CICSPlex SM topology for those CICS regions independently of the lifecycle of the platform.

If a CICS region has more than one purpose in the platform, you can include it in more than one region type, as a shared region. You can also share CICS regions with region types in other platforms. If your platform design involves sharing CICS regions between region types, be aware that a created region type can only contain created regions, and an adopted region type can only contain adopted regions. For more information about region sharing, see Sharing regions between region types in a platform.

A CICS TS Version 5.1 region connected as a MAS to a CICS TS Version 5.1, 5.2 or 5.3 CMAS can be part of a platform together with CICS TS Version 5.2 and CICS TS Version 5.3 regions connected as a MAS to a CICS TS Version 5.2 or CICS TS Version 5.3. It is possible to install applications created with CICS TS Version 5.2 or Version 5.3 on platforms that include CICS TS Version 5.1 regions. However, installation of CICS TS Version 5.2 or Version 5.3 applications into CICS TS Version 5.1 regions in a platform has the following limitations:

- Private resources for applications, such as private PROGRAM or LIBRARY resources, are not supported in CICS TS Version 5.1 regions, and are not created in those regions. If multiple versions of the application are installed on the platform, resource name clashes can therefore occur in the CICS TS Version 5.1 regions. In this situation, the duplicate resource fails to install in the CICS TS Version 5.1 regions with message DFHAM4950 or DFHAM4834 being issued, and the CICS bundle for the new application version cannot be enabled in those regions.

- CICS bundles with the same ID and version are not supported in CICS TS Version 5.1 regions. If multiple versions of the application are installed on the platform that include CICS bundles with the same ID and version, the CICS bundles fail to install in the CICS TS Version 5.1 regions, and the regions will issue message DFHAM4952. In this situation, the application is in INCOMPLETE status, and it cannot be enabled in the CICS TS Version 5.1 regions.

Because of these limitations, although you can install multiple versions of an application created with CICS TS Version 5.2 or Version 5.3 on a platform that includes CICS TS Version 5.1 regions, it is likely that the later versions of the application will fail to install in the CICS TS Version 5.1 regions. To avoid installation errors, only include CICS TS Version 5.2 or Version 5.3 regions in the region types where you are installing multi-versioned applications.

Designing new CICS regions for a platform

When you design a platform, you can include new CICS regions in created region types to meet the exact requirements of the applications that you want to deploy on your platform. You can use created regions to provide extra functions that supplement existing CICS regions that you plan to adopt as part of the platform. Or you can design a platform that consists entirely of created regions.

In your platform design, consider the characteristics that are required for the CICS regions in each created region type. Created region types can specify the properties of the CICS regions that they contain. You can clone certain region attribute values for all the CICS regions in a region type by specifying the attributes at a region type level. Only CICS regions that can accept the required settings can be part of that region type. The setting in the CICS region can be the same as the setting for the region type, or the setting can be absent in the CICS region, in which case it is supplied from the setting for the region type. However, the setting in the CICS region cannot conflict with the setting for the region type. Shared regions must be able to accept settings from all the region types in which they are included.

You can specify the following region attribute values at a region type level:

Eligible as Routing Region (WLMSTATUS attribute)
: Whether or not this CICS region is to participate in its associated workload as a routing region when the CICS region is started.

Eligible as Target Region (DYNROUTE attribute)
: Whether or not this CICS region is to be active as a target region and accept work for the workload for which it is a target at CICS startup.

Enable BAS Install (AUTOINST attribute)
: Whether resources associated with the CICS region through a resource description should be automatically installed when the MAS connects to the CMAS.

BAS Install Failure Action (AINSFAIL attribute)
: The action to be taken in the event of a BAS install failure.

If the architecture of your platform requires that all the CICS regions in a region type have particular capabilities or restrictions in these areas, specify the appropriate values at a region type level when you are setting up the region type. If a created region type has no special requirements for an attribute, do not specify any value for that attribute, so that any setting is allowed in the CICS regions. When you specify a region attribute value at a region type level, that attribute

value is locked and cannot subsequently be changed in a CICS region that is part of the region type.

Mapping existing CICS regions to a platform

When evaluating your existing systems for possible candidates to become a platform, look for any top-level groups that contain two or more system group definitions (CSYSGRPs) and multiple CICS regions. The top-level group could potentially be a good candidate to be re-implemented as a platform, and the CSYSGRPs could potentially be good candidates for region type adoption. You can more easily manage and deploy resources and applications to the CICS regions in the platform by packaging parts of your existing topology as a platform.

Each CICS system group (CSYSGRP) that you include as part of a platform must meet the following requirements:

- The group has not already been adopted by a platform that is already installed. If the group is already associated with a platform, it cannot be adopted as a region type.
- The group does not contain any subgroups.
- The group will not require modification (for example, a group that is involved in WLM or RTA). Platforms require a lock on the groups that are used as region types.
- All the CICS regions in the group have the CICSPlex SM system parameter MASPLTWAIT(YES) specified. MASPLTWAIT(YES) is also required for Business Application Services. This parameter is required to automatically install resources for an application or platform when the CICS region is initialized.

If you have CICS regions in a CSYSGRP that does not meet these requirements, and you want to use the CICS regions as part of the platform, add their system definitions (CSYSDEFs) to a new CSYSGRP that you create specifically for the platform.

CICSPlex SM data repository (EYUDREP) size

If you are planning a large-scale deployment, check that the size of your CICSPlex SM data repository is adequate. The data repository is the VSAM data set where system configuration and definition data for CICSPlex SM is stored. Each CMAS has its own data repository.

The resources for platforms and applications are managed from the data repository, not from the CICS CSD, so the data repository needs to have enough space for the definitions for your platform and the applications and bundles deployed on it. You can determine the current size of the data repository by using the LISTCAT function of the IDCAMS utility.

If you want to expand the data repository, use the REPRO function of the IDCAMS utility. An example of the JCL to do this is in the EYUJXDRP member of the CICSTS51.CPSM.SEYUSAMP library. In that JCL, on the RECORDS(xx,yy) statement, specify a primary (xx) and a secondary (yy) value that are appropriate for your environment. The initial values are 500 and 3000.

Sharing CICS regions between region types in platforms

When you are designing your platform, you might decide to share some CICS regions between multiple region types, either within a platform or between multiple platforms.

Sharing regions between region types can be a valuable configuration and easy to set up within a platform. Although there are more considerations than keeping everything separate, these can be made less severe with planning.

There are various reasons why you might want to share CICS regions. For example, in a platform on your development CICSplex, you may want to use a single CICS region that is shared by all of your region types. Using a single CICS region that is shared by all of your region types simplifies new development because you do not have to configure, manage and maintain multiple CICS regions. You might also want to share CICS regions in a production environment, where you have multiple platforms that share a single set of TORs or an FOR.

You can share CICS regions between region types either when you are setting up your platform, or in an installed and active platform. A created region type can only contain created CICS regions, and an adopted region type can only contain adopted CICS regions.

In created region types, you can clone certain region attribute values for all the CICS regions by specifying the attributes at a region type level, in the platform project. "Platform design" on page 319 lists the options that you can set at this level, which relate to CICSPlex SM workload management and BAS install. Shared CICS regions must be able to accept these settings from all the region types in which they are included. For example, you might create a region type that specifies that its CICS regions must be eligible as routing regions for workload management. This region type cannot share a CICS region with a region type where the CICS regions are required to be ineligible as routing regions. However, it can share a CICS region with a region type where the CICS regions are also required to be eligible as routing regions, or where that setting has not been specified at the region type level.

Whether you are sharing CICS regions within a platform or between multiple platforms, care must be taken to ensure that conflicts between specific CICS resources do not exist. Bundle installation for platforms and applications only checks for conflicts at the CICS bundle level. If two different bundles are installed into the same region and both create the same resource, the duplicated bundle part becomes unusable.

If multiple platforms are sharing a group of CICS regions, and both platforms require the same resources to be installed, only one of the platforms should attempt to install those bundles. This platform should be installed and discarded first. It is possible for both platforms to install the same set of bundles, but the results of this approach are uncertain. The platform install will state that no bundles were installed for the second platform. The second platform will show a platform ENABLESTATUS of INCOMPLETE, and bundle installation failures will also be recorded in the shared region. With this second approach, you also have to be careful about discarding a platform: because only one platform has the bundles installed, you must make sure that the other platform has the bundles installed that it needs to function. The worst case scenario with this technique is where both platforms are installed at the same time. In this case each platform may install a subset of the bundles and so both platforms will show a platform ENABLESTATUS

of INCOMPLETE. However, even in this scenario, the platform will correctly clean up all of the bundles that are left behind when the platforms are discarded.

Preparing zFS for platforms

Before you can create and deploy a platform, you must configure your platform home directory in zFS. Create a dedicated file system, set up the file system security, and set up FTP security for access from CICS Explorer.

Procedure

1. Create a z/OS UNIX file system data set to use as the zFS platform home directory. The purpose of this is to create a dedicated file system for use by all CICS regions in the platform.

   Note: The default platform home directory is `/var/cicsts/CICSplex/platform1`, where *CICSplex* is the name of the CICSplex where the platform will be installed, and *platform1* is the name of your platform. Keep this default as a best practice. If you need to use a different directory as the platform home directory, you must change the platform bundle to specify the alternative directory name using the platform descriptor editor after you create the CICS Platform project.

   a. If you are using non-shared zFS, mount the data set onto `/var` as `/var/cicsts`, as a read-write file system.

   b. If you are using a shared file system in a multi-system (LPAR) environment, mount the data set onto the root file system (`/`) as `/cicsts`, and then for each system that requires access, create a symbolic link from `/var/cicsts` to the shared `/cicsts` directory.

   c. If you have a multi-system or cross-sysplex environment where file systems cannot be shared between all the systems, duplicate the structure that you set up for the platform home directory in each of the zFS file systems. You will need to ensure that the contents of the platform home directory are duplicated to each of the zFS file systems whenever you export a platform, application, or CICS bundle. You can repeat the export process in CICS Explorer and select the appropriate z/OS connection for each individual file system.

   d. If the directories do not already exist, create the `/var/cicsts/CICSplex` and `/var/cicsts/CICSplex/platform1` subdirectories. If you are using CICS Explorer, these directories are created for you. If you are not using CICS Explorer, you must create the */CICSplex* and */platform1* directories.

2. Set up file system security. This file system security ensures that all CICS regions in the platform, including the CICSPlex SM CMAS regions, can read the bundle files in the platform home directory.

   a. Change the owner of the directories in `/var/cicsts` to the user ID that is used to create the bundle files.

   b. Change the group ownership of the directories in `/var/cicsts` to a group that all the CICS regions in the platform belong to.

   c. Give the owner of the directories read, write, and execute permissions, and give the group read and execute permissions. For example, `rwxr-x---`.

   d. Optional: If write access is required by multiple administrator user IDs, or read access is required by different groups, you can use UNIX System Services (USS) access control list (ACL) entries to add additional group or owner permissions. You can achieve this by activating the **FSSEC** resource class and by using the **setfacl** command.

3. Set up FTP security. This level of security ensures that bundles exported from CICS Explorer can be written to the platform home directory on zFS, and read by all the CICS regions in the platform.
   a. Set the file mode creation mask for the z/OS FTP daemon to ensure that the owner has write permissions and the group has read permissions. To configure this, use the **UMASK** statement in the `FTP.DATA` configuration file.
   b. Optional: If you are also using ACL entries to control security, ensure that the default ACLs are inherited from the zFS platform home directory, for example `/var/cicsts/CICSplex/platform1`, where *CICSplex* is the name of your CICSplex and *platform1* is the name of your platform.

Results

Your zFS environment is now configured with the correct directories and permissions.

What to do next

You can now create a platform bundle using a CICS Platform project in CICS Explorer.

Creating a platform

Create a CICS Platform project in the CICS Explorer to define a platform bundle, which is a type of management bundle that describes a platform. The platform bundle specifies the region types for the platform.

Before you begin

Before you create your CICS Platform project, review your environment and identify how you might map your existing CICS regions to a platform. Alternatively, identify what new CICS regions you might create specifically to use in a platform. For more information, see Platform design.

Set up a platform home directory for the platform in zFS before you create the CICS Platform project. The best practice is to use the default platform home directory, `/var/cicsts/CICSplex/platform1`, where *CICSplex* is the name of the CICSplex where the platform will be installed, and *platform1* is the name of your platform. If you need to use a different directory as the platform home directory, you can change to the alternative directory name using the platform descriptor editor after you create the CICS Platform project. For instructions to configure the platform home directory, see Creating a platform home directory in zFS.

If you want to deploy CICS bundles at the level of your platform, you can create these ready to add them when you create your CICS Platform project. For example, you can deploy a CICS bundle containing a resource that is required in all the CICS regions in the platform, or a policy that applies to multiple applications deployed on the platform. If you do not yet have any applications, resources, or policies set up for your platform, you can add CICS bundles to the platform after you create it, or at any time after you deploy it. For instructions to create a CICS bundle, see Defining CICS bundles.

About this task

A platform includes one or more region types. Region types are used to classify and contain CICS regions according to their type. For example, all CICS regions that handle connections to DB2 could belong to the same group. You can share CICS regions between region types in the same platform or with region types in other platforms.

You can create new region types to use in the platform. You can also choose to adopt existing CICS system groups (CSYSGRPs) as region types in the platform. A single platform can include both created region types and adopted region types.

Created region types can specify the properties of the CICS regions that they contain. You can clone certain region attribute values for all the CICS regions in a region type by specifying the attributes at a region type level. Only CICS regions whose definitions have the same values specified for those attributes, or have no values specified for those attributes, can be part of that region type. For a list of the attributes that can be specified at a region type level, see Platform design.

If you want to re-implement an existing CICSPlex SM topology as a platform, you can adopt an existing CICS system group (CSYSGRP) as an adopted region type. Each CSYSGRP that you adopt as part of a platform must meet the requirements listed in Platform design. For example, the group must not contain any subgroups.

For detailed help about the following steps to create a platform, see Working with platforms in the CICS Explorer product documentation.

Procedure

1. Create a new CICS Platform project in the CICS Explorer. Enter a name for the project, and a name and description for the platform itself. The project location specifies where the CICS Platform project is saved in your local workspace.
2. Add one or more region types for the platform. Name each region type, then specify whether it is a created region type with a new system group, or an adopted region type using an existing system group. For a created region type, enter a name for the CICS system group (CSYSGRP) that will be created for the region type. To add an existing CICS system group as an adopted region type, you must have a connection to CICSPlex SM.
3. Optional: Specify any CICS bundles that you want to deploy with the platform, then select the region types where they will be deployed. If you do not have any CICS bundles ready to deploy with the platform, skip this stage.
4. Click **Finish** to create the project. When you finish the wizard, the platform project is created in the CICS Explorer workspace.
5. Use the platform descriptor editor in the CICS Explorer to edit the CICS Platform project to check and complete your specifications for the platform bundle. The platform descriptor editor opens automatically after you create a platform project. To open the platform descriptor editor at a later time, double-click any of the `.xml` files for the platform bundle, except the `manifest.xml` file.
   a. If you need to use a different directory instead of the default platform home directory, browse for the home directory that you have set up and specify it as the platform home directory.
   b. Verify your region types, and add or remove created and adopted region types as required. After a platform is installed and active, you can add and

remove individual CICS regions in region types. However, you cannot modify the region types in an installed platform, so finalize your region types before installing the platform.

c. For each of the created region types in your platform, specify any required settings that must apply in all the CICS regions in the region type. Only CICS regions that can accept the required settings can be part of that region type. If your created region type has no special requirements for an attribute, do not specify any value for that attribute, so that any setting is allowed in the CICS regions.

d. For each of the created region types in your platform, add one or more CICS region definitions for CICS regions that will be part of the region type. A default CICS region definition is provided, which you must replace with a real definition for a CICS region in the created region type. Specify the basic properties for each CICS region, and the created region types where the region will be included.

e. Verify the CICS bundles that are to be deployed with the platform. Add or remove bundles as required, and verify or change the region types where each CICS bundle is to be deployed. You can deploy further CICS bundles at the level of the platform after you have installed the platform, as you develop your applications and policies.

What to do next

When you have created the CICS Platform project, export it to zFS, set up CICS regions to match each CICS region definition that you created in a region type in your platform, and install the platform in the CICSplex to make it available. For instructions on how to do this, see Deploying a platform.

Deploying a platform

To deploy a platform, you export the CICS Platform project from CICS Explorer to the platform home directory in zFS, set up CICS regions to match each CICS region definition that you created in your platform, and create and install a platform definition (PLATDEF) in CICSPlex SM. The platform resource specifies the location of the platform in zFS and provides lifecycle management.

Before you begin

You must set up a platform home directory for the platform in zFS, as described in Creating a platform home directory in zFS, then create a CICS Platform project in the CICS Explorer, as described in Creating a platform.

About this task

The export process packages the CICS bundles that are referenced in the CICS Platform project and exports all the files for the platform bundle and the CICS bundles to the platform home directory in zFS. During the export, the wizard checks that the subdirectories of the platform home directory exist, and creates them if they do not exist. For more information about the platform directory structure, see Platform directory structure in z/OS UNIX.

For detailed help about the following steps to deploy a platform, see Working with platforms in the CICS Explorer product documentation.

Procedure

1. Using the CICS Cloud perspective in the CICS Explorer, export the CICS Platform project from CICS Explorer to the platform home directory in zFS.
2. Use the CICS Explorer to create a platform definition (PLATDEF), which points to the platform home directory in zFS, and identifies the target CICSplex for the platform. You can choose to create a platform definition immediately after exporting your platform project, by checking the box in the platform export wizard. To create your platform definition at another time, use the New Platform Definition wizard in the CICS Explorer. The platform definition is created in the data repository of the CMAS.
3. For each CICS region definition that you created in a region type in your platform project, set up an actual CICS region. As an alternative to setting up a new CICS region, you can use an existing CICS region that was not previously managed by CICSPlex SM.
   a. Create the CICS region with an APPLID, SYSID, and other attributes that match the system definition that you created. As a best practice, CICS regions in a region type should be clones of each other. For instructions to do this, see Setting up a CICS region.
   b. Perform the steps on the CICS region to make it known to CICSPlex SM as a managed application system (MAS). For instructions to do this, see Setting up a CICSPlex SM managed application system (MAS).

   Do not start the CICS regions yet.
4. Use the CICS Explorer to install the platform definition to create the platform in the target CICSplex.
5. For any CICS regions in created region types, start the regions now, using your normal method, then refresh the Cloud Explorer view in the CICS Explorer and confirm that the status of the platform is now ACTIVE.
6. Optional: If you have any CICS bundles deployed with the platform, right-click the platform and click **Enable**.

Results

When you install the platform definition, CICSPlex SM creates a PLATFORM resource to represent the platform in the CICSplex. CICSPlex SM also creates a record for the platform in the data repository, which is used in recovery processing for the bundles for platforms.

You can view the state of the platform, the region types, and the management parts in the Cloud Explorer view in the CICS Cloud perspective in the CICS Explorer. The platform is initially installed in a disabled state. When you enable the platform, any CICS bundles that are deployed with the platform, such as policies, are installed across the CICS regions in the appropriate region type.

The relationship between the platform and each installed CICS bundle is saved in a management part. The management part is a MGMTPART record that is created automatically for each CICS bundle during the platform installation process. The management part records the CICS regions where the bundle is installed, and tracks the status of the bundle in the CICS regions.

What to do next

If you experienced any problems during the installation process, see Diagnosing platform deployment errors.

When your platform is installed and enabled, you can deploy applications on the platform, and add policies to the platform to manage the environment and the applications.

Chapter 6. Setting up access for CICS Explorer

To configure and manage CICS regions from CICS Explorer, you must set up the CICS management client interface (CMCI). You can create a CMCI connection to a CICSPlex SM environment or to a stand-alone CICS region (SMSS).

Setting up CMCI with CICSPlex SM

To install the CICS management client interface (CMCI) in a CICSPlex SM environment, you must prepare a CICS region as a Web User Interface (WUI) server and set the **CMCIPORT** WUI system initialization parameter.

Before you begin

Before you use CMCI to manage CICSPlex SM managed resources, you must configure CICSPlex SM.

See Setting up CICSPlex SM for guidance about setting up a basic CICSPlex SM environment.

About this task

When you set up CMCI in a WUI server, you can use CMCI to manage all the CICS regions in all CICSplexes associated with the connected CMAS either directly or through other CMASs.

Procedure

To set up CMCI, you use essentially the same process as to set up any WUI server. The CICS system that you select must be a dedicated CICSPlex SM MAS connected to a CMAS. Follow the procedure that is described in Setting up a CICSPlex SM Web User Interface server to ensure that you set up your CMCI WUI server correctly.
As part of this process, take account of the following issues:

- When you specify the WUI server initialization parameters, you must specify a unique value for the **CMCIPORT** parameter. This parameter allocates a TCP/IP port number to CMCI. Setting a value for the **CMCIPORT** parameter ensures that CMCI is installed on the WUI server with the necessary URIMAP and TCPIPSERVICE definitions. CMCI must use a different TCPIPSERVICE definition to the WUI server.
- By default, URIMAP, and TCPIPSERVICE resource definitions are autoinstalled with security settings derived from the **SEC** CICS system initialization parameter and the **TCPIPSSL** WUI server initialization parameter. You can override the default CMCI TCPIPSERVICE settings by using the optional **CMCIAUTH** and **CMCISSL** parameters to enable greater security by using SSL certification. See CICSPlex SM Web User Interface security access overview for more information about setting up security for the WUI.
- Consider setting a nonzero value for the **DEFAULTWARNCNT** WUI server initialization parameter. Setting an appropriate value for this parameter prevents the retrieval of unacceptable amounts of data and can avoid long waits and

© Copyright IBM Corp. 1989, 2017

potential storage problems when making requests on CICS resources. See "Record count warnings in CMCI" on page 336 for guidance about the warning count mechanism in CMCI.

- To avoid possible storage problems, ensure that you set appropriate values for the following parameters:
  - The **MEMLIMIT** JCL parameter, to set storage for the above-the-bar storage for your WUI server.
  - The **MAXAUXCPSM** and **MAXAUXTOTL** CICSPlex SM system parameters, to set auxiliary storage for the CMAS.

  See "Estimating storage requirements for CMCI" on page 335 for guidance about setting these values.
- Ensure that resource definition group DFHWU is installed. Group DFHWU contains the resource definitions that are required for a CMCI environment.

Note: For CICS Explorer to be able inquire on CICS resources on the CSD, for example, programs, files, or transactions, it requires the system initialization parameter **CSDSTRNO** to be at least four (**CSDSTRNO=4**). If lower than four, the request might fail with: CNX0591E RESP=CSDERR RESP2=5 (insufficient VSAM strings).

Setting up CMCI in a stand-alone CICS region

To use the CICS management client interface (CMCI) in a stand-alone CICS region (SMSS) without using CICSPlex SM, you must add the SEYUAUTH and SEYULOAD libraries to the startup JCL of your CICS region and then install URIMAP and TCPIPSERVICE resource definitions.

About this task

When CMCI is set up in a CICS region, you can use it to manage CICS operational resources and CICS definitional resources. See CICS management client interface resource names in Reference for a complete list of the operational resources that the interface can manage.

Sample resource definitions are provided for setting up the CICS management client interface. DFH$WUUR is a sample URIMAP definition and DFH$WUTC is a sample TCPIPSERVICE definition. The samples are included in the CICS system definition file (CSD) in group DFH$WU. You can install these resources as they are, or more typically you can copy and modify them to tailor them for your environment. However, you must always specify the URIMAP path as `CICSSystemManagement/*`. This procedure does not activate security on the interface. For more information about security settings, see "Setting up CMCI in a CICS region with active security" on page 334.

If security is not active, messages that are produced by auditing system programming interface commands contain the default user ID of the region.

Use the following procedure to set up the CICS management client interface by using the samples provided.

Procedure

1. Change your CICS startup JCL:
   a. Add the *hlq*.CPSM.SEYUAUTH library to the STEPLIB concatenation, where *hlq* is your high-level qualifier; for example CICSTS54

b. Add the *hlq*.CPSM.SEYULOAD library to the DFHRPL concatenation, where *hlq* is your high-level qualifier; for example CICSTS54

These libraries must be at the same CICS TS level as those for CICS; that is, the same as the CICS *hlq*.CICS.SDFHAUTH and CICS *hlq*.CICS.SDFHLOAD libraries in the STEPLIB concatenation.

2. Ensure that the system initialization parameter CPSMCONN is set to NO for your CICS region. CPSMCONN must be set to NO for CICS Explorer to connect to a CMCI standalone region.
3. Ensure that the RDO group DFHFEPI is being installed in a list that is included in the group list (GRPLIST) at CICS startup. The DFHFEPI group is included in the default CICS startup list DFHLIST. The group contains resources that are required by CMCI.
4. Start your CICS region.
5. Install the sample URIMAP definition, DFH$WUUR. This sample URI map uses transaction CWWU and calls program DFHWBA to analyze the CICS web request. DFH$WUUR includes the following attribute values:

Attribute name	**Attribute value**
Description	Sample System Management Interface URI map
Group	DFH$WU
Host	*
Path	CICSSystemManagement/*
Port	No
Program	DFHWUIPG
Scheme	HTTP
Status	Enabled
TCP/IP service name	DFH$WUTC
Transaction	CWWU
URI map	DFH$WUUR
Usage	Server

Other attributes in the sample definition retain their default values.

Note: The TCP/IP service name must match your TCPIPSERVICE definition. If you use a TCPIPSERVICE definition with a name other than DFH$WUTC, ensure that you rename the TCP/IP service name in the URIMAP definition accordingly.

6. Copy and rename the sample TCPIPSERVICE definition, DFH$WUTC and change the port number to a unique value. DFH$WUTC includes the following attribute values:

Attribute name	**Attribute value**
Authentication level	No
CICS transaction ID	CWXN
Description	Sample System Management Interface TCPIP service
Group	DFH$WU
Host	Any

Attribute name	Attribute value
Port	1490
Protocol	HTTP
Queue backlog limit	0
SSL	No
Status	Open
TCP/IP service name	DFH$WUTC
Timeout for socket close	No
User-replaceable module name	DFHWBAAX

Other attributes in the sample definition can retain their default values.

7. Install the TCPIPSERVICE definition.

Tips:

- You must define the following transactions to RACF, or an equivalent external security manager, and ensure that CMCI users are authorized to access the transactions: CODB, COD0, COD1, COD2, COSH, COLU, CWWU, and CWXN.
- Ensure that the DFHCNV table used by CICS contains a DFHWBUD entry.
- If you use your own version of the DFHCNV source module, you must assemble and link-edit it using the new macros or ensure that you include the necessary code pages.

What to do next

Use CICS Explorer or a browser to check that your setup works correctly.

If necessary, you can use the following checks:

1. Check that the TCPIPSERVICE is OPEN and URIMAP is ENABLED on the stand-alone region. Confirm the port and that the URIMAP references the TCPIPSERVICE with the new name and not the old DFH$WUTC resource that you copied.
2. Try to access the URL by using the following request in your browser. If the browser returns a result, your setup is working as expected. The CMCI URIMAP responds only to requests for URLs that begin as follows:

   ```
   /CICSSystemManagement/: http://hostname:port/CICSSystemManagement/CICSTask/
   ```

3. Check for any SMSS related messages on the region job log, for example:

   ```
   EYUXL0022I xxxxxxxx SMSS Phase I initialization complete
   EYUXL0007I xxxxxPhase II initialization complete
   EYUNL0099I xxxxxxxx SMSS LRT initialization complete
   ```

4. Check for any installation errors in both the job log and the MSGUSR log.
5. Try to connect CICS Explorer to the CMCI port to check whether that connection works.

Setting up CMCI in a CICS region with active security

To set up CMCI with active security in a standalone CICS region (SMSS), you must change the settings in the sample definitions. Copy and rename the samples, then change the TCPIPSERVICE definition to enable security, and update the URIMAP definition to refer to the updated TCPIPSERVICE definition.

About this task

You can choose to use either HTTP basic authentication, or for a higher level of security, secure sockets layer (SSL) authentication. See Security for CICS Web support for information about the security measures you can use to protect access to the interface.

If security is active, messages produced by auditing system programming interface commands contain the user ID used to log on to CICS Explorer.

To set up the CICS management client interface with active security using the sample definitions, use the following procedure.

Procedure

1. Copy and rename the sample TCPIPSERVICE definition, DFH$WUTC, and the sample URIMAP definition, DFH$WUUR. These samples are included in the CICS system definition file (CSD) in group DFH$WU.
2. Change the TCPIPSERVICE definition to incorporate the security features that you want. See Creating TCPIPSERVICE resource definitions for CICS web support for guidance about creating TCPIPSERVICE definitions that include security for web clients.
3. Change the TCPIPSERVICE attribute in your URIMAP definition to refer to your renamed TCPIPSERVICE. You can also change the SCHEME attribute from HTTP to HTTPS, but this is not essential because this change is made automatically to an installed URIMAP if its associated TCPIPSERVICE has security enabled.
4. Install the definitions into your CICS region. See Ways of defining CICS resources for an explanation of the methods that you can use to install these resource definitions.

   Note: You must define the CWWU and CWXN transactions to RACF, or an equivalent external security manager, and ensure that CMCI users are authorized to access the transactions.

Estimating storage requirements for CMCI

To avoid possible storage problems when you use the CICS management client interface, you must specify appropriate levels of 64-bit (above-the-bar) storage in the CICS region, and auxiliary storage. Use the z/OS **MEMLIMIT** parameter to set the limit for 64-bit storage in your WUI server or CICS region, and the **MAXAUXCPSM** and **MAXAUXTOTL** CICSPlex SM system parameters to set auxiliary storage for the CMAS.

About this task

Running CMCI with large workloads can lead to short-on-storage situations on the WUI server and possible CMAS shutdowns caused by running out of auxiliary storage.

CMCI stores retained results sets for the WUI server in 64-bit storage in the CICS region, in subpool WU_64 in the GCDSA.

During a CMCI request, the CMAS collects and stores the requested resource records, which are then backed up in auxiliary storage. Running requests concurrently through the CMCI multiplies the number of records held by the CMAS with each request.

Procedure

1. Calculate the storage requirement for a typical request. Select the resource that is likely to generate the largest number of records in a request and multiply the number of records by the size of each record. See the appropriate table in CICSPlex SM resource tables in Reference to determine the record size.
2. Calculate your total requirements for 64-bit storage for retained result sets. Estimate your expected maximum number of retained result records for a single request and multiply this figure by your estimate of the storage required for each request. For example, if you can have 100,000 CICS terminals in a single request, multiply this figure by the size of the resource record as determined in step 1.
3. If you are setting up CMCI in a CICSPlex SM environment, estimate your auxiliary storage requirements for your CMAS. Estimate the maximum number of concurrent requests that you can expect and multiply this figure by your estimate of the storage required for each request as calculated in step 2. You can derive your estimate of concurrent requests from the total number of users that you expect to be using CMCI at any one time and how many simultaneous requests they are likely to make.
4. Consider adding more storage for metadata. CMCI stores all requested resource records for each request for a new retained result. For the initial request of a new resource type, for example, a first request for CICS programs, a small amount of attribute metadata is also stored. For large requests, the size of the attribute metadata and any other storage used while making the request is negligible compared to the storage required for records themselves. Consider adding an extra 2% to your final estimate to cover any extra metadata used internally by CMCI on the WUI server. This extra metadata is not necessary for the CMAS calculation.

What to do next

- If CMCI is installed in a CICSplex SM environment, use your estimate of auxiliary storage to set values for the **MAXAUXCPSM** and **MAXAUXTOTL** parameters on the CMAS associated with your WUI server.
- Use your estimate of the 64-bit storage required for retained result sets to help determine the z/OS **MEMLIMIT** value for your WUI server, or your CICS region if you are running the single server version of CMCI. You must allow for the other CICS facilities that use 64-bit storage.

  For information about the **MEMLIMIT** value for CICS, and instructions to check the value of **MEMLIMIT** that currently applies to the CICS region, see Estimating, checking, and setting MEMLIMIT in Improving performance. For further information about **MEMLIMIT** in z/OS, see Limiting the use of memory objects in the z/OS MVS Programming: Extended Addressability Guide.

Record count warnings in CMCI

Setting up record count warnings causes a CICS management client interface (CMCI) request to fail if a request is likely to result in the retrieval of an unacceptably large amount of data. You set up record count warnings by specifying a nonzero value for the **DEFAULTWARNCNT** initialization parameter during WUI server configuration when setting up CMCI.

DEFAULTWARNCNT can take an integer value in the range 0 - 99999999. The default value is 0, meaning that no warnings are issued.

If the warning mechanism determines that the number of records to be returned is greater than the value of the warning count, the request fails with an HTTP 403 response code. Users must issue a new request with different values for the SCOPE, CRITERIA and, in some cases, PARAMETER options to retrieve any results. If the returned value is less than or equal to the warning count value, the request is processed, in the usual way.

Note: In CMCI requests, the options **count**, **index** and **SUMMONLY** can be used to limit or prevent the display of records. However, these options do not influence the number of records requested and have no affect on the record count warning mechanism.

How to allow users to bypass record count warnings

If **DEFAULTWARNCNT** is set to a nonzero value, a warning count limit is in effect. Users can bypass the warning count limit by using the CMCI URI option **OVERRIDEWARNINGCOUNT** in their requests. For the CMCI URI option **OVERRIDEWARNINGCOUNT** to take effect, when setting up CMCI, you must set the **RESOURCELIMIT** initialization parameter to WARNING during WUI server configuration. If you want to prevent users from bypassing the warning count limit, set **RESOURCELIMIT** to FAIL.

When RESOURCELIMIT(WARNING) is in effect, if a request results in the retrieval of an amount of data larger than the warning count limit, a 403 HTTP response is returned and indicates that the errorInfo attribute **override_warning_count_allowed** is set to yes. The user can then bypass the warning count limit by using the CMCI URI option **OVERRIDEWARNINGCOUNT** in the request.

However, when RESOURCELIMIT(FAIL) is in effect, such requests are denied, and in the 403 HTTP response, the errorInfo attribute **override_warning_count_allowed** is set to no. Issuing a request with the **OVERRIDEWARNINGCOUNT** option will still result in the same 403 HTTP response.

Which CMCI resources are applicable

This feature does not apply to all resources. In the CMCI, record count warnings apply only to CMCI requests associated with the following resources:

CMCI Resources	
AIMODEL	
EPADAPT	
LOCFILE	
TASKESTG	
ATOMSERV	
EVCSPEC	
LOCTRAN	
TASKFILE	
BRFACIL	
EVNTBIND	
LSRPBUF	
TASKRMI	

CMCI Resources	
BUNDLE	
EVNTGBL	
LSRPOOL	
TASKTSQ	
BUNDPART	
EXCI	
MODENAME	
TCPIPGBL	
CFDTPOOL	
EXITGLUE	
MONITOR	
TCPIPS	
CICSDSA	
EXITTRUE	
MQCON	
TDQGBL	
CICSPAGP	
EXTRATDQ	
MQCONN	
TERMNL	
CICSRGN	
FEPICONN	
MQINI	
TRANCLAS	
CICSSTOR	
FEPINODE	
MVSESTG	
TRANDUMP	
CLCACHE	
FEPIPOOL	
MVSTCB	
TSKSPOLS	
CMDT	
FEPIPROP	
MVSTCBGL	
TSKSPOOL	
CONNECT	
FEPITRGT	

CMCI Resources	
MVSWLM	
TSMODEL	
DB2CONN	
HOST	
PARTNER	
TSPOOL	
DB2ENTRY	
HTASK	
PIPELINE	
TSQGBL	
DB2TRN	
INDTDQ	
PROCTYP	
TSQNAME	
DBCTLSS	
INTRATDQ	
PROFILE	
TSQSHR	
DOCTEMP	
IPCONN	
PROGRAM	
UOW	
DOMSPOOL	
IPFACIL	
RECOVERY	
UOWDSNF	
DSNAME	
JRNLMODL	
REMFILE	
UOWENQ	
DSPGBL	
JRNLNAME	
REMTDQ	
UOWLINK	
DSPMODE	
JVM	
REMTRAN	
URIMAP	

CMCI Resources	
DSPPOOL	
JVMPOOL	
REQID	
URIMPGBL	
EJCOBEAN	
JVMPROF	
RPLLIST	
WEBSERV	
EJCOSE	
JVMSERV	
RQMODEL	
WORKREQ	
EJDJAR	
LIBDSN	
STREAMNM	
XMLTRANS	
EJDJBEAN	
LIBRARY	
SYSDUMP	
ENQMODEL	
LOADACT	
TASK	
ENQUEUE	
LOADER	
TASKASSC	
EPADSET	
EPAINSET	
EVCSDATA	
EVCSINFO	
EVCSOPT	
MASHIST	
OSGIBUND	
OSGISERV	
RULE	
SYSPARM	
WEBSERV	

Chapter 7. Setting up event processing

Before you can use event processing, your CICS system must be correctly configured. You can stop and start event processing in CICS using a number of methods, including using IBM CICS Explorer.

About this task

You might want to stop event processing for an upgrade or system maintenance, and then start event processing again. This task explains the process usingCICS Explorer.

Use the LOCALCCSID system initialization parameter to specify the coded character set identifier for your local CICS region. This is the default code page for application programs. The default setting for LOCALCCSID is the EBCDIC code page 037. CICS translates the data content of event binding files into this code page before using it for event capture filtering operations, and assumes that any captured character data is in this code page.

Event processing is enabled by default when a START=INITIAL or START=COLD parameter is used during the startup of your CICS systems.

When a START=WARM or START=EMERGENCY parameter is used, the settings from the previous run of CICS are used.

Note: Do not change the status of event processing (that is, set to start, drain, or stop) while a unit of work that captures synchronous transactional events is in progress because you might cause the events to be backed out and the transaction to end abnormally.

Procedure

Use this task to stop or start event processing.

1. Using the CICSplex Explorer view in CICS Explorer, click the CICSplex or CICS region to select the CICSplex or CICS region on which you want to stop event processing.
2. Using the CICS Explorer toolbar, click **Operations** > **Event Processing**. The status of event processing is displayed.
3. Right-click to select the region on which you want to stop event processing and click **Stop**. Then click **OK**.

   Note: The status does not update until you click the **refresh** icon at which point the `STOPPED` status is displayed in the **Event Processing Status** column, indicating that event processing is stopped.
4. To restart event processing, right-click to select the region for which you want to start event processing and click **Start**. Then click **OK**.

   Note: The status does not update until you click the **refresh** icon at which point the `STARTED` status is displayed, indicating that event processing is started.

© Copyright IBM Corp. 1989, 2017

Results

When event processing is STOPPED, event capture is stopped immediately. All events on the dispatcher queue are deleted.

When event processing is STARTED, for in flight transactions, the capture of nontransactional events starts immediately and the capture of transactional events starts at the next sync point.

What to do next

You must enable an event binding and an EP adapter before you can emit any events for processing by an event consumer. For more information, see Enabling an event binding in the CICS Explorer product documentation and Enabling an event adapter in the CICS Explorer product documentation.

If you are using the WebSphere MQ EP adapter to format and process your events, you must set up the CICS-WebSphere MQ adapter to provide a connection between the CICS region and IBM MQ on z/OS. For instructions to do this, see Setting up the CICS-WebSphere MQ adapter.

Notices

This information was developed for products and services offered in the U.S.A. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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 rights 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 grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

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

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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 jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information 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.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

© Copyright IBM Corp. 1989, 2017

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

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.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming interface information

CICS supplies some documentation that can be considered to be Programming Interfaces, and some documentation that cannot be considered to be a Programming Interface.

Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 4 are included in the following sections of the online product documentation:

- Developing applications
- Developing system programs
- RACF security overview
- Developing for external interfaces
- Reference: application developmenth
- Reference: system programming
- Reference: connectivity

Information that is NOT intended to be used as a Programming Interface of CICS Transaction Server for z/OS, Version 5 Release 4 , but that might be misconstrued as Programming Interfaces, is included in the following sections of the online product documentation:

- Troubleshooting and support
- Reference: diagnostics

If you access the CICS documentation in manuals in PDF format, Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 4 are included in the following manuals:

- Application Programming Guide and Application Programming Reference
- Business Transaction Services
- Customization Guide
- C++ OO Class Libraries
- Debugging Tools Interfaces Reference
- Distributed Transaction Programming Guide
- External Interfaces Guide
- Front End Programming Interface Guide
- IMS Database Control Guide
- Installation Guide
- Security Guide
- Supplied Transactions
- CICSPlex SM Managing Workloads
- CICSPlex SM Managing Resource Usage
- CICSPlex SM Application Programming Guide and Application Programming Reference
- Java Applications in CICS

If you access the CICS documentation in manuals in PDF format, information that is NOT intended to be used as a Programming Interface of CICS Transaction Server for z/OS, Version 5 Release 4 , but that might be misconstrued as Programming Interfaces, is included in the following manuals:

- Data Areas
- Diagnosis Reference
- Problem Determination Guide
- CICSPlex SM Problem Determination Guide

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability
These terms and conditions are in addition to any terms of use for the IBM website.

Personal use
You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use
You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM online privacy statement

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below:

For the CICSPlex SM Web User Interface (main interface):
Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user’s user name and other personally identifiable information for purposes of session management, authentication, enhanced user usability, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface (data interface):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's user name and other personally identifiable information for purposes of session management, authentication, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface ("hello world" page):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect no personally identifiable information. These cookies cannot be disabled.

For CICS Explorer:
Depending upon the configurations deployed, this Software Offering may use session and persistent preferences that collect each user’s user name and password, for purposes of session management, authentication, and single sign-on configuration. These preferences cannot be disabled, although storing a user's password on disk in encrypted form can only be enabled by the user's explicit action to check a check box during sign-on.

If the configurations deployed for this Software Offering provide you, as customer, the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM Privacy Policy and IBM Online Privacy Statement, the section entitled “Cookies, Web Beacons and Other Technologies” and the IBM Software Products and Software-as-a-Service Privacy Statement.

Index

Special characters

.STEPLIB, CICS load library 134

Numerics

3270 terminals (non-SNA)
  eligible for autoinstall 247
3614 and 3624 devices
  ineligible for autoinstall 247

A

above the line, loading 266
ADYN, dynamic allocation transaction 76
agile service delivery
  creating 325
AMP parameter for specification of the GCD 136
AMT (autoinstall model table) 245
AOR 319
API improvements 122
application bundles 275
Application owning region (AOR) 319
applications 319
  bundles 277
ARMREGISTERED, named counter server 218
ASLTAB (z/OS Communications Server macro) 305
Atom feed 277
authorized libraries 134
AUTOCONNECT attribute
  TYPETERM 242, 250, 253
autoinstall
  and Communications server 244
  and security 257
  automatic TCTTE deletion 253
  control program 242
  effect on ATI 242
  effect on TCTUA 243
  effect on TLT 244
  eligible devices 247
  ineligible devices 247
  installing terminal resource definitions 304
  logon process 244
  models 241, 244
  QUERY function 244
  recovery and restart 250
  terminal definitions 304
  transaction routing restriction 244
autoinstall control program
  for connection autoinstall 259
  for program autoinstall 263
autoinstall model table (AMT) 245
autoinstall model terminal definitions
  installing 265
autoinstalling
  APPC connections 257
autoinstalling *(continued)*
  IPIC connections 260
  journals 265
  MVS consoles 253
  programs, map sets, and partition sets 261
  z/OS Communications Server terminals 242
automatic logoff 251
automatic sign-off 251
automatic start 115, 142
automatic TCTTE deletion 251
automatic teller machines (3614 and 3624)
  ineligible for autoinstall 247
automatic transaction initiation 242
  autoinstalled terminals 242
  lock on TCT entry preventing deletion 247
autostart override record 51
auxiliary temporary storage data set 31, 135
  control interval size 32
  job control statement for CICS execution 32
  job control statements to define 31
  number of control intervals 33
  space considerations 31
auxiliary trace data sets 60
  job control statements for CICS execution 60
  job control statements to allocate 62

B

BASESCOPE 299
BDAM data sets
  creating and loading 73
  opening and closing 76
BMS (basic mapping support) 32
  selecting versions of BMS 107
BMS ROUTE requests
  autoinstalled terminals 242
BSAM devices 307
  DD statements for 131, 307
BTS data set, DFHLRQ 138
bundle 295
  platform 325
bundle recovery 301
bundle resource interdependency 282
bundle security 301
bundled LIBRARY resources 282
bundled PROGRAM resources 282
bundled TRANSACTION resources 282
bundled zFS artifacts 289
bundles 282, 289
  defining 275
  in platforms 299
  scoping 299
  types of application 277
BWO (backup while open)
  data facility data set services (DFDSS) 30
  data facility hierarchical storage manager (DFHSM) 30
  disabling activity keypointing 29
  introduction 28
  restrictions 29
  storage management facilities 29
  storage management subsystem (SMS) 30

C

CADS 20
card reader 307
card reader and line printer 307
cataloged procedures
  starting CICS as a batch job 143
  starting CICS as a started task 139, 144
cataloging
  program autoinstall 262
CCSO transient data destination 136
CEDA transaction
  defining consoles devices 310
  installing SNA LU terminal definitions 304
  recovery and backup 15
  sharing a CSD between more than one CICS region 1
CEMT INQUIRE TERMINAL
  lock on TCT entry 244
CEMT master terminal transaction 65
CESF GOODNIGHT transaction 308
CESF LOGOFF transaction 308
CESN transaction 311
CESO transient data destination 136
channel-based service 277
characteristics of bundled resources 282
CI (control interval) 32
CICS global catalog
  autoinstall models 244
CICS journal utility program (DFHJUP) 50
CICS management client interface
  CICS region setup 332
  DFH$WUTC sample 331, 332
  DFH$WUUR sample URI map 331, 332
  security set up 335
  set up 331, 332, 335
  storage requirements 335
CICS server support for system-managed processes 230
CICS system definition file (CSD)
  offline utility program, DFHCSDUP 235
CICS-supplied TYPETERM definitions
  AUTOCONNECT attribute 242, 250
CICSplex 319

© Copyright IBM Corp. 1989, 2017

CICSPlex SM 122
CLSDST, issued by CICS at logon 244
CLT (command list table) 270
  CLT (command list table) 266
CMAC support, messages data set 81
CMCI
  DEFAULTWARNCNT 336
  set up 331
  setting record count warnings 336
  storage requirements 335
CMCIPORT parameter 331
cold start
  nonrecovery of autoinstalled TCT entries 250
  use of GRPLIST to re-create tables 314
  using to remove table entries 314
command list table (CLT) 270
  CLT (command list table) 266
command logs, RDO 19
commands
  CEDA command syncpoint criteria 18
  CEDA DEFINE 304
  CEDA INSTALL GROUP(groupname) 304
  DEFINE TERMINAL CONSNAME(name) 311
  DEFINE TERMINAL CONSOLE(number) 312
  DFHCSDUP INITIALIZE 312
  LIST ALL OBJECTS 5
  MODIFY command 309
  RDO CEDA INSTALL 50
  REPRO, to run IDCAMS 70
commit process 315
configuring
  agile service delivery 319
  platform 319
configuring CICS region 23
connecting CICS Explorer 331
connection autoinstall
  and recovery and restart 260
  autoinstall control program for 259
consoles
  autoinstalling 253
  console messages for CICS startup 145
  DEFINE TERMINAL CONSNAME(name) command 311
  DEFINE TERMINAL CONSOLE(number) command 312
  defining a TSO user 311
  defining to CICS 309
  devices 309
  entering system initialization parameters 113, 145
  ineligible for autoinstall 247
  TSO user, defining as a console device 309
  z/OS console as a CICS master terminal 309
consumer transaction facility (3614 and 3624)
  ineligible for autoinstall 247
control interval (CI) 32
control tables
  naming and suffixing 269
  preparing 266
copybooks
  DFH$TCTS 307
coupling facility
  managing storage 224
coupling facility data tables
  defining resources for 169
  starting a server 168
coupling facility list structures
  for coupling facility data tables 169
CSD
  sharing between releases 11
CSD (CICS system definition file)
  DD statements in CICS startup job 21
  defining 1
  definitions for the Japanese language feature 22
  dynamic allocation 21
  emergency restart and backout 17
  job control statements for CICS execution 21
  job to define and initialize 3
  making the CSD available to CICS 21
  offline utility program, DFHCSDUP 235
  recovery and backup 15
  space for data set 2
CSD (CICS system definition) file
  not used at restart 315
CSD file
  installing
    at CICS initialization 314
CSFU, CICS file utility transaction 77, 79
CSSL transient data destination 136
CSYSGRP 319
CXRF queue 37
CXRF transient data queue 37

D

data definitions
  installing a limited number of 316
data facility data set services 30
data facility hierarchical storage manager (DFHSM)
  backup while open 28
data sets
  auxiliary temporary storage 31
  auxiliary trace 60
  BDAM 73
  catalog data sets 50, 56
  CDBM SUPPORT data set 79
  created by DFHCMACI job 27
  created by DFHCOMDS job 26
  created by DFHDEFDS job 26
  CSD 1
  debugging profiles 84
    creating 85
    defining 85
  defining user files 74
  defining, transient data (extrapartition) 36
data sets *(continued)*
  defining, transient data (intrapartition) 34
  dump 62
  dynamic allocation using CEMT 76
  GTF data sets 28
  messages data set 81
  MVS system data sets used by CICS 27
  SDUMP data sets 27
  SMF data sets 27
  transient data (extrapartition) 34
  transient data (intrapartition) 34
  user data sets
    BDAM 73
    closing 77
    defining to CICS 74
    loading VSAM data sets 69
    opening 76
    VSAM 68
  VSAM bases and paths 69
data tables
  closing 79
  loading 79
  opening 79
  overview 78
  types of 78
DB2 load library
  requirement for DSNTIAR and DSNTIA1 135
DB2, RCT suffix option of CICS startup 127
debugging
  preparing for 140
debugging profiles data sets
  creating 85
  defining 84, 85
    as remote files 89
    as VSAM non-RLS files 88
    as VSAM RLS files 87
defining debugging profiles data sets
  as remote files 89
  as VSAM non-RLS files 88
  as VSAM RLS files 87
deleting
  existing entry at installation 316
  resource definitions from system tables
    at autoinstall log off 244
  TCT entry at log off (autoinstall) 244
deploying platforms 327
DEVTYPE macro (MVS) 64
DFH$TCTS, copybook 307
DFH$TDWT (transient data write-to-terminal sample program) 34
DFH$WUTC sample TCP/IP service 331, 332
DFH$WUUR sample URI map 331, 332
DFH99, sample DYNALLOC utility program 76
DFHAUXT auxiliary trace data set 60
DFHBUXT auxiliary trace data set 60
DFHCCUTL, local catalog initialization utility program 58
DFHCNV macro 266
DFHCOMP1, CSD resource definition group 12

DFHCOMP2, CSD resource definition group 12
DFHCSDUP
  definitions for the Japanese language feature 22
DFHCSDUP system definition utility program 235, 239
  command processing considerations 241
  invocation from a user program 239
  invoking as a batch program 237
  processing system definition file 235
  running under TSO 239
DFHCSVC, the CICS type 3 SVC 148
DFHCXRF data set, transient data extrapartition 37
  DD statements for 38
  in active CICS regions 37
DFHDBFK data set 81
  job control statements for CICS execution 81
  job control statements to define and load 80
DFHDCTG, group of transient data definitions 19, 20
DFHDPFMB
  debugging profiles data set
    creating 85
  defining
    as remote files 89
    as VSAM non-RLS files 88
    as VSAM RLS files 87
DFHDPFMP
  debugging profiles data set
    creating 85
  defining
    as remote files 89
    as VSAM non-RLS files 88
    as VSAM RLS files 87
DFHDPFMX
  debugging profiles data set
    creating 85
DFHGCD, global catalog data set 51
DFHISTAR job 26, 273, 275
DFHJUP, CICS journal utility program 50
DFHLCD, local catalog data set 57
DFHLRQ, BTS data set 138
DFHNCMN, named counter server region program 213
DFHNCO macro 208
DFHNCOPT, named counter options table 208
DFHPGADX—assembler program for program autoinstall exit 264
DFHPGAHX—C program for program autoinstall exit 264
DFHPGALX—PL/I program for 264
DFHPGALX—PL/I program for program autoinstall exit 264
DFHPGAOX—COBOL definition for 264
DFHPGAOX—COBOL definition for program autoinstall exit 264
DFHRPL, module load library 135
DFHRSTAT 196
DFHSIT keywords and operands 97
DFHSIT keywords and operands *(continued)*
  undefined keywords error message 107
DFHSIT macro
  GRPLIST operand 314
  START operand 314
DFHSM (data facility hierarchical storage manager)
  backup while open 28
DFHSTART, sample startup procedure 129
DFHTC2500, close terminal warning message 308
DFHTC2507, close terminal warning message 308
DFHTCT5$, sample TCT 307
DFHTCTDY, dummy TCT 108
DFHTEP, terminal error program 308
DFHXQMN system initialization parameter
  for pool list structure creation 156
DFHXQMN, TS server program 156
discovering 122
domains
  kernel 56
  parameters 56
DSA (dynamic storage areas)
  RENTPGM, storage for read-only DSAs 132
DSNTIA1 135
DSNTIAC 135
DSNTIAR 135
dump data sets 62
  dump table facility 63
  job control statements to allocate 65
  space calculations 63
dumps
  controlling with dump table 63
  effect of START= parameter 119
duplicate resource definition names 316
dynamic allocation
  ADYN, dynamic allocation transaction 76
  DFH99, sample DYNALLOC utility program 76
dynamic allocation of the CSD 21

E

emergency restart
  recreation of tables 315
  temporary recovery of autoinstalled entries 250
event processing 277
EXEC CICS START command
  autoinstalled terminals 242
exits
  XDUCLSE, dump global user exit 65, 67
  XDUOUT, dump global user exit 65, 67
  XDUREQ, dump global user exit 65, 67
  XDUREQC, dump global user exit 65, 67
extrapartition transient data 34
extrapartition transient data *(continued)*
  CSSL, and other destinations used by CICS 136
extrapartition transient data queues 136

F

facilities
  generalized trace facility (GTF) 27
  temporary storage 31
File owning region (FOR) 319
FILSTAT operand 79
FINAL, TYPE= macro 270
FOR 319
format of macros 268

G

GCD (global catalog data set)
  buffer space 55
  description 50
  job control statement for CICS execution 51
  job control statements to define and initialize 50
  space calculations 52
generalized trace facility (GTF) 27
global catalog
  AMP parameter for specification 136
  at installation 315
  for resource definitions 114
  use in restart 114
  with autoinstall 245
global catalog data set (GCD)
  buffer space 55
  description 50
  job control statement for CICS execution 51
  job control statements to define and initialize 50
  space calculations 52
groups of resource definitions
  installing
    while CICS is running 316
GRPLIST
  DFHSIT operand 314
  not used at restart 315
GTF (generalized trace facility) 27

I

IDCAMS, AMS utility program 70
IEV017 error message 107
initial start
  use of GRPLIST to re-create tables 314
  using to remove table entries 314
INITIAL, TYPE= macro 269
initializing CICS
  cold start 314
  emergency restart 315
  initial start 314
  warm start 315
INSTALL command
  to install a group of resource definitions 316

installation of resource definitions
and deletion of existing entry 316
cold start 314
emergency restart 315
initial start 314
quiescing resources first 316
warm start 315
intrapartition transient data 34, 136
intrapartition transient data queues
defining the intrapartition data set 34

J

Japanese language feature
installing definitions in the CSD 22
Java applications 277
JCL (job control language)
CICS startup 129
as a batch job 129
as a started task 139
job control language (JCL)
for CICS as a batch job 129, 143
for CICS as a started task 139, 144
job stream
CICS startup 129
jobs
DFHISTAR 26
journaling
BWO 28
JOURNALMODEL definitions 44
journals
autoinstalling 265

L

Language Environment runtime library, SCEERUN 130
Language Environment runtime library, SCEERUN2 130
LCD (local catalog data set)
description 56
job control statement for CICS execution 57
job control statements to define and initialize 57
use in restart 115
libraries
SCEERUN, Language Environment runtime library 130
SCEERUN2, Language Environment runtime library 130
line printer 307
list structure
storage considerations 212
list structure, defining 196, 211
loading, above the line 266
local catalog data set (LCD)
description 56
job control statement for CICS execution 57
job control statements to define and initialize 57
use in restart 115
locks, internal
put on by CEMT INQUIRE TERMINAL 244
log manager 38
log streams
mapping system log and journal names to 45
log-on process, with autoinstall 244
LOGA transient data destination 136
logging
defining CICS journals for general logs
forward recovery logs 43
user journals 43
defining CICS logs 38
defining CICS system logs 38
defining dummy logs 39
JOURNALMODEL definitions 44
log autoinstall 44
logoff (autoinstall), deletion of TCT entry 244
LU 304
LU, defining 304
LUTYPE 0 terminals
eligible for autoinstall 247
LUTYPE 1 terminals
eligible for autoinstall 247
LUTYPE 2 terminals
eligible for autoinstall 247
LUTYPE 3 terminals
eligible for autoinstall 247
LUTYPE 4 terminals
eligible for autoinstall 247
LUTYPE6.1
CICS-CICS ISC links and sessions
ineligible for autoinstall 247
CICS-IMS links and sessions
ineligible for autoinstall 247
MRO sessions
ineligible for autoinstall 247

M

macro definitions
format of 268
macros
ASLTAB (z/OS Communications Server macro) 305
DEVTYPE (MVS macro) 64
MDLTAB (z/OS Communications Server macro) 305
MGCR (to issue MVS commands) 312
MVS SDUMP 27
manifest
bundle 295
manifest definition 295
map sets
autoinstalling 261
MCT (monitoring control table) 267, 270
MDLTAB (z/OS Communications Server macro) 305
MEMLIMIT parameter 128
messages
console messages for CICS startup 145
messages data set
job control statements to define and load 82
messages data set for CMAC facility 81
MGCR macro, to issue MVS commands 312
MNPS 313
model terminal support (MTS) 257
MODIFY command 309
module load library concatenation, DFHRPL 135
monitoring 119
monitoring control table (MCT) 267, 270
MTS (model terminal support) 257
multinode persistent sessions 313
MVS consoles
autoinstalling 253
MVS SDUMP macro 63
MVS START command, to start CICS 139, 144

N

named counters
automatic restart manager parameters 215
controlling server regions 217
debug trace parameters 216
defining an options table 208
defining and starting server 213
deleting or emptying pools 220
dumping pool list structures 223
list structure parameters 216
list structure, defining 211
options table
defining 208
making available to CICS 211
parameters 209
parameters, server 214
server overview 206
server parameters 214
SET command 218
starting a server 206
unloading and reloading pools 221
warning parameters 217
XES 218
naming control tables 269
NETNAME attribute 250
no entry in TCT for 244
role in logging on with autoinstall 244
NEWSIT, system initialization parameter
effect on warm start 116
NOPS 313
NTO (network terminal option) 247

O

operator communication for initialization parameters 114, 146
OPNDST, issued by CICS at logon 244
OSGi bundles 277
OSGi recovery 301
overriding system initialization parameters
from the console 113, 145

overriding system initialization parameters *(continued)*
from the SYSIN data set 112
overview of coupling facility data table server 168
overview of temporary storage data sharing server 152

P

PARM startup parameter
system initialization parameters 132
PARMSRCE parameter 122
PARMTYPE parameter 122
partition sets
autoinstalling 261
persistent sessions 313
pipeline terminals
ineligible for autoinstall 247
PLATDEF 319, 327
platform 319
creating 325
deploying 327
PLATFORM 327
platform design 319
platform project 325, 327
platforms 319
PLT (program list table) 267
preparing control tables 266
preset security 257
printers
eligible for autoinstall 247
program autoinstall
and recovery and restart 264
cataloging 262
for autoinstall control program 263
model definitions 263
program autoinstall exit
DFHPGADX—assembler program for 264
DFHPGAHX—C program for 264
program autoinstall functions 264
program list table (PLT) 127, 267
programs
autoinstalling 261
PSDINT 313
PSTYPE 313

Q

QUERY function
with autoinstall 244

R

RBA (relative byte address) 36
RDO (resource definition online)
CICS system definition data set (CSD) 1
record count warnings 336
record-level sharing (RLS)
VSAM data sharing 70
recoverable service table (RST) 267
recovery and restart
and connection autoinstall 260
and program autoinstall 264
recovery and restart *(continued)*
autoinstall 250
recovery, OSGi bundles 301
RECOVNOTIFY 313
RECOVOPTION 313
referencing zFS artifacts 289
REGION parameter 128
region status
starting a server 195
region status server
list structure, defining 196
region type 319
relative byte address (RBA) 36
RELOAD, named counter pool 221
removing resource definitions from system tables
at a cold start 314
at an initial start 314
at autoinstall log off 244
RENTPGM, storage for read-only DSAs 132
resource definition online (RDO)
CICS system definition data set (CSD) 1
command logs 19
resource definitions
at system initialization
cold start 314
emergency restart 315
initial start 314
warm start 315
committing 315
installing 315
installing in CICS system
using the INSTALL command 316
removing from system tables
at a cold start 314
at an initial start 314
at autoinstall log off 244
restart
emergency
recreation of tables 315
temporary recovery of autoinstalled entries 250
warm
nonrecovery of autoinstalled entries 250
recreation of tables 315
result set warning counts 336
RLS, record-level sharing
VSAM data sharing 70
RST (recoverable service table) 267

S

sample job streams
CICS startup 129
samples
data for loading a BDAM data set 74
DFH$TDWT (transient data write-to-terminal program) 34
DFHSTART, sample startup procedure 129
DFHTCT5$, sample TCT 307
JCL to create and load a BDAM data set 73
samples *(continued)*
sample job to define auxiliary data sets on disk 62
scoping bundles 299
SCS printers
eligible for autoinstall 247
SDLC 3767 devices
eligible for autoinstall 247
SDUMP data sets 27
SDUMP macro 63
security
and autoinstall 257
bundle resources 301
for CICS management client interface 335
preset 257
sequence numbers
See named counters 206
sequential terminal devices 307
closing (quiescing) the devices 308
coding input for 308
DFHTC2500, close terminal warning message 308
DFHTC2507, close terminal warning message 308
end-of-file 308
logical close to quiesce 138
terminating input data 308
SET command, named counters 218
setting up a CICS region 23
SETXCF, delete named counter pool 220
sharing the CSD
freeing internal locks 12
protection by internal locks 8
sharing between CICS regions 9
single session terminals
APPC
eligible for autoinstall 247
single-node persistent sessions 313
SIT (system initialization table)
DFHSIT keywords and operands 97
DFHSIT TYPE=CSECT 97
DFHSIT TYPE=DSECT 97
supplying system initialization parameters to CICS 109
SIT parameters
retrieving information 122
SMS (storage management subsystem) 28, 30
SMSS
set up 332
set up with security 335
SNPS 313
space calculations
CSD 2
defining data sets 23
disk space 2
dump data sets 63
global catalog 52
START command, MVS 139, 144
START operand, DFHSIT 314, 315
start, cold
nonrecovery of autoinstalled TCT entries 250
start, emergency
temporary recovery of autoinstalled TCT entries 250

START, system initialization parameter
START=AUTO 115, 142
START=COLD 117, 143
START=INITIAL 116, 143
START=STANDBY 143
start, warm
nonrecovery of autoinstalled TCT entries 250
started task, CICS as a 139, 144
starting CICS regions 126, 141
as a batch job 143
as a started task 139, 144
MVS START command 139, 144
sample job stream 129
specifying the type of startup 114
START=AUTO 115, 142, 143
START=COLD 117
START=INITIAL 116, 143
START=STANDBY, for an XRF alternate CICS 143
startup job streams
terminals 304
startup procedure, DFHSTART 129
statistics 119
storage calculations 154
storage management subsystem (SMS) 28, 30
backup while open facility 28
introduction 30
release information 30
storage requirements
for the CICS management client interface 335
storage requirements for CICS regions 128
suffixes of CICS control tables 272
suffixing control tables 269
SVC
specifying DFHCSVC in the startup job 148
SYSIN data stream 132
SYSPARM resource table 122
system console 309
system data sets 23
system dumps 63
system initialization
for an alternate CICS (XRF=YES) START=STANDBY 143
how CICS determines the type of startup 114
START=AUTO 115, 142, 143
START=COLD 117
START=INITIAL 116, 143
system initialization parameters 122
entering at the console 113, 145
from operator's console 109, 114, 146
from the SYSIN data set 132
how to specify 89
in the PARM parameter 109, 111
in the SYSIN data set 109, 112
PGAICTLG 262
specifying on the PARM statement 132
system initialization table (SIT)
DFHSIT keywords and operands 97
supplying system initialization parameters to CICS 109
system management facilities
for CICS statistics 27
system startup 126, 141
startup job stream 129
system tables, CICS
deletion of existing entry (reinstalling resource definitions) 316
system task 139, 144
System–managed duplexing 232
System–managed rebuild 230

T

TCP/IP service
DFH$WUTC sample 331, 332
TCT (terminal control table) 268, 270
TCT entry, autoinstalled 243, 244, 245
TCT entry
automatic deletion, with autoinstall 253
automatic TCTTE deletion 253
deleted at logoff, with autoinstall 244
TCT user area (TCTUA) 243
effect of autoinstall 243
TCTUA (TCT user area) 243
effect of autoinstall 243
TDQ 20
TDQUEUE 20
temporary storage
starting up temporary storage servers 152
TS data sharing server 152
temporary storage data sharing
defining TS pools for TS data sharing 153
temporary storage server
sample startup job 156
terminal control table (TCT) 268, 270
dummy control table, DFHTCTDY 108
TCT entry, autoinstalled 243, 244, 245
terminal definition 304
TERMINAL definition
installation when out of service 316
TERMINAL 247
TERMINAL attribute 247
terminal error program, DFHTEP 308
terminal list table (TLT) 244, 268, 270
effect of autoinstall 244
Terminal owning region (TOR) 319
time sharing option (TSO)
the DFHCSDUP program 239
TLT (terminal list table) 244, 268, 270
effect of autoinstall 244
TLX terminals using NTO
eligible for autoinstall 247
TOR 319
trace
defining auxiliary trace data sets 60
DFHAUXT auxiliary trace data set 60
DFHBUXT auxiliary trace data set 60
job control statements to allocate auxiliary trace data sets 62
trace *(continued)*
sample job to define auxiliary data sets on disk 62
transaction list table (XLT) 268
transaction routing
autoinstall restriction 244
transactions
ADYN, dynamic allocation transaction 76
CESF GOODNIGHT 308
CESF LOGOFF 308
CESN 311
CSFU, CICS file utility transaction 77
transient data (extrapartition) data sets 34
defining 36
transient data (intrapartition) data set
defining 34
failure to open 35
multiple extents and volumes 36
other considerations 36
transient data queue 20
transient data queues 34
transient data write-to-terminal sample program (DFH$TDWT) 34
TSO (time sharing option)
the DFHCSDUP program 239
TSO users 311
TWX terminals using NTO
eligible for autoinstall 247
TYPE=CSECT, DFHSIT 97
TYPE=DSECT, DFHSIT 97
TYPE=FINAL macro 270
TYPE=INITIAL macro 269
types of data tables 78
TYPETERM definition 245

U

UNLOAD, named counter pool 221
URI map
DFH$WUUR sample 331, 332
user file definitions 67
user files
coupling facility data table server 168
utility programs
DFHCCUTL, local catalog initialization utility program 58
DFHJUP, CICS journal utility program 50
IDCAMS, AMS utility program 70
utility programs, offline
initializing CICS system definition file, DFHCSDUP 235
running under TSO 239

V

variables 303
VERBEXIT, IPCS parameter 63
VSAM data sets 68
bases and paths 69
loading empty VSAM data sets 69
opening and closing 76
VSAM intrapartition data set 34

W

warm restart
  nonrecovery of autoinstalled
    entries 250
  recreation of tables 315

X

XDUCLSE, dump global user exit 65, 67
XDUOUT, dump global user exit 65, 67
XDUREQ, dump global user exit 65, 67
XDUREQC, dump global user exit 65, 67
XES, named counter server response to 218
XLT (transaction list table) 268
XML-based service 277
XRES system initialization parameter 301

Z

z/OS Communications Server
  ACB at CICS startup 120
  deletion of TCT entry by autoinstall at end of session 244
  logging on to CICS through 244
  NETNAME 244, 250
  persistent sessions support 313
  terminals, statements for 305
z/OS Communications Server terminals
  autoinstalling 242
z/OS console, defining to CICS 309
z/OZ Communications ServerSNA
  and autoinstall 244

