CICS Batch Application Control for z/OS

User's Guide

Version 1 Release 1 Modification 1

SC34-6321-06

CICS Batch Application Control for z/OS

User's Guide

Version 1 Release 1 Modification 1

SC34-6321-06

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 195.

Seventh edition (June 2009)

This edition applies to Version 1 Release 1 Modification 1 of CICS Batch Application Control, program number 5697-I94, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

At the back of this publication is a page entitled “Sending your comments to IBM”. If you wish to send comments by mail, please address them to:

User Technologies Department
Mail Point 095
IBM United Kingdom Laboratories
Hursley Park
WINCHESTER
Hampshire
SO21 2JN
United Kingdom

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

Contents

Figures

Tables

Preface

This book describes CICS® Batch Application Control, generally referred to in this book as CICS BAC for short. The book covers:

- An overview of the product and its main components
- The system requirements for CICS BAC
- The installation and operation of CICS BAC
- The commands that you can issue from the batch request utility to control CICS resources
- A brief overview of the workstation client
- The messages and abend codes issued by CICS BAC
- The sample migration exit program.

Who this book is for

This book is intended for anyone who needs information or guidance on using CICS Batch Application Control for z/OS®. This includes system designers, programmers, and those responsible for setting up, administering, and maintaining CICS BAC.

What you need to know

This book assumes that you are familiar with CICS Transaction Server for z/OS and MVS™ batch processing.

Terminology

For convenience, the term CICS BAC is used in this book as an abbreviation for CICS Batch Application Control.

Summary of changes

This is the first edition of this book and therefore there are no changes to describe. Any changes to future editions will be listed in this section.

Chapter 1. Concepts and facilities

This section describes the concepts of CICS Batch Application Control (CICS BAC), and the facilities it provides, under the following topics:

- Overview
- Architecture

Overview

CICS BAC enables you to manage easily batch processes that have to coexist and share resources with one or more CICS online transaction systems.

Much of the most valuable commercial data in the world is processed primarily through CICS transactions. The most critical data is also usually the most-used, and this means there is potential for contention over data access. From time to time, a CICS administrator needs to take resources offline from CICS to perform batch processes. CICS BAC is a product that meets the most commonly found requirements to help you manage this process.

CICS gives you dynamic control of CICS file control, transient data, transaction, and program resources. With CICS BAC, you can create batch jobs that use control requests to release, and later reallocate, resources associated with CICS applications. These batch jobs can:

- Issue control requests that process either individual resources or groups of resources
- Invoke processes that use the resources while they are released
- Invoke CICS programs and transactions, including CEMT.

The core of CICS BAC is the batch request utility, which runs as a standard batch job step and enables you to control the states of CICS resources within multiple CICS regions. By invoking CICS BAC in successive batch jobs, and by scheduling multiple batch jobs throughout the working day, you can optimize the way resources are managed in your particular environment.

Architecture

CICS BAC consists of a number of components that work together to provide you with an effective control mechanism for sharing CICS resources between CICS systems and your batch applications. These components are:

- "The CICS BAC communication server" on page 3
- "The CICS BAC workstation administration client" on page 4
- "The CICS BAC control file" on page 5
- "The CICS BAC request server" on page 5
- "The CICS BAC startup processor" on page 6
- "The CICS BAC batch request utility" on page 6

An outline diagram of all these components is shown in Figure 1 on page 2.

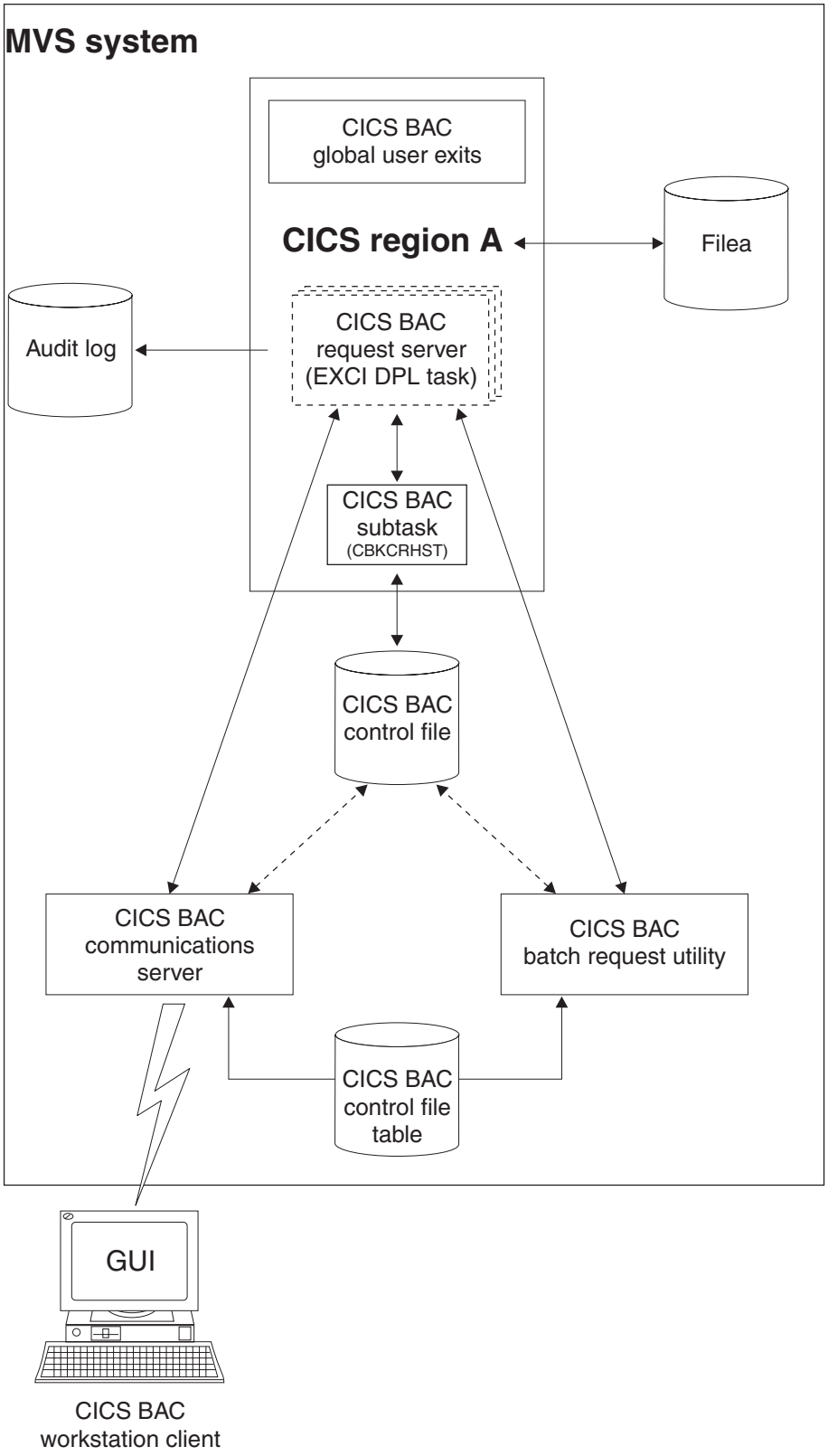


Figure 1. Diagrammatic view of CICS BAC operating during normal CICS running. The replication of the broken lines round the CICS BAC request servers indicates that there can be more than one. For example, there could be one supporting an EXCI DPL request from a communication server and one or more EXCI DPL requests from batch request utilities. Note that this diagram does not show the file maintenance utility or the ISPF administration interface.

The CICS BAC communication server

You can choose to define resource definitions for CICS BAC using the Windows workstation client (see "The CICS BAC workstation administration client" on page 4), in which case you need to create one or more communication servers. A CICS BAC communication server, which runs in its own MVS address space, provides the functions you need on the mainframe to send and receive requests from the client over a standard TCP/IP connection. The communication server channels requests from the workstation to the appropriate CICS region to update its CICS BAC control file. If the CICS region is not available, the communication server itself attempts to allocate and open the control file, and process the request. For more information about how the communication server handles requests to update a CICS BAC control file, see Chapter 5, "CICS BAC workstation administration client," on page 63.

Note that you need a communication server only if you plan to use the CICS BAC workstation administration client. If you perform all administration tasks by using either the ISPF administration interface or the file maintenance utility, you do not need a CICS BAC communication server.

You can start the communication server as either a started task or batch job, and it runs entirely in problem-program state. It does not need to run as APF-authorized, unless you want it to perform client request-security checking. You can run multiple communication servers on a single MVS image, but each server requires a unique server identifier and TCP/IP port number. The port number (in conjunction with the communication server's TCP/IP address) is used by the workstation client to direct its requests to the appropriate server. There is no tie between a particular communication server and a specific CICS region. Different servers can communicate with the same CICS region and a communication server can communicate with different CICS regions.

You notify the communication server about the CICS regions that support CICS BAC by means of a control file table that you define (see "The control file table" on page 23). The communication server loads the control file table at startup. When it receives a workstation request for a CICS region, it determines whether or not CICS BAC is running in the CICS region by attempting to acquire a systems-level enqueue that is normally held by the CICS region when CICS BAC is running in the region. If the server cannot get the systems-level enqueue (implying that CICS BAC is active in the destination region), it sends the request to CICS for servicing. The server uses the CICS external CICS interface (EXCI) facility to send its requests to the CICS region by means of a distributed program link (DPL) command. This allows the server to communicate with any CICS region in the same sysplex that is running EXCI support

The communication server is not constrained by the 32 KB commarea size limit. If the amount of data on a request to CICS is larger than 32 KB, CICS BAC breaks the single logical request into multiple DPL requests that are processed by the CICS BAC request server when all request data has been sent to the region. A similar technique is used when the length of data being returned to the server as a result of a DPL request is greater than 32 KB.

If CICS BAC is not active in the destination CICS region, the communication server services the request itself, provided the request does not require direct access to CICS resources or information. When servicing the request itself, the server

determines the data set name of the CICS BAC control file, for the destination CICS region, by searching its control file table. If found, it dynamically allocates and opens the file, then services the request. If the request requires direct access to CICS resources (for example, a request to return a list of files defined to CICS), and CICS BAC is not active in the CICS region, the server fails the request with the appropriate return code.

For information about setting up and starting a communication server, see "Setting up and starting the communication server" on page 25.

The CICS BAC administration tools

CICS BAC requires a small amount of initial and some ongoing administration. To make the administration tasks as flexible as possible and offer you a choice of method of working, CICS BAC provides three administration tools. Because each tool provides virtually the same functionality, you have the option of choosing which tool or combination of tools fits best in your environment. The three methods, which are introduced briefly in the following topics, are:

- "The CICS BAC workstation administration client"
- "The CICS BAC ISPF administration interface"
- "File maintenance utility" on page 5

You can choose to use one of the tools exclusively, or choose to use any combination of the three tools, even in the same MVS image. However, you need to be aware of data integrity considerations if you are using the ISPF administrative interface with, either of the other two (see "Data integrity considerations" on page 66).

The CICS BAC workstation administration client

CICS BAC provides a Windows-based workstation administration client that enables you to perform administration tasks from a Windows-based PC. The client communicates directly with a CICS BAC communication server through a TCP/IP connection. The CICS BAC communication server is responsible for passing on the requests to the appropriate CICS region to be processed or, if CICS BAC is not active in the target region, processing the changes itself by directly accessing the appropriate CICS BAC control file.

For more general information about the workstation administration client, see Chapter 5, "CICS BAC workstation administration client," on page 63. For detailed information about how to install and use the workstation administration client, see the *CICS BAC Workstation User's Guide*.

The CICS BAC ISPF administration interface

CICS BAC also provides an ISPF-based administration user interface that enables you to carry out CICS BAC administration tasks from a TSO/E ISPF session. Unlike the workstation administration client, the ISPF administration interface does not use the services of a CICS BAC communication server. Instead, it issues CICS EXCI requests to the CICS BAC request server if CICS BAC is active in the target CICS region. If CICS BAC is not active in the target CICS region, the ISPF administration interface processes requests directly by dynamically allocating and opening the target CICS region CICS BAC control file.

For an overview of the ISPF administration interface, see Chapter 6, "CICS BAC ISPF administration interface," on page 65. The ISPF administration interface online help facility provides the information you need on how to complete the

various CICS BAC ISPF panels. For information on how to get customize and started with the ISPF interface, see “Setting up and starting the ISPF administration interface” on page 35

File maintenance utility

CICS BAC provides a file maintenance utility that enables you to maintain a CICS region control file through an MVS batch job. You run the file maintenance utility as a batch job, which passes file maintenance commands to the CICS BAC request server in the target CICS region. If the target CICS region is down, or CICS BAC is not active, the file maintenance utility processes commands directly in the target CICS control file.

The file maintenance utility enables you to create and modify information in a CICS BAC VSAM KSDS control file. You can also list information held in a CICS region control file. For information about this batch utility, see Chapter 7, “CICS BAC file maintenance utility,” on page 67.

The CICS BAC control file

Each CICS region has its own control file that CICS BAC uses to maintain all control and state information for the CICS region. When a CICS BAC batch request utility job step has a request for a CICS region that is not active, the batch job step itself allocates and opens the CICS region control file and updates the state information for the specified resources. However, the batch job step can do this only if the control file is not open to its owning CICS region, a communication server, or another batch request utility job step. Access to a specific control file might be required from different MVS images within a sysplex, therefore access to a control file is serialized by means of a systems-level enqueue obtained by the CICS BAC startup processor (see “The CICS BAC startup processor” on page 6), which it holds as long as CICS BAC is active in the CICS region. This prevents any CICS BAC batch request utility job step or communication server from attempting to open a control file while it is open to its CICS region. All control file information is updated at the time a request is made by either the CICS region, the communication server, or the batch request utility job step initiating the request.

CICS BAC also updates the CICS region control file when the region changes the state of a resource being tracked by CICS BAC; for example, as a result of CEMT or EXEC CICS SET FILE commands. In this case, the CICS BAC global user exit is invoked as a result of the file request, and it calls the CICS BAC subtask to update the control file record with the changed file status. In this case, the request server subtask performs the work independently of the transient CICS request server task.

CICS BAC also allows you to control resources that are not explicitly defined in the CICS region control file. For example, you can close and disable a file using the CICS BAC batch request utility program, even if there is not an entry for the file in the CICS region control file.

For information about creating a control file for each CICS region, see “Defining and initializing a CICS BAC control file” on page 11.

The CICS BAC request server

All requests sent to a CICS region are handled by the CICS BAC request server, which runs as a standard CICS application program initiated by a CICS distributed program link (DPL) request using the external CICS interface (EXCI). Requests can originate in a CICS BAC communication server or a batch job step running the CICS BAC batch request utility. The updates to the CICS region control file are

performed by the CICS BAC request server subtask, which is started by the CICS BAC startup procedure and then runs in the CICS address space as long as CICS BAC is active in the CICS region.

The CICS BAC startup processor

Each CICS region operating with CICS BAC support performs a CICS BAC startup procedure. This is initiated either by an entry for the startup processor in the CICS region PLTPI, or by a supplied transaction. The primary function of the CICS startup processor is to ensure that all resources reflect their states as known to CICS BAC by means of the control file. For example, if a CICS region named CICSHUR1 is not running, and a CICS BAC batch job issues a request to close FILEA on CICSHUR1, the batch request utility itself updates the CICS region control file with the new file state (CLOSED), then continues processing. If CICSHUR1 is then started before the batch job issues a CICS BAC request to re-open the file, the startup processor running during PLTPI processing recognizes that FILEA should remain closed, and issues the appropriate CICS commands to ensure FILEA stays closed.

The startup processor also starts the CICS BAC request server subtask. This ensures that the control file is allocated uniquely to the CICS region and prevents either the communication server or a batch request utility allocating and accessing the file directly.

When the startup processor has completed its work, it terminates, leaving the CICS BAC request server subtask running to handle batch request utility or communication server requests that are sent to the CICS region, and which require control file access. This is illustrated in Figure 1 on page 2.

For information about initiating the CICS BAC startup processor, see "Setting up and starting the CICS component" on page 13

The CICS BAC batch request utility

The CICS BAC batch request utility runs as a job step in a batch job stream. You use the utility to issue commands to one or more CICS regions to ensure that the state of a given resource is as requested (see Chapter 4, "CICS BAC batch request utility commands," on page 43 for details of the utility commands). The batch request utility communicates with the CICS request server using DPL requests over EXCI links. If CICS BAC is not active in the target CICS region, the batch request utility updates the appropriate records in the CICS region control file to reflect the required resource state. The batch request utility must know the data set name of the control files for all CICS regions with which it attempts to communicate, in case a CICS region is not available. It determines the data set name by searching the CICS BAC control file table member, CBKCFTBL. CICS BAC uses one of the following methods, in this order of precedence, for locating the CBKCFTBL member.

- Searches the CBKPARMS data set, defined in the batch job step JCL on the CBKPARMS DD statement, for the CBKCFTBL member that defines the CICS applids and their respective CICS BAC control file data set names
- Searches the MVS logical parmlib concatenation for the CBKCFTBL member that defines the CICS applids and their respective CICS BAC control file data set names.

The batch request utility uses the first member that it finds.

For more information about setting up and running a CICS BAC batch request utility, see “Setting up and running a CICS BAC batch request utility” on page 31.

Chapter 2. CICS BAC system requirements

This chapter describes the minimum hardware and software requirements for installing and using CICS Batch Application Control. Before beginning the installation process, you should ensure that your environment contains the minimum requirements as listed below.

Software requirements

The following are the minimum software requirements for CICS BAC:

- OS/390® Version 2 Release 10 or above, or z/OS Version 1 Release 1 or above
- CICS Transaction Server for OS/390 Version 1.3 (5655-147); or CICS Transaction Server for z/OS, Version 2 (5697-E93); or CICS Transaction Server for z/OS, Version 3 (5655-M15); CICS Transaction Server for z/OS, Version 4 (5655-S97); or, CICS Transaction Server for z/OS, Version 5 (5655-Y04).

Hardware requirements

About this task

CICS BAC runs on any S/390® or System z® machine on which the applicable CICS Transaction Server systems are running.

You must also have TCP/IP connectivity between the host mainframe and any workstation running the CICS BAC workstation administration client.

Workstation requirements

About this task

To run the CICS BAC workstation administration client, you need the following:

- An Intel-based PC (or compatible) with at least a Pentium III processor and 256 MB memory
- 20 MB of available disc storage
- Windows 2000 Professional or Windows XP Professional operating system
- TCP/IP connectivity between the CICS BAC workstation administration client and the CICS BAC communication server, which runs in its own MVS address space. The communication server ensures that requests from the client reach the appropriate CICS region or services the requests itself.

Chapter 3. Implementation and customization

About this task

This chapter describes the steps necessary to implement and customize CICS BAC. Before starting any of these tasks, complete all the installation procedures described in the *Program Directory*, GI10-2574, then complete the tasks in this chapter in the order they are presented.

The main topics covered are:

- "Defining and initializing a CICS BAC control file"
- "Setting up and starting the CICS component" on page 13
- "Defining CICS BAC data sets" on page 23
- "Setting up and starting the communication server" on page 25
- "Setting up and running a CICS BAC batch request utility" on page 31
- "Setting up and starting the ISPF administration interface" on page 35.

Defining and initializing a CICS BAC control file

About this task

Each CICS region that uses CICS BAC support requires its own unique CICS BAC control file. You define this as a VSAM key-sequenced data set (KSDS) and initialize it using the sample JCL shown in Figure 2 on page 12. This sample job is also supplied as member CBKDIFIL in the SCBKSAMP data set.

```
//CBKDIFIL JOB (accounting informaton)
//* Define a control file as a VSAM KSDS
//*
//DEFINE    EXEC PGM=IDCAMS,REGION=1M
/*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DEFINE CLUSTER (NAME(hlq.control.file) -
         UNIQUE                -
         INDEXED               –
         CYL(3 1)              -
         SHR(1)                -
         RECORDSIZE(2048 32760) -
         KEYS(96 0)            –
         FREESPACE(20 20)      -
         VOLUMES(volume) )     -
        DATA                   -
         (NAME(hlq.control.file.data)  –
         CISZ(32768) )         –
        INDEX                  -
         (NAME(hlq.control.file.index) -
          NOIMBED              -
          NOREPLICATE)
//*
//*  Initialize the newly created data set
//*
//INITFILE EXEC PGM=CBKIFILE,PARM='applid'
//STEPLIB  DD  DISP=SHR,DSN=hlq.SCBKLOAD
//CBKCNTL  DD  DISP=SHR,DSN=hlq.control.file
//SYSPRINT DD  SYSOUT=*
//SYSABEND DD  SYSOUT=*
```

Figure 2. Sample JCL to define and initialize a CICS BAC control file

Note: Edit the sample CBKDIFIL job shown in Figure 2 by substituting your own values for the names shown in bold italic characters. Specify your own names for:

- The control file KSDS
- The volume ID, unless you are using SMS-managed volumes, in which case you can omit the volume parameter
- The applid of the CICS region for which you are defining the data set.
- The high-level qualifier of the CICS BAC load library.

When a CICS region KSDS has been defined by IDCAMS in the first job step shown in Figure 2, the CICS BAC control file initialization program, CBKIFILE, then initializes the data set in job step 2. This program creates one default CICS region properties object record and one default object record for each of the resource types:

- File
- Program
- TD queue
- Transaction ID

All these default records have the same identifier, namely $DEFAULT, and they provide default values when object records are being created in the following circumstances:

- When you are running the file maintenance utility to add records to the CICS region control file and you omit some parameters from an ADD *resource_type* command. The missing parameter values are taken from the appropriate $DEFAULT record.

- When the CICS BAC request server is creating a control file record as a result of processing a command from the batch request utility, or the callable API, for an undefined object.
- When the CICS BAC CICS state monitor detects an undefined object and the CICS region properties record specifies that a control file record is to be created in this situation.

For information about undefined object processing, see the description of the CREATERECORD parameter on the UPDATE REGION command on page 105.

You can update these default object records to set your own default values, using either the file maintenance utility UPDATE commands, or the workstation administration client. To see what parameter values are set in these $DEFAULT records, you can use the file maintenance utility "LIST" on page 91 command to get a listing of each record or view the records using the workstation administration client. For information about using the file maintenance utility to maintain a CICS region control file after you have defined and initialized it, see Chapter 7, "CICS BAC file maintenance utility," on page 67.

Setting up and starting the CICS component

About this task

This section describes the steps needed to implement and customize CICS BAC in a CICS region. Complete these steps, in the order presented, for each region in which you plan to use CICS BAC support. Before starting any of these tasks, complete all the installation procedures detailed in the CICS Batch Application Control for z/OS *Program Directory*, GI10-2574.

Preparing CICS regions for CICS BAC

About this task

The CICS BAC components that you need in your CICS regions consist of:

- CICS BAC load modules
- The CICS BAC command processor transaction
- Several control programs and a mapset
- A control file
- CICS global user exit programs
- A program, CBKCHRST, that runs as an MVS subtask within the CICS region's address space

You might also want to take advantage of CICS TS features that allow you to activate automatically CICS BAC support during CICS initialization and terminate it automatically during CICS shutdown processing.

Checklist

About this task

The following checklist summarizes the main steps described in the following topics:

- Add the required CICS BAC load modules to your CICS region (see "Adding CICS BAC load modules to CICS" on page 14).

- Define the required CICS BAC transaction, program, mapset, and control file resource definitions to your CICS region (see “Defining the required CICS BAC resource definitions”).
- Install the required CICS BAC resource definitions (see “Installing the CICS BAC resource definitions” on page 17).
- Check that the required external CICS interface (EXCI) definitions are installed (see “Other CICS services used” on page 17).
- Decide how you are going to start CICS BAC, and if you decide to use the PLTPI program, generate your PLT with the CICS BAC program entry (see “Using the CICS BAC PLTPI program” on page 18).
- Plan your shutdown method (see “Shutting down CICS BAC during CICS shutdown” on page 19).

Adding CICS BAC load modules to CICS
About this task

Ensure the CICS BAC load modules are available to your CICS region before attempting to activate CICS BAC. The load modules are installed in the SCBKLOAD load library when you installed CICS BAC. The CICS BAC *Program Directory* provides information about the location and name of the load library. To make the load modules available to CICS, you can:

- Add SCBKLOAD to the DFHRPL concatenation in the CICS region JCL, or
- Copy the members of SCBKLOAD to another data set that is already in the DFHRPL concatenation.

Although the first method is recommended, it requires you to restart the CICS region after you've modified the JCL. The second method allows you make the CICS BAC modules available without restarting CICS. However, keeping product load modules in their respective data sets is normally recommended

Defining the required CICS BAC resource definitions
About this task

CICS BAC requires the following CICS resource definitions to enable CICS BAC support in a CICS region:

- The CICS BAC transaction, default name KBKM
- CICS BAC program and mapset definitions
- The CICS BAC control file definition

You can use the CICS CEDA transaction to define the above, as shown in the examples later in this section. Alternatively, the SCBKSAMP data set contains JCL and input statements that you can customize and use to define the resources using the CICS DFHCSDUP batch utility program. The SCBKSAMP members for the transaction, program, and file definitions are CBKTRANS, CBKPROGS, and CBKFILES, respectively. For more information about defining CICS resources using the DFHCSDUP utility, see the *CICS Resource Definition Guide.* for your release of CICS TS.

Regardless of which of the two methods you choose to define the CICS BAC resources, you should allow any parameters not explicitly specified in the commands described in this section to take their default values. Also, for ease of use, use the same group name for all definitions.

CICS BAC transaction definition
About this task

Define the CICS BAC transaction so that you can activate and control CICS BAC in the CICS region. You should define the CICS BAC transaction, even though you can start CICS BAC by including an entry in the CICS region PLTPI table.

Except for the last character, the CICS BAC transaction ID is not fixed, but KBKM is the preferred ID. If the KBK transaction prefix conflicts with existing definitions, or does not meet your installation standards, specify a different prefix, ensuring the first three characters are in the character set allowed by CICS. The fourth character of the transaction identifier must be the letter M. To avoid confusion, we recommend that you use the KBK prefix if possible. Note that you cannot change the prefix of CICS BAC program names.

You can use the CEDA command shown in Figure 3 to define the CICS BAC transaction resource definition in an active CICS region.

```
CEDA DEFINE GROUP(cbkgroup) TRANSACTION(kbkM) PROGRAM(CBKCMNDS)
TASKDATALOC(ANY) TASKDATAKEY(CICS)
```

Figure 3. CEDA command to define the KBKM transaction in your CSD

Note:

1. In the example shown in Figure 3, you can choose your own values for the strings shown in italics, but you are recommended to use KBKM as the transaction identifier. The transaction ID must end with the letter M.
2. As an alternative to the CEDA commands shown in Figure 3, you can use the CICS BAC sample JCL in the SCBKSAMP data set member, CBKTRANS, to define the transaction using the CICS DFHCSDUP utility.

CICS BAC program and mapset definitions
About this task

You can use the sample commands in Figure 4 on page 16 to define the programs and mapsets that CICS BAC requires.

```
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKCBIRS) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKCENQU) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKCMNDS) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKCIOBJ) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKCMODA) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKCMODB) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKCMODC) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKCMODD) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKCMODE) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKCRHST) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKCSETF) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKCSETP) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKCSETQ) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKCSETT) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKCSHUT) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKCSRSC) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKCSRVR) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKCSTAT) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKFSXnn) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKMTENU) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKMTxxx) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) PROGRAM(CBKSPXnn) DATALOCATION(ANY) LANGUAGE(ASSEMBLER) EXECKEY(CICS)
CEDA DEFINE GROUP(cbkgroup) MAPSET(CBKMAP1)
```

Figure 4. CEDA commands to define the CBK programs and mapset in your CSD

Note:

1. In the CEDA commands shown in Figure 4, replace *cbkgroup* with any character string allowed by the CEDA transaction that meets your installation standards. Replace the *nn* in the program names CBKFSX*nn* and CBKSPX*nn* with 53 if you are installing CICS BAC in a CICS TS V1.3 region; 62 if you are installing CICS BAC in a CICS TS V2.2 region; 63 if you are installing CICS BAC in a CICS TS V2.3 region; 64 if you are installing CICS BAC in a CICS TS V3.1 region; 65 if you are installing CICS BAC in a CICS TS V3.2 region; 66 if you are installing CICS BAC in a CICS TS 4.1 region; 67 if you are installing CICS BAC in a CICS TS V4.2 region; or 68 if you are installing CICS BAC in a CICS TS V5.1 region. You can also define the programs for all these CICS releases if you want to.
2. In the CEDA commands shown in Figure 4, CBKMT*xxx* represents some optional language modules that you can use in CICS BAC. Always define the module CBKMTENU. You should also define any other language modules for languages that are supported by your CICS region and supported by corresponding CICS BAC CBKMT*xxx* modules. For more information, see Appendix B, “CICS BAC national language support (NLS),” on page 189.
3. As an alternative to the CEDA commands shown in Figure 4, you can use the CICS BAC sample JCL in the SCBKSAMP data set member, CBKPROGS, to define the programs and mapset using the CICS DFHCSDUP utility.

CICS BAC control file definition

About this task

Define the CICS BAC control file using the sample CEDA command shown in Figure 5.

```
CEDA DEFINE GROUP(cbkgroup) FILE(CBKCNTL) DSNAME(hlq.control.file) LSRPOOLID(NONE)
```

Figure 5. CEDA command to define the CICS BAC control file in your CSD

Note:

1. In the CEDA commands shown in Figure 5 on page 16, replace *cbkgroup* with any character string allowed by the CEDA transaction that meets your installation standards. Replace the nn in the program names CBKFSXnn and CBKSPXnn with 53 if you are installing CICS BAC in a CICS TS 1.3 region; 62 if you are installing CICS BAC in a CICS TS 2.2 region; 63 if you are installing CICS BAC in a CICS TS 2.3 region; 64 if you are installing CICS BAC in a CICS TS 3.1 region; 65 if you are installing CICS BAC in a CICS TS 3.2 region; or 66 if you are installing CICS BAC in a CICS TS 4.1 region; 67 if you are installing CICS BAC in a CICS TS V4.2 region; or 68 if you are installing CICS BAC in a CICS TS V5.1 region. You can also define the programs for all these CICS releases if you want to.
2. As an alternative to the CEDA command shown in Figure 5 on page 16, you can use the CICS BAC sample JCL in the SCBKSAMP data set member, CBKFILES, to define the control file using the CICS DFHCSDUP utility.

Installing the CICS BAC resource definitions

About this task

When you have defined the CICS BAC transaction, programs, and file resources, you can install the definitions in the running CICS region using the CEDA INSTALL command:

```
CEDA INSTALL GROUP(cbkgroup)
```

where *cbkgroup* is the group name you specified for the CICS BAC resource group in the CSD.

To install the CICS BAC resources automatically during CICS initialization, add your CICS BAC group name to a CICS startup list using the CEDA ADD command:

```
CEDA ADD GROUP(cbkgroup) LIST(listname)
```

where *cbkgroup* is the group name you specified for the CICS BAC resource group in the CSD, and *listname* is the name of the startup list to which you want to add the group name.

Alternatively, you can edit the ADD GROUP command in one of the sample DFHCSDUP jobs, such as CBKFILES, to specify an appropriate group name.

Other CICS services used

About this task

CICS BAC also uses following CICS services. These services are:

- **Global user exits**: CICS BAC uses the XFCSREQC global user exit to track state changes to VSAM files originating from CEMT or SPI commands. It also uses the XEISPOUT exit to track state change requests for transient data queues, programs, and transactions originating from CEMT or SPI commands. These two exits are automatically enabled by CICS BAC startup processor.
- **External CICS Interface (EXCI)**: CICS BAC uses EXCI to ship communication server requests and batch request utility requests to a target CICS region for processing. Ensure that you either (1) install the CICS-supplied sample resource definition group, DFH$EXCI, or (2) create and install similar definitions of your own, to provide the connection resource definitions that you need to support EXCI connections. Note that, if you create your own EXCI definitions, CICS BAC requires a generic connection definition. See the *CICS External Interfaces Guide* for information about defining CICS resource definitions for EXCI support.

In addition to its own comprehensive security mechanisms to protect CICS BAC resources (see Chapter 9, “CICS BAC security,” on page 123), CICS BAC also uses CICS external interface (EXCI) security; see “CICS BAC support for CICS EXCI security” on page 128.

- **CICS temporary storage**: CICS BAC uses a CICS temporary storage queue to save CICS BAC control information pertaining to the CICS region in which it is running. The name of the CICS BAC temporary storage queue is CBKCTLQ1. There is one queue by this name in each CICS region where CICS BAC has been started since the last time the region was started. The queue is not deleted if CICS BAC is shut down in the region. It exists until the CICS region terminates. Also, the queue is region specific, and you should ensure that:
  - The queue is accessed by only one CICS region; it must not be shared by two or more CICS regions
  - The queue is local to the region in which CICS BAC is running. You must not use CICS function shipping to access the queue remotely from another region, such as a queue owning region (QOR).
  - The queue is not set up as a shared queue that can be shared by other CICS regions in a CICSPlex.
  - The queue is not defined as a recoverable temporary storage queue.

Starting CICS BAC in a CICS region

About this task

There are two methods by which you can start CICS BAC in a CICS region, which are discussed in the following topics:

- “Using the CICS BAC PLTPI program”
- “Using the KBKM transaction” on page 19

Using the CICS BAC PLTPI program

About this task

You can use the CICS BAC command processor program, CBKCMNDS, to start CICS BAC during CICS region initialization. To do this, add an entry to the CICS region PLTPI table. Figure 6 shows a partial DFHPLT table containing the DFHDELIM and CBKCMNDS entries.

```
DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
DFHPLT TYPE=ENTRY,PROGRAM=CBKCMNDS
```

Figure 6. DFHPLT entry for starting CICS BAC during CICS initialization

As shown in Figure 6, ensure you add the statement for CBKCMNDS immediately after the DFHDELIM statement. This ensures it executes before any programs that could otherwise attempt to use resources whose state can be altered by CICS BAC startup processing.

The CICS BAC SCBKSAMP data set member, CBKPLTPI, contains a sample PLTPI member.

If you choose to use the PLTPI to activate CICS BAC, you can pass the CICSSTARTMODE parameter to CBKCMNDS for the startup process using the CICS INITPARM system initialization parameter. The following is an example of an INITPARM system initialization parameter for CBKCMNDS:

```
INITPARM=(CBKCMNDS='CICSSTARTMODE(UPDATE)')
```

The CICSSTARTMODE parameter is described fully under "CICS region commands for CICS BAC." Note that if you omit the INITPARM system initialization option, the CICSSTARTMODE parameter defaults to CICSSTARTMODE(SET) when you start CICS BAC using the PLTPI program. Under the SET option, startup processing reads the records of all resources defined in the region control file and applies their last requested state to the installed CICS resource definitions. As CICS BAC sets each required state in the CICS region, it updates the region control file with information indicating that the state has been set in the CICS region. This action is safe when CICS BAC starts during the PLTPI processing phase, and by the time control is given to CICS, all installed CICS resources that are defined in the region control file are set to their last known state.

Using the KBKM transaction

About this task

You can also start CICS BAC in your CICS regions using the commands provided by the KBKM transaction. The KBKM commands are described under "CICS region commands for CICS BAC"

Shutting down CICS BAC during CICS shutdown

About this task

You can shut down CICS BAC support automatically during CICS shutdown processing. To do this, add an entry to the region PLT shutdown table as follows:

```
DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
DFHPLT TYPE=ENTRY,PROGRAM=CBKCMNDS
```

Figure 7. DFHPLT entry for shutting down CICS BAC during CICS shutdown

Make the entry for the CICS BAC PLT program, CBKCMNDS, the last entry in the PLT to ensure correct processing.

You can also shut down the CICS BAC component using the KBKM transaction (see "CICS region commands for CICS BAC"

CICS region commands for CICS BAC

About this task

This section describes the syntax and purpose of the commands that you can use to control the CICS BAC component that runs in a CICS region. You enter these commands from a 3270 terminal connected to a CICS region. The output destination for these commands is the terminal from which you entered the command. For some of these commands, the CICS BAC component also displays messages on the system console.

The general format for CICS BAC request server commands is:

KBKM *COMMAND*,parm(*value*)

where KBKM is the CICS BAC command processor transaction ID, *COMMAND* is one of the CICS BAC component commands; and parm(value) is an optional parameter specified on the command. If a value is italicized, it means that you can specify one of a range of values. If a value is not italicized, you specify the value

exactly as shown. The syntax diagrams used for these commands is in the same form as the batch request utility commands (see “Syntax notation for batch request utility commands” on page 46)

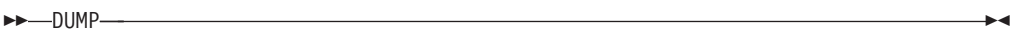
When you enter a KBKM command with a parameter, ensure the parameter is separated from the command name by a comma or space.

DUMP
Purpose

Capture a CICS BAC component dump.

Format

DUMP



Example: KBKM DUMP

Usage

Use the DUMP command to capture a dump of the CICS BAC component data areas. The dump is directed to the current CICS dump data set.

Note: You might be asked to use this command to assist a CICS BAC Technical Support representative with problem diagnosis.

There are no parameters on the DUMP command.

SHUTDOWN
Purpose

Shut down the CICS BAC component in the CICS region.

Format

SHUTDOWN

Example: KBKM SHUTDOWN

Usage

When you issue the SHUTDOWN command, from a CICS terminal, the CICS BAC component in the region shuts down. When shutdown is completed, the CICS region cannot service any batch request utility or communication server requests. All resources are released, and the CICS BAC global user exit programs are disabled.

Note: Note: The effect of SHUTDOWN command is as though the CICS BAC component had never been started. To enable CICS BAC to resume accepting and processing batch request utility and communication server requests, you must restart the CICS BAC component.

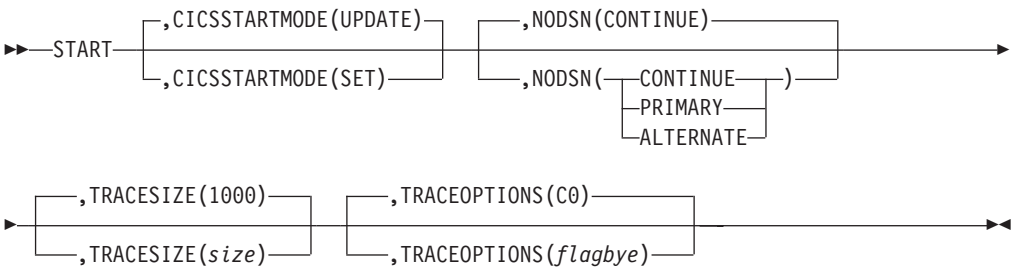
There are no parameters on the SHUTDOWN command.

START
Purpose

Start the CICS BAC component in a CICS region.

Format

START



Example: KBKM START,CICSSTARTMODE(SET)

Usage

Use this command to start the CICS BAC component in a CICS region. As soon as CICS BAC is active, the CICS region can accept and process batch request utility and communication server requests.

Note: The KBKM START command is an alternative to starting CICS BAC using the PLTPI program, CBKCMNDS (see "Using the CICS BAC PLTPI program" on page 18 for details). If you use the PLT method, you can use the CICS system initialization parameter INITPARM to provide the startup options.

Parameters

CICSSTARTMODE({UPDATE | SET})
Specifies the processing mode that to be used for the start command.

UPDATE
Startup processing does not set the state of any resources during startup processing. It updates the region control file only with the current state of known resources.

Note:
This is the default value only when you are using the KBKM START command. When you start CICS BAC at region startup during PLTPI processing, the default value for the CICSSTARTMODE parameter is SET.

SET Startup processing reads the records of all resources defined in the region control file and applies their last requested state to the installed CICS resource definitions. As CICS BAC sets each required state in the CICS region, it updates the region control file with information indicating that the state has been set in the CICS region.

Note: Be aware when using the SET option that starting CICS BAC after control has been given to CICS could mean that some resources

are open and in use by CICS, contrary to their last requested state in the CICS region control file. Using SET in these circumstances could have undesirable consequences.

NODSN({CONTINUE | PRIMARY | ALTERNATE})
Specifies the action taken by the CICS BAC request server when it is about to attempt to open a file in the CICS region and the file does not yet have a data set name associated with it. This can occur either during startup processing or when processing a CICS BAC batch request utility SET FILE command. Note that this parameter is ignored if the file is currently allocated to the CICS region; for example, if it is defined by a DD statement in the CICS region JCL.

CONTINUE
OPEN processing continues without attempting to alter the file data set name beforehand. This typically results in an open failure unless you use some other method to provide the data set name before the operating system attempts to open the file.

PRIMARY
CICS BAC sets the file data set name, as part of the OPEN request, to the primary data set name as specified in the CICS BAC control file record for the file being opened. If the primary data set name is blank, the OPEN request has the same result as if you had specified CONTINUE.

ALTERNATE
CICS BAC sets the file data set name, as part of the OPEN request, to the alternate data set name as specified in the CICS BAC control file record for the file being opened. If the alternate data set name is blank, the OPEN request has the same result as if you had specified CONTINUE.

TRACEOPTIONS({C0 | options})
Specifies the types of trace entries you want CICS BAC to create. There are eight trace entry types, each controlled by a single bit in the TRACEOPTIONS hexadecimal byte flag. Specify this option only at the direction of a Technical Support representative. For details of the hexadecimal byte flag, see the TRACEOPTIONS parameter under "Communication server runtime parameters" on page 27

C0 The hexadecimal value C0 is the default trace setting.

options
Specify the hexadecimal byte flag that represents the type of trace entries CICS BAC is to create, as directed by a Technical Support representative.

TRACESIZE({1000 | number-of-entries})
Specifies the size of the trace table in terms of the number of entries to be traced for diagnostic purposes. Change this value only at the direction of Technical Support.

1000 The default trace size.

number
Specify the trace table size as requested by Technical Support.

STATUS
Purpose

Display the current status of the CICS BAC component in the CICS region.

Format

STATUS

Example: KBKM STATUS

Usage

The CICS BAC component displays its response on the CICS terminal at which you entered the STATUS command. The possible status values displayed are:

- Active
- Shutting down
- Shut down
- Never started.

Additional information is displayed, depending on the status of CICS BAC in the region.

Defining CICS BAC data sets

About this task

The communication server, the ISPF administration interface, the file maintenance utility, and the batch request utility all require access to the control file table (CBKCFTBL). The communication server and batch request utility optionally require a CBKPARMS member to provide their runtime parameters. You can define the communication server parameters in a member called CBKSRVR, and the batch request utility parameters in a member called CBKBATCH. CICS BAC also uses an audit log to write audit messages.

You define the control file table, the parameters data set, and an audit log, as described in the following topics:

- "The control file table"
- "The CBKPARMS data set" on page 24
- "The audit log data set" on page 25

The control file table

About this task

The CICS BAC communication server, the ISPF administration interface, the file maintenance utility, and the batch request utility all need to know the fully-qualified data set name of the control file of each CICS region. This is so that these tasks can allocate and access the appropriate control file in the event that CICS BAC is not active in the owning CICS region, or the owning CICS region is down. In the normal course of events, the communication server, the ISPF administration interface, the file maintenance utility, and the batch request utility read and update the control file through the CICS BAC request server task in the CICS region. To define the corresponding data set name for each CICS region, enter the details in a partitioned data set member named CBKCFTBL, in the following format:

CICS APPLID
Specify the CICS region APPLID in the first 8 bytes, beginning in column 1, padded with trailing spaces as required.

Separator character
Specify the separator character in column 9 as a space. This is required only for readability but must be present.

Data set name
Specify the fully-qualified 44-byte data set name in columns 10 to 53.

For example:

```
+---+----1----+----2----+----3----+----4----+----5---
CICSHUR1 CICSHUR1.CICSAOR1.CBKCNTLF
```

There is a sample control file table in SCBKSAMP, member name CBKCFTBL, that you can modify with your own names.

Note: The control file table member name must be CBKCFTBL. This can be a member of:

- A CICS BAC parmlib data set that is defined to the communication server or the batch request utility on a CBKPARMS DD statement
- Any data set that is included in the MVS logical parmlib concatenation.

When a CICS BAC communication server task or a CICS BAC batch request utility job needs to allocate a CICS region control file, it attempts to find the control file table in one of the following places:

- The task or job first checks for a CBKPARMS DD statement in its JCL. If there is a CBKPARMS statement, the task or job attempts to find a member named CBKCFTBL. If it finds a CBKCFTBL member, it uses that member and doesn't look anywhere else. If it is found to be an empty member, CICS BAC issues an error message and terminates the communication server request or batch request utility job step.
- If there is not a CBKPARMS DD statement, or the CBKPARMS partitioned data set does not contain a CBKCFTBL member, CICS BAC searches the MVS logical parmlib concatenation for the CBKCFTBL member. If it cannot be found, or is found but is empty, CICS BAC generates an error message and terminates the communication server request or batch request utility job step.

The CBKPARMS data set

About this task

If you decide to provide runtime parameters for the communication server and the batch request utility in a CBKPARMS member, define a partitioned data set for these members with the following attributes:

- Record length (LRECL)=80
- Record format (F) or (FB) for fixed-length unblocked or fixed-length blocked, respectively.
- Block size (BLKSIZE) as an exact multiple of 80.
- Data set organization (DSORG)=PO
- Directory blocks must be a minimum of 1.

The audit log data set

About this task

The CICS region and the batch request utility are both able to use the audit log, DD name CBKLOG, to record the results of processing batch request utility requests. In the case of the batch request utility, it uses the audit log only when it processes requests directly itself because the target CICS region is not available. The level of audit logging, and whether it takes place at all, depends on the options you specify in the CICS region properties object record. Also, unless you specify that log records are written to SYSOUT, you need to define an audit log data set that is referenced by the CBKLOG DD statement. Define your log data set with the following attributes:

- Record length (LRECL) as 133
- Record format (RECFM) as FBA
- Block size (BLKSIZE) as an exact multiple of 133 (the LRECL).
- Data set organization (DSORG) as PS.

You can include a DD statement for CBKLOG in the CICS region JCL or in the CICS BAC batch request utility JCL to allocate the log data set at job step initiation. Alternatively, you can let CICS BAC dynamically allocate the audit log when it is first required, based on the audit log attributes in the CICS region properties record in the region control file. You can define, view, or alter the audit log attributes using the workstation administration client. In the workstation administration client, open the Region Properties window and select the Miscellaneous(1) options tab. The Miscellaneous(1) options tab allows you to direct the audit log to SYSOUT, or to a named data set, and to indicate the types of audit record you want CICS BAC to write to the log. You can also specify the logging attribute using the LOG*xxxx* parameters of the file maintenance utility (see the “UPDATE REGION” on page 100 command on page “UPDATE REGION” on page 100).

Setting up and starting the communication server

About this task

This topic describes how you setup, start, and run a CICS BAC communication server. It provides the sample JCL that you can use and information about the data sets and parameters used by the communication server. Figure 8 on page 26 contains an example of the JCL you use to submit the communication server as an MVS started task.

Note that you need a communication server only if you plan to use the CICS BAC workstation administration client. If you perform all administration tasks by using either the ISPF administration interface or the file maintenance utility, you do not need a CICS BAC communication server.

```
//CBKCSRVR  PROC
//CBKCSRVR EXEC PGM=CBKSMAIN,REGION=0M
//STEPLIB  DD DISP=SHR,DSN=hlq.SCBKLOAD
//         DD DISP=SHR,DSN=hlq.SDFHEXCI
//CBKPARMS DD DISP=SHR,DSN=hlq.CBKPARMS
//SYSABEND DD SYSOUT=*
//************************************************
//* Un-comment the following DD statement if you need
//* to add a SYSMDUMP DD statement. You might need to
//* modify this statement to meet your installation's
//* requirements.
//************************************************
//*SYSMDUMP  DD DISP=(,CATLG),DSN=hlq.sysmdump.dataset,
//*          UNIT=SYSDA,SPACE=(CYL,(50,25)),
//*          DCB=(DSORG=PS,RECFM=FBS,LRECL=4160,
//*          BLKSIZE=24960
```

Figure 8. CICS BAC communication server sample startup procedure

Note:

1. Change the characters *hlq* to the high-level qualifiers you specified for these data sets:
   - SCBKLOAD is the name of the CICS BAC load library
   - SDFHEXCI is the name of the CICS TS EXCI load library
   - CBKPARMS is the name of your CICS BAC parmlib.
2. Also change *hlq.sysmdump.dataset* to the name of your own dump data set.
3. When you have customized CBKCSRVR, add the procedure SYS1.PROCLIB or another suitable procedure library,
4. If you put the CBKCFTBL member and the optional CBKSRVR parameter member in the MVS logical parmlib concatenation, authorize the communication server task to have READ access to all the data sets in the MVS logical parmlib concatenation. The communication server searches for CBKCFTBL and CBKSRVR in the logical parmlib concatenation if either of these members is not found in the CBKPARMS data set.

Checklist

About this task

The following is a list of the tasks necessary to setup a CICS BAC communication server:

- Customize the sample communication server started task procedure
- Ensure that your SCBKLOAD data set and SDFHEXCI data set are APF-authorized or are included in the linklist concatenation if you want the communication server to perform security checking for workstation requests.
- Ensure the CBKCFTBL member is available in either the CBKPARMS data set, or a data set in the MVS logical parmlib concatenation. For information about defining the CICS BAC control file table, see "The control file table" on page 23.
- Specify the runtime parameters if you want other than the default values (see "Communication server runtime parameters" on page 27).
- Ensure the parameters that the communication server needs are provided in the CBKSRVR member in the CBKPARMS data set, or another parmlib data set in the MVS logical parmlib concatenation.

- Ensure that the communication server ID (that is, the user ID that the server runs under) is authorized to access TCP/IP services. If the server ID is not authorized to your external security manager, the communication server startup fails.
- Start the communication server started task or job.

You will find a sample JCL member called CBKCSRVR, in the CBKSAMP library, that you can use to create your startup procedure .

Communication server runtime parameters

About this task

There are several parameters that you can pass to the communication server at its startup, all of which are optional. You can specify communication server parameters using any combination of the following methods:

- Allow the communication server to use its built-in defaults.
- Define your own values in CBKSRVR, which can be a member of either a CBKPARMS data set, or a data set in the MVS logical parmlib concatenation.
- Define your own values in the PARM string of the EXEC JCL statement.

The communication server searches for startup parameters, and processes them, in the following way:

1. **Default values**: CBKSMAIN first sets the default values for each parameter and continues at step 2.
2. **CBKPARMS data set**: The communication server searches for a CBKSRVR member in the CBKPARMS data set, if present. If the server finds a CBKSRVR member it uses any runtime parameters in the CBKSRVR member to override the defaults. If the member is empty, the defaults are not changed. When the communication server has finished processing the CBKSRVR member, it continues at step 4.
3. **Logical parmlib concatenation**: If the communication server cannot find a CBKSRVR member in CBKPARMS, it searches for a CBKSRVR member in the MVS logical parmlib concatenation. If the server finds a CBKSRVR member it uses any runtime parameters in the CBKSRVR member to override the defaults. If the member is empty, the defaults are not changed.
4. **PARM string**: The communication server reads any parameters specified in the PARM string of the JCL EXEC statement, and uses these to override default or CBKSRVR parameter values.

Note: If you want to use all default values, or specify parameters in the PARM string only, do one of the following:

- Include an empty CBKSRVR member in the CBKPARMS data set. The communication server skips to step 4 above when it finds an empty CBKSRVR member.
- Authorize the communication server task to have READ access to all the data sets in the MVS logical parmlib concatenation. The communication server continues with step 4 if it cannot find a CBKSRVR member. If the communication server does not have at least READ access to the logical parmlib concatenation, the server fails with a 913 abend.

All parameters can be truncated to their shortest unique character string. However, be aware that a unique abbreviation for this release of CICS BAC might not be

unique in future releases. Therefore, you should not use the shortest abbreviation for a parameter, especially when the parameter is being stored in many different data sets.

CBKID(*name***)**
Specifies the name for the communication server as a 1-8 alphanumeric character name. Ensure that each communication server that you start has a unique name within the MVS system. If you omit this parameter, the name defaults to the job name of the communication server address space.

CICSMIRRORTRANS({CSMI | *transid*})
Specifies the name of the CICS mirror transaction to be specified on EXCI requests issued by the communication server. When the EXCI request is executed in the target CICS region, it runs under this transaction ID as an MRO mirror transaction.

CSMI This is the normal CICS EXCI default mirror transaction ID.

transid Specifies the 4-character name that you want CICS to use as the name of the MRO mirror transaction when it executes the EXCI requests it receives from the communication server. To avoid confusion with other mirror transactions that might be running in your CICS regions, you might want to choose a unique ID to identify CICS BAC communication server mirrors. Define the appropriate transaction ID in the CICS region for a CICS mirror transaction, ensuring the following:

- The program associated with the transaction ID must be DFHMIRS
- The profile for the transaction ID must be DFHCICSA.

You might consider using the CICS-supplied CSMI transaction definition as a basis for this transaction ID.

LANGUAGE({ENU | JPN})
Specifies the language used by the communication server for console and job log messages. See Appendix B, "CICS BAC national language support (NLS)," on page 189 for more information. The values you can specify are:

ENU US English

JPN Japanese

Note: Specify the language parameter only on the PARM string of the JCL EXEC statement for the communication server. The parameter is ignored if it is specified in the CBKSRVR member of either the CBKPARMS data set or the logical parmlib concatenation.

TCPPORTNUM({56677 | number})
Specifies the TCP/IP port number that the communication server is to use to receive TCP/IP requests from the workstation client.

56677 The default port number.

number
Specify the TCP/IP port number in the range 1 to 65535.

TCPTIMEOUT({60 | number-of-seconds})
Specifies the length of time in seconds after which the communication server times out a request if it has not received additional data.

60 The default is 60 seconds.

seconds
Specify the time-out value in seconds in the range 1 to 99.

TRACEOPTIONS({C0 | options})
Specifies the types of trace entries you want CICS BAC to create. There are eight trace entry types, each controlled by a single bit in the TRACEOPTIONS hexadecimal byte flag. Specify this option only at the direction of a Technical Support representative.

The meaning of each bit position is as follows (bits from left to right):

Bit position	Type of trace entry
0	Module entry
1	Module exit
2	Module flow
3	Module loop
4	Module call
5	Reserved
6	Reserved
7	Reserved

C0 The hexadecimal value C0 is the default trace setting.

options
Specify the hexadecimal byte flag that represents the type of trace entries CICS BAC is to create, as directed by a Technical Support representative.

TRACESIZE({1000 | number-of-entries})
Specifies the size of the trace table in terms of the number of entries to be traced for diagnostic purposes. Change this value only at the direction of Technical Support.

1000 The default trace size.

number
Specify the trace table size as requested by Technical Support.

Communication server operator commands

About this task

The syntax for all communication server commands is:
`{MODIFY|F} jobname,COMMAND`

where MODIFY is the MVS console command (and F is the abbreviation); *COMMAND* is one of the communication server commands.

The communication server returns the output from your console MODIFY commands to the operator console (system log).

DISPLAYPARMS

Function

To display the communication server startup parameters.

Syntax

DISPLAYPARMS

Example: MODIFY *jobname*,DISPLAYPARMS

Usage

You can use the DISPLAYPARMS command to display the parameter values that the communication server uses for its initialization. The information the communication server displays on the console includes the parameter name and parameter value. It also displays the source of the startup parameter, as follows:

- **EXEC PARM** indicating the parameter was taken from the PARM string of the JCL EXEC statement in the communication server startup procedure.
- **CBKPARMS** indicating the parameter was taken from the CBKPARMS data set, in member CBKSRVR
- **PARMLIB** indicating the parameter was taken from the MVS logical parmlib concatenation
- **Default** indicating a default value.

The response to the DISPLAYPARMS command is displayed in messages CBK*xx*1200I and CBK*xx*1201I.

Note that the communication server displays the startup parameters automatically during initialization as if you has issued the command.

There are no parameters on the DISPLAYPARMS command.

SHUTDOWN
Function

Shutdown the communication server

Syntax

SHUTDOWN

Example: MODIFY *jobname*,SHUTDOWN

Usage

When you issue the SHUTDOWN command the communication server performs a normal shutdown.

There are no parameters on the SHUTDOWN command.

Setting up and running a CICS BAC batch request utility

About this task

You can invoke the CICS BAC batch request utility program as a job step in one of your regular batch jobs, where you need obtain access to resources that might be allocated and open in a CICS region. For example, if you have a routine batch job that needs to open for update a data set that is typically open and online in a CICS region, you can include a job step that invokes the CICS BAC batch request utility, and through the utility pass a request to the CICS BAC request server to close the data set that is open as a CICS file. The following steps summarize what happens in this example:

- Your CICS BAC job step passes a CICS BAC SET FILE(*filename*) OPENSTATUS(CLOSED) ENABLESTATUS(DISABLED) command as an EXCI DPL request to the CICS target region. The region can either be specified explicitly on the command or it can be the default region set in an earlier CICS BAC DEFAULT command.
- CICS receives the DPL request and executes the CICS BAC request server program to process the command, which results in an EXEC CICS SET FILE CLOSED DISABLED command. Note that the disabled option ensures that the file cannot be opened implicitly by a CICS transaction issuing an EXEC CICS READ command, and it must be explicitly enabled before it can be used again by CICS.
- Access to the CICS BAC control file is handled by the CICS BAC request server subtask (CBKCRHST), which updates the status of the file record in the control file to CLOSED DISABLED.
- The CICS BAC request server returns the results of the command to the CICS BAC batch request utility. If the command has executed successfully, your batch job can continue with the main job step to process the file.

CICS BAC batch request utility job step

About this task

A CICS BAC batch request utility job step requires the following statements in the JCL:

- An EXEC statement for program CBKBMAIN (note the program name must be CBKBMAIN).
- A DD statement for the CBKPARMS data set that contains:
  - The control file table member, CBKCFTBL, that identifies the control file data set name for each CICS region
  - The parameter member, CBKBATCH, that contains the batch request utility runtime parameters

  Alternatively, you can include the CBKCFTBL and CBKBATCH members in a data set that is in the MVS logical parmlib concatenation, in which case you do not need the CBKPARMS DD statement. However, if you put the CBKCFTBL member and the optional CBKBATCH parameter member in the MVS logical parmlib concatenation, authorize the batch request utility to have READ access to all the data sets in the MVS logical parmlib concatenation. The batch request utility searches for CBKCFTBL and CBKBATCH in the logical parmlib concatenation if either of these members is not found in the CBKPARMS data set.

- A STEPLIB DD statement that references the SCBKLOAD and the SDFHEXCI data set containing the load modules required by the batch request utility if the data sets are not in the MVS linklist.
- A DD statement for the CBKIN data set, from which the CICS BAC batch request utility reads the commands it is to execute.
- A DD statement for the CBKPRINT data set, to which the CICS BAC batch request utility writes its output messages.

The sample JCL in Figure 9 illustrates a CICS BAC batch request utility job step.

```
//CBKBTCH  EXEC PGM=CBKBMAIN
//STEPLIB  DD DISP=SHR,DSN=hlq.SCBKLOAD
//         DD DISP=SHR,DSN=hlq.SDFHEXCI
//CBKPARMS DD DISP=SHR,DSN=hlq.CBKPARMS
//CBKGROUP DD DISP=SHR,DSN=your.cicsgrps.dataset
//SYSABEND DD SYSOUT=*
//*****************************************************************
//* Un-comment the following DD statement if you need to add a
//* SYSMDUMP DD statement. You might need to modify this statement
//* to meet your installation's requirements.
//*****************************************************************
//*SYSMDUMP DD DISP=(,CATLG),DSN=hlq.sysmdump.dataset,
//*         UNIT=SYSDA,SPACE=(CYL,(50,25)),
//*         DCB=(DSORG=PS,RECFM=FBS,LRECL=4160,
//*         BLKSIZE=24960
//CBKPRINT  DD SYSOUT=*//CBKIN     DD *
DEFAULT CICS(CICSHUR1),NOTACTIVE(TERMINATE)
SET FILE(FILEA),OPENSTATUS(CLOSED),ENABLESTATUS(DISABLE)
```

Figure 9. Sample JCL for a CICS BAC batch request utility job step

Note:

1. Change the characters *hlq* to the high-level qualifiers for these data sets:
   - SCBKLOAD is the name of the CICS BAC load library
   - SDFHEXCI is the name of the CICS TS EXCI load library
   - CBKPARMS is the name of your CICS BAC parmlib.
2. Also change *hlq.sysmdump.dataset* to the name of your own dump data set.
3. If you put the CBKCFTBL member and the optional CBKBATCH parameter member in the MVS logical parmlib concatenation, authorize the batch request utility to have READ access to all the data sets in the MVS logical parmlib concatenation.
4. See "The CBKPARMS data set" on page 24 for information about defining the CBKPARMS data set.
5. The CBKGROUP DD statement is optional. See "CICS group and alias function" on page 117 for more information.

There is some sample JCL for a batch request utility job step shown in member CBKSAMP1, which you can find in the SCBKSAMP library. You can copy this and modify it for your own use.

CICS BAC batch request utility runtime parameters

About this task

There are several runtime parameters that you can pass to the batch request utility at its startup, all of which are optional. You can specify these parameters using any combination of the following methods:

- Allow the batch request utility to use its built-in defaults.
- Define you own values in CBKBATCH, which can be a member of either a CBKPARMS data set, or a data set in the MVS logical parmlib concatenation.
- Define you own values in the PARM string of the EXEC JCL statement.

The batch request utility searches for startup parameters, and processes them, in the following way:

1. **Default values**: The batch request utility first sets the default values for each parameter and continues at step 2.
2. **CBKPARMS data set**: The batch request utility searches for a CBKBATCH member in the CBKPARMS data set, if present. If the utility finds a CBKBATCH member it uses any runtime parameters in the member to override the defaults. If the member is empty, the defaults are not changed. When the batch request utility has finished processing the CBKBATCH member, it continues at step 4.
3. **Logical parmlib concatenation**: If the batch request utility cannot find a CBKBATCH member in CBKPARMS, it searches for a CBKBATCH member in the MVS logical parmlib concatenation. If the utility finds a CBKBATCH member it uses any runtime parameters in the member to override the defaults. If the member is empty, the defaults are not changed.
4. **PARM string**: The batch request utility reads any parameters specified in the PARM string of the JCL EXEC statement, and uses these to override default or CBKBATCH parameter values.

Note: If you want to use only default values, or specify parameters in the PARM string only, do one of the following:

- Include an empty CBKBATCH member in the CBKPARMS data set. The batch request utility skips to step 4 above when it finds an empty CBKBATCH member.
- Authorize the batch request utility to have READ access to all the data sets in the MVS logical parmlib concatenation. The batch request utility continues with step 4 if it cannot find a CBKBATCH member. If the batch request utility does not have at least READ access to the logical parmlib concatenation, the job step fails with a 913 abend.

The following are the runtime parameters that you can specify when starting a batch request utility job step:

CICSGROUPDSN(*datasetname***)**
Specifies the name of the PDS in which CICS BAC is to look for member names that match CICS(*applid*) names. You can use CICS group data set members set to redirect CICS BAC commands to a different CICS region or to a group of CICS regions. See "CICS group and alias function" on page 117 for more information about this parameter. You can override this parameter in the batch request utility job step JCL using the CBKGROUP DD statement.

CICSMIRRORTRANS({CSMI | *transid*})
Specifies the name of the CICS mirror transaction to be specified on EXCI

requests issued by the batch request utility. When the EXCI request is executed in the target CICS region, it runs under this transaction ID as an MRO mirror transaction.

CSMI This is the normal CICS EXCI default mirror transaction ID.

transid Specifies the 4-character name that you want CICS to use as the name of the MRO mirror transaction when it executes the EXCI requests it receives from the batch request utility. To avoid confusion with other mirror transactions that might be running in your CICS regions, you might want to choose a unique ID to identify CICS BAC batch request utility mirror transactions. Define the appropriate transaction ID in the CICS region for a CICS mirror transaction, ensuring the following:

- The program associated with the transaction ID must be DFHMIRS
- The profile for the transaction ID must be DFHCICSA.

You might consider using the CICS-supplied CSMI transaction definition as a basis for this transaction ID.

LANGUAGE({ENU | JPN})
Specifies the language used by the batch request utility for console, job log, and CBKPRINT messages. See Appendix B, "CICS BAC national language support (NLS)," on page 189 for more information. The values you can specify are:

ENU US English

JPN Japanese

Note: Specify the language parameter only on the PARM string of the JCL EXEC statement for the batch request utility. The parameter is ignored if it is specified in the CBKBATCH member of either the CBKPARMS data set or the logical parmlib concatenation.

TRACEOPTIONS({C0 | options})
Specifies the types of trace entries you want CICS BAC to create. There are eight trace entry types, each controlled by a single bit in the TRACEOPTIONS hexadecimal byte flag. Specify this option only at the direction of a Technical Support representative.

C0 The hexadecimal value C0 is the default trace setting.

options
Specify, in hexadecimal form, the options requested by Technical Support. For more details of the hexadecimal byte flag, see the TRACEOPTIONS parameter under "Communication server runtime parameters" on page 27

TRACESIZE({1000 | number-of-entries})
Specifies the size of the trace table in terms of the number of entries to be traced for diagnostic purposes. Change this value only at the direction of Technical Support.

1000 The default trace size.

number
Specify the trace size as requested by Technical Support.

Setting up and starting the ISPF administration interface

About this task

The primary component of the ISPF administration interface is the REXX EXEC that you use to start the interface, and the first step in setting up the function is to customize the EXEC, followed by a number of other customization tasks. These are discussed in the following sections:

- "Customizing the REXX EXEC"
- "Starting the REXX EXEC" on page 38
- "Tailoring existing ISPF menus" on page 38
- "Specifying the control file table data set" on page 39
- "Selecting a target CICS region" on page 40
- "Help and tutorial facilities" on page 41
- "ISPF administration interface messages" on page 41

Specifying access to a CICS region control file

About this task

As discussed under "The control file table" on page 23, the ISPF administration interface is one of the CICS BAC components that needs access to the CICS BAC control file of each CICS region. However, unlike its counterpart, the workstation administration client, the ISPF administration interface does not use the services of the communication server. Instead, the ISPF administration interface obtains access to the appropriate control file in one of the following ways:

- Indirectly, through the owning CICS region by means of the CICS external communication interface (EXCI). In the normal course of events, the ISPF administration interface reads and updates the control file through the CICS BAC request server task in the CICS region, with which it communicates through EXCI requests. To enable the ISPF administration interface to have access to the EXCI function, ensure the EXCI load library, SDFHEXCI, is available to the interface. This means it must either be in the LINKLIST or included in the TSO logon PROC used by users of the ISPF administration interface.

  Note: If you add the SDFHEXCI load library to the logon PROC STEPLIB concatenation, it might have to be APF-authorized, depending on your system requirements. CICS BAC does not require SDFHEXCI to be APF-authorized, but it is possible that other TSO applications require all data sets in the STEPLIB concatenation to be APF-authorized.
- Directly, by allocating the VSAM data set data in the event that CICS BAC is not active in the owning CICS region, or the owning CICS region is down.

If access to EXCI fails because the SDFHEXCI load library is not in either the linklist or the user TSO logon PROC, the ISPF administration interface fails with a system abend code 806, reason code 04.

Customizing the REXX EXEC

About this task

You start the CICS BAC ISPF administration interface by invoking by a REXX EXEC, which is supplied as member CBKRCBAC in the SCBKEXEC data set. Before you attempt to use the CICS BAC administration interface, first customize

the REXX EXEC. When you customize the EXEC, it is important that you apply your customization changes to a copy of CBKRCBAC, and do not change the supplied CBKRCBAC member in SCBKEXEC. If you modify the supplied version of CBKRCBAC in data set SCBKEXEC, your changes could be overwritten by any future maintenance to the CICS BAC-supplied member. You can use one of the following methods to avoid modifying the original:

- Create a copy of CBKRCBAC with a new name in the SCBKEXEC data set, choosing a name that suits your own naming standards, such as CBKRCUST
- Create a copy of CBKRCBAC in a different REXX EXEC data set that you plan to use for the ISPF administration interface, and keep the name of the copy as CBKRCBAC

Where relevant, the discussion in this manual is based on the first of these methods using the EXEC name CBKRCUST and the data set name SCBKEXEC.

When you have created a copy of CBKRCBAC, customize it to suit your installation standards. Some of the customization is required, and some is optional. (Note that character strings need to be enclosed in double quotation marks as shown in the parameter descriptions.) The required changes are to REXX EXEC parameters that specify data set names, and also the mirror transaction identifier, described as follows:

CBKCMIR = "CSMI"|"*tranid***"**
Specifies the name of the CICS mirror transaction to be specified on EXCI requests issued by CICS BAC ISPF administration interface. When the EXCI request is executed in the target CICS region, it runs under this transaction ID as an MRO mirror transaction.

CSMI This is the normal CICS EXCI default mirror transaction ID.

transid Specifies the 4-character name that you want CICS to use as the name of the MRO mirror transaction when it executes the EXCI requests it receives from CICS BAC ISPF administration interface. To avoid confusion with other mirror transactions that might be running in your CICS regions, you might want to choose a unique ID to identify CICS BAC ISPF administration interface mirror transactions. Define the appropriate transaction ID in the CICS region for a CICS mirror transaction, ensuring the following:

- The program associated with the transaction ID must be DFHMIRS
- The profile for the transaction ID must be DFHCICSA.

You might consider using the CICS-supplied CSMI transaction definition as a basis for this transaction ID.

CBKLLIB="*&hlq***.SCBKLOAD"**
CBKEXEC="*&hlq***.SCBKEXEC"**
CBKMLIB="*&hlq***.SCBKMENU"**
CBKPLIB="*&hlq***.SCBKPENU"**
CBKTLIB="*&hlq***.SCBKTENU"**
Specify your own high-level qualifiers that you chose for these five ISPF administration interface data sets when you installed them. In CBKRCBAC, the data set names are defined on their respective parameters with the prefix *&hlq*. Substitute the actual high-level qualifier for the string *&hlq*. The five parameters, and their corresponding data sets, are as follows:

CBKLLIB
Specifies SCBKLOAD, the CICS BAC module load library

CBKEXEC
Specifies SCBKEXEC, the CICS BAC REXX EXEC library

CBKMLIB
Specifies SCBKMENU, the CICS BAC ISPF administration interface English messages data set

CBKPLIB
Specifies SCBKPENU, the CICS BAC ISPF administration interface panels data set

CBKTLIB
Specifies SCBKTENU, the CICS BAC ISPF administration interface English table data set containing the default PF key settings.

CBKSUMLQ="*middle_qualifier***"**
Specifies the middle qualifier of the data set that the ISPF administration interface uses to save information about each user of the interface. This is referred to as the TSO user's CICS BAC ISPF table. The REXX EXEC allocates the data set with a name in the form *TSOuserid.cbksumlq*.CBKTABL, where *TSOuserid* is the TSO user ID, and *cbksumlq* is the middle qualifier. The first time you start the ISPF administration interface it creates the CICS BAC TSO user data set automatically for you, and thereafter allocates the data set each time you start the interface.

middle_qualifier
Specifies the middle qualifier of the data set that the ISPF administration interface maintains for each TSO user of the interface. The high-level qualifier is the TSO user ID and the low-level name is CBKTABL. The ISPF administration interface use this TSO user's CICS BAC ISPF table to save your data when you select various options from the primary menu. See "Specifying the control file table data set" on page 39 and "Selecting a target CICS region" on page 40.

The remaining parameters in the EXEC that you can tailor for your own needs are all optional.

CBKCFTDS="*table_dataset_name***"**
Specifies a default control file table data set name. If you specify this parameter, the ISPF administration interface uses this table data set name whenever a user starts the ISPF administration interface for the first time; that is, when the user data set specified by the CBKSUMLQ does not exist. The default control file table data set name specified by CBKCFTDS is stored in your TSO user's CICS BAC ISPF table as an initial value, and is the same for all new user's of the CICS BAC ISPF interface.

When the CICS BAC REXX EXEC starts, it checks to see if the TSO user's CICS BAC ISPF table (as specified using the CBKSUMLQ parameter) already exists, and if it does, the TSO user is not a new user and the ISPF administration interface ignores the CBKCFTDS parameter.

If the TSO user's CICS BAC ISPF table does not exist, the ISPF interface checks to see if CBKCFTDS specifies a data set name (that is, not a null value). If CBKCFTDS specifies a valid data set name, it is entered in the TSO user's CICS BAC ISPF table as the name of the control file table data set name.

Using this parameter removes the need for new users of the interface to specify their control file table data set name using option 3 on the Primary Option Menu. However, although the data set is specified by the CBKCFTDS

parameter, it is not selected for the ISPF session and you still have to select the data set using option 2 on the Primary Option Menu.

Note: If you specify the CBKCFTDS in your CICS BAC REXX EXEC parameters, it is used for all new users of the CICS BAC ISPF administration interface and thus can be used only when the control file table data set name specified is common to all users of the interface.

CBKCFTCO
Specifies a default control file table data set comment about the data set specified on CBKCFTDS. This is used only when you also specify the CBKCFTDS parameter.

CBKSMAXD=12|*number*
Specifies the maximum number of 4KB pages the ISPF interface is to allocate as a buffer for downloading data from the target CICS region. You can specify a number of pages in the range 1 through 999; the default is 12.

Note: Note that a command fails if the allocated buffer is too small for the command.

CBKSBRWE=4096||*number*
Specifies the maximum number of browse entries the ISPF interface is to allow to be downloaded for any request. This affects the list of resources for a selected region, as well as prompt lists for resource types. You can specify a number in the range 1 to 4096; the default is 4096.

CBKSLPRM="N"|"Y"
Specifies that the ISPF administration interface is automatically to use the MV logical parmlib concatenation when looking for the control file table CBKCFTBL.

N On initial startup do not look for CBKCFTBL in the MVS parmlib concatenation

Y On initial startup, automatically look for CBKCFTBL in the MVS parmlib concatenation.

All the required and optional customization fields are fully commented in the CBKRCBAC REXX EXEC.

Starting the REXX EXEC
About this task

When you have completed the customization of your ISPF administration interface CBKRCUST REXX EXEC, you can start the interface by entering the following command on the ISPF Command Shell command line (the ISPF Command Shell is usually option 6 on the ISPF Primary Option Menu):

```
EXEC 'hlq.SCBKEXEC(CBKRCUST)'
```

If you've stored the EXEC in a different data set, replace *hlq*.SCBKEXEC with the your own data set name.

Tailoring existing ISPF menus
About this task

You can also add the CICS BAC ISPF administration interface to an existing ISPF menu. The following is an example of the steps needed to do this, where the administration interface is defined as option C.

1. First, add a line similar to the following to the ")AREA" section of the menu panel source:

   ```
   C CICS BAC      CICS BAC ISPF administration interface
   ```

2. Next, add the following line in the ")PROC" section of the menu panel source:

   ```
   C,'CMD(CBKRCUST) NOCHECK NEWAPPL(CBKA)'
   ```

Also ensure that the data set containing the REXX EXEC CBKRCUST is added to your TSO logon PROC SYSPROC concatenation. The REXX EXEC is the only CICS BAC data set you need to add to the logon PROC, because CICS BAC dynamically allocates all the other required data sets during the ISPF administration interface startup.

Specifying the control file table data set

About this task

If you have correctly customized the CBKRCUST REXX EXEC, when you start the CICS BAC ISPF administration interface it displays its initial menu, entitled CICS BAC Primary Option Menu, which is illustrated in Figure 10.

```
  File  Region  Preferences  Help
--------------------------------------------------------------------------------
                        CICS BAC Primary Option Menu
Option ===>

1  Select a CICS region

2  Select a control file table data set
3  Maintain the control file table data set list

4  Maintain user default properties

5  Preferences

X  Exit

Current control file data set . : NBELL.DOC.PARMS









 F1=Help    F2=Split   F3=Exit    F7=Up      F8=Down    F9=Swap   F12=Cancel
```

Figure 10. The CICS BAC ISPF administration interface Primary Option Menu

You cannot select a CICS region from this menu until you have first selected the data set that contains a valid CICS BAC control file table member (CBKCFTBL); see “The control file table” on page 23 for information about the control file table. You can specify the control file table data set in one of two ways:

1. If have chosen to define the CICS BAC control file table as a member of a data set in the MVS logical parmlib concatenation, you can select the logical parmlib concatenation as location of the control file data set by first choosing option 2 **Select a control file table data set** and pressing ENTER. On the next panel displayed, select option 1 **(MVS logical parmlib concatenation**) and press

ENTER to make the logical parmlib concatenation the current data set for the CICS BAC control file table for your TSO user ID.

2. If you have chosen to create the CICS BAC control file table as a member of a partitioned data set that is *not* part of the MVS logical parmlib concatenation, identify the data set to the ISPF administration interface for your TSO user ID. There are two ways you can do this:
   a. By specifying a default control file table data set name on the CBKCFTDS parameter in the CICS BAC REXX EXEC parameters (see page 37). In this case, you go straight to option 2 in the menu and select the data set for use. Note that the use of the CBKCFTDS parameter is possible only if all CICS BAC TSO users use the same control file table data set name.
   b. Through the Primary Option Menu by selecting option 3 **Maintain the control file table data set list** on the CICS BAC ISPF administration interface Primary Option Menu, and press ENTER, then follow the instructions to identify the data set. Adding the data set causes it to be used automatically as the current CICS BAC control file table data set for your user ID.

   Note: The options a TSO user selects are unique to each user and the ISPF administration interface saves the information for each TSO user. For this reason, if CBKCFTDS is not used, each TSO user of the ISPF administration interface needs to perform step 2 at least once so that the relevant information can be saved for later use of the interface. The data is saved in a data set that has a name determined by the CBKSUMLQ parameter (see "Customizing the REXX EXEC" on page 35).

Selecting a target CICS region

About this task

When you have identified the control file table data set that you are going to use, you can use option 1 **Select a CICS region** on the ISPF administration interface Primary Option Menu to select the CICS region for which you want to perform administration functions on its control file. The ISPF panel displayed when you select option 1 is shown in Figure 11. To select a region form the list, type S against the region you want and press ENTER.

```
  File  Preferences  Help
-------------------------------------------------------------------------------
                         CICS Region Selection List          Row 1 to 7 of 7
Command ===>                                                Scroll ===> HALF

Select one or more CICS regions below and press ENTER to continue.

S Select
Sel Region name  Control file data set name
    CICS220T     MVS10.CICSBAC.CBKCTL10
    BEPCICS1     MVS10.CICSBAC.CBKCTL10
    CICSHT01     MVS10.CICSBAC.CBKCTL10
    CICSHA01     MVS11.CICSBAC.CBKCTL11
    CICSHA02     MVS11.CICSBAC.CBKCTL11
    CICSHA03     MVS12.CICSBAC.CBKCTL12
    CICSHA04     MVS13.CICSBAC.CBKCTL13
****************************** Bottom of data ******************************
```

Figure 11. Selecting a CICS region

When you have identified and selected the target CICS region, the ISPF administration displays the Region Resources panel shown in Figure 12. This panel enables you to list, update, and add the CICS BAC resource object records listed on the panel. When you have completed your work for a selected CICS region, you can go back to the CICS Region Selection List panel and choose a different CICS region.

```
  File  Preferences  Help
-------------------------------------------------------------------------------
                              Region Resources
Option ===>

Current CICS region . : CICS220T

Selection filter  . . .           (For options 1-6, "*" or blank selects all)

Note: Enter $Default (exactly as shown) as the selection filter to view the
default properties record for a resource type.

1  VSAM files
2  Transient data queues
3  Transaction IDs
4  Programs
5  Application groups
6  Application lists

7  Region properties

X  Exit




 F1=Help    F2=Split   F3=Exit    F7=Up      F8=Down    F9=Swap   F12=Cancel
```

Figure 12. ISPF Region Resources panel showing CICS BAC resource object types

Help and tutorial facilities

The CICS BAC ISPF administration interface uses standard ISPF tutorial facilities to provide extensive online help. Every processing panel has an associated help panel that describes the function of the panel and any options or selections the panel contains. To view the online help for a panel, simply press the HELP function key (typically PF1) while using the panel for which you need help.

ISPF administration interface messages

Information, warning, and error messages that are displayed by the ISPF administration interface usually contain both short message text and long message text. In these cases, the short message is displayed first in the upper left hand corner of the panel. If you need additional explanation of the message, press the HELP function key to display the long message text for the message. The long message text is displayed automatically if a message does not contain short message text. You can display additional information pertaining to a message by pressing HELP when the long message text is displayed on the panel. This causes a pop-up panel to be displayed that provides a detailed explanation of the message, and where appropriate, the system action that will be taken and the recommended user action.

Chapter 4. CICS BAC batch request utility commands

This chapter describes the commands that CICS BAC provides to enable you to specify the requests you want to send to the CICS BAC request server in a CICS region.

Batch request utility processing

The batch request utility first opens the input data set that contains the commands and then performs validation on all the commands, checking for syntax errors. When the utility has determined that there are no syntax errors, it starts to process each command sequentially, preparing to send the commands to the appropriate CICS region (if CICS BAC is active in the region). As the batch request utility processes each command, it checks for a change in CICS applid. If there is no change, and to optimize performance, each command is moved into a buffer ready for sending to the target CICS region. The CICS BAC batch request utility sends the commands stored in the buffer when it detects:

- A new CICS applid specified on a command
- A new DEFAULT CICS*(applid)* that changes the CICS applid
- The end of the commands input stream.

After it has sent a batch of buffered commands to a CICS region, the CICS BAC batch request utility waits until control is returned from the EXCI DPL request, then resumes with the next command in sequence in the input data set. If the previous buffer sent was full, the next command could be for the same target region as the request sent previously. This process, and what happens when the request server receives a set of buffered commands from the utility is described in the next topic.

CICS BAC request server processing

How a request server processes each DPL request it receives depends on whether the commarea is complete or whether there are more commands in a follow-on commarea.

A single complete commarea

If all the commands for a specific CICS region fit in a single 32 KB commarea, all the data is sent to CICS in one EXCI request. This is received by CICS, which starts an EXCI mirror transaction as the CICS BAC request server. This request server transaction proceeds to process each of the commands in the commarea.

If one of the commands included in the commarea is a LINK command, the request server transaction links to the specified application program. If one of the commands included in the commarea is a RUNCEMT command, the request server transaction links to DFHEMTA.

When all commands in the request are processed, the request server terminates and returns the results to the batch request utility.

Multiple commareas

If the batch request utility cannot fit all the commands for a CICS region in the same 32 KB commarea, the batch request utility sends the commarea as a first part of the total request. The processing in the CICS region is then as follows:

- A request server is initiated in the receiving CICS region to process the request. The request server detects that it is only a partial command block, therefore it writes the commarea to a CICS TS queue and returns to the batch request utility.
- When the CICS region receives a follow-on commarea from the batch request utility, it initiates a request server transaction, which recognizes the input as another partial command block. The request server writes this second command block as the second item in the TS queue, then returns to the batch request utility again.
- Each initiated CICS BAC request server stores partial requests in the same TS queue until the server receives the final command block. The request server reads the previous items from the TS queue and reassembles all the original commareas into a single piece of contiguous storage.
- When all the partial requests are reassembled, the request server transaction links to another CICS BAC program to service the requests. If one of the requests is a LINK or RUNCEMT command, CICS BAC links to either the specified application program or DFHEMTA to service the command. When all commands are processed (or if an error is detected), the CICS BAC processing program returns control back to the CICS BAC request server program.
- If the CICS request server finds that the return commarea is larger than 32 KB, it breaks it up into an appropriate number of blocks and writes the second and subsequent blocks to a TS queue. The CICS request server puts the first block in the return commarea, then returns to the batch request utility
- The batch request utility recognizes that it has only received part of the results and saves it in main storage, and issues another EXCI request to the CICS request server to return the rest of the data. This process continues until the request server has exhausted the TS queue, with the batch request utility appending the partial results in its main storage area until complete. When the final block of returned data has been received, the batch request utility continues processing as if the returned data had fitted into a single commarea.

Accessing the CICS region control file

Only one application can allocate and access the CICS BAC control file at a time, and access is serialized by means of an exclusive systems-level enqueue. In this context, the application could be one of the following:

- A CICS region in which CICS BAC is active (until CICS BAC is shut down in the region)
- A CICS BAC batch request utility job step
- The CICS BAC communication server when it is processing requests from the workstation client

When the CICS BAC startup procedure runs in the CICS region, it starts the CICS BAC subtask, which obtains the enqueue and holds it until the CICS region is shut down, either normally or abnormally. If the CICS BAC batch request utility cannot obtain a systems-level enqueue using the same resource name (the region applid) used by the CICS BAC request server, it knows that the subtask is active and that its batch request utility input commands must be processed by EXCI requests passed to the CICS region. If it succeeds in obtaining the enqueue, it means that the CICS BAC request server subtask is not active and the batch request utility can allocate and open the control file itself and process itself those input commands that can be applied to the control file. This could be any of the SET commands but not the DEFAULT command, which is applicable, and local, only to the CICS BAC batch request utility.

By means of this enqueue technique, the requested status of a CICS resource can be set directly in the control file without the CICS region being active. Also, by ensuring that the CICS BAC startup procedure runs *before* any CICS resource can be opened, you make sure that the status specified in the control file is honored by the CICS region, because the CICS BAC startup processor checks all the resources in the control file and issues the necessary CICS SPI commands to enforce the required status in the CICS region.

Command format

The basic format of a CICS BAC batch request utility input command is as follows:

```
COMMAND NAME Parameter1(value1),Parameter2(value2),...,Parametern(valuen)
```

The rules for coding commands are as follows:

- The command line is restricted to column positions 1 to 71 inclusive
- You indicate that a parameter string continues on the next line by specifying any non-blank character in column 72.
- The command name must be the first non-blank word on the line, and it must be followed by one or more spaces.
- You follow the command name with one or more parameter(*value*) strings (or none).
- You can abbreviate parameters and enumerated parameter values (such as "OPEN" and "CLOSED") to their shortest unique string. However, CICS BAC does not guarantee that an abbreviation that is valid in this release will be valid in a future release. Therefore, you should consider using abbreviations that are not the shortest abbreviation possible for a parameter.
- If you specify more than one parameter string they must be separated by a comma.
- Some parameters are optional, and others are required.
- Some parameters have a default value if you omit them, and these defaults are underscored in the syntax.
- If you omit an optional parameter for which there is no default value, that parameter is not used in the command.
- The default value for some parameters depends on the value of others.
- You pass your commands to the CICS BAC batch request utility in a data set with the DDNAME CBKIN.
- The order in which command parameters are shown in the syntax diagrams in this chapter does not imply a required sequence. You can order the parameters within a command in any sequence.

The above rules are explained in further detail, as appropriate, in the description of each command. The commands provided by CICS BAC are:

- "DEFAULT" on page 46 (see page "DEFAULT" on page 46)
- "LINK" on page 48 (see page "LINK" on page 48)
- "RUNCEMT" on page 50 (see page "RUNCEMT" on page 50)
- "SET FILE" on page 51(see page "SET FILE" on page 51)
- "SET GROUP & SET LIST" on page 57(see page "SET GROUP & SET LIST" on page 57)
- "SET PROGRAM" on page 58(see page "SET PROGRAM" on page 58)
- "SET TDQUEUE" on page 59(see page "SET TDQUEUE" on page 59)
- "SET TRANSID" on page 60 (see page "SET TRANSID" on page 60)

• “START” on page 61(see page “START” on page 61)

Syntax notation for batch request utility commands

The syntax for each batch request utility input command is presented in the form of a diagram, commonly referred to as a **railroad diagram**. The syntax, which is interpreted by following the dashed lines and arrows from left to right, and top to bottom, is described in the following table.

Symbol	Meaning
`>>-----A-----><`	A required parameter that you must provide.

`>>-----A-----><`
`+-B-+`
`+-C-+`
	A set of alternative parameters, one of which you must provide

`+-A-+`
`>>-----------><`
`+-B-+`
`+-C-+`
	A set of optional parameters, where A is the default. You may provide only one of these options.

`>>-----------><`
`+-A-+`
`+-B-+`
	A set of optional parameters. None are required, and there is no default. You may provide only one of these.

`>>-- Name +-><`
 	
`Name:`
`>>--A----B---->`
	Denotes a named section provided elsewhere in the command syntax.

`Punctuation and upper case characters`	Code exactly as documented.
`Lowercase, italicized characters`	Requires you to code your own text to replace these fields.

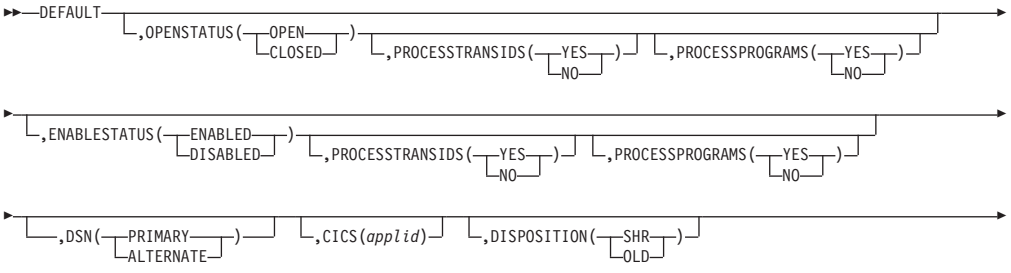
DEFAULT

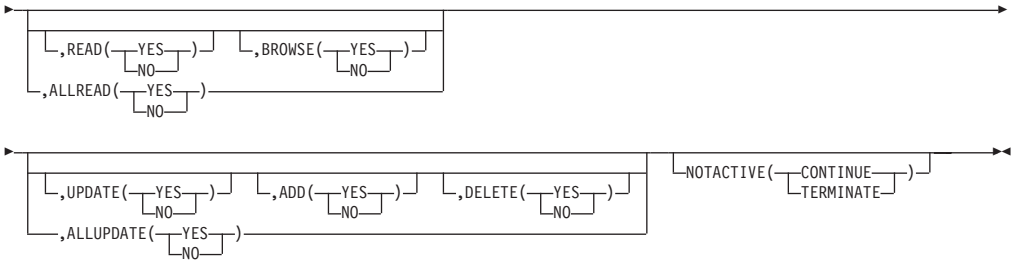
Function

To set the default values for parameters to be applied to subsequent input commands.

Syntax

DEFAULT





Description

You can use the DEFAULT command to specify default values to be applied to subsequent input commands. If you specify a parameter on a DEFAULT command that is not applicable to a subsequent input command, the parameter value is ignored for that command. If you later specify a DEFAULT parameter explicitly on a different command, or on another DEFAULT command, the equivalent parameter value specified on the earlier DEFAULT command is ignored.

You can specify multiple DEFAULT commands in the same input file. Each DEFAULT command completely overrides the values specified on any previous DEFAULT command. Specifying the DEFAULT command without any parameters causes subsequent input commands to be processed without any default parameter values. Similarly, omitting just one parameter from a DEFAULT command causes that parameter default only to be removed.

Parameters

CICS(*applid*)
Specifies the default applid to which the CICS BAC batch request utility is to direct commands if the command does not itself specify an explicit applid.

DISPOSITION({SHR | OLD})
Sets the default value for this parameter for the SET FILE command. For details, see the DISPOSITION parameter on the “SET FILE” on page 51 command.

DSN({PRIMARY | ALTERNATE})
Sets the default value for this parameter for the SET FILE, SET GROUP, and SET LIST commands. For details, see the DSN parameter on the “SET FILE” on page 51 command.

ENABLESTATUS({ENABLED | DISABLED})
Sets the default value for this parameter for the SET FILE, SET GROUP, SET LIST, SET PROGRAM, and SET TDQUEUE commands. For details, see the ENABLESTATUS parameter on the “SET FILE” on page 51 command, or any of the others.

OPENSTATUS({OPEN | CLOSED})
Sets the default value for this parameter for the SET FILE, SET GROUP, SET LIST, and SET TDQUEUE input commands. For details, see the OPENSTATUS parameter on the “SET FILE” on page 51 command, or any of the others.

PROCESSPROGRAMS({YES | NO})
Sets the default value for this parameter for the SET FILE and SET TDQUEUE commands. For details, see the PROCESSPROGRAMS parameter on the “SET FILE” on page 51 command, or the “SET TDQUEUE” on page 59 command.

PROCESSTRANSIDS({YES | NO})
Sets the default value for this parameter for the SET FILE and SET TDQUEUE input commands. For details, see the PROCESSTRANSIDS parameter on the "SET FILE" on page 51 command, or the "SET TDQUEUE" on page 59 command.

NOTACTIVE({CONTINUE | TERMINATE})
Specifies the default action you want the CICS BAC batch request utility to take for the LINK, RUNCEMT, and START commands if the target CICS region is not available, or if the CICS BAC startup procedure has not been run in the region. For details, see the "LINK" command.

ALLREAD({YES | NO})
Sets the default value for this parameter for the SET FILE command.

You cannot specify ALLREAD with either READ or BROWSE.

ALLUPDATE({YES | NO})
Sets the default value for this parameter for the SET FILE command.

You cannot specify ALLUPDATE with any of the ADD, UPDATE, or DELETE options.

ADD({YES | NO})
Sets the default value for this parameter for the SET FILE command.

BROWSE({YES | NO})
Sets the default value for this parameter for the SET FILE command.

DELETE({YES | NO})
Sets the default value for this parameter for the SET FILE command.

READ({YES | NO})
Sets the default value for this parameter for the SET FILE command.

UPDATE({YES | NO})
Sets the default value for this parameter for the SET FILE command.

The following options all relate to the file operations as defined by the CICS file resource definition and are applicable to the CICS BAC SET FILE command only. For details of all these options, see the PROCESSTRANSIDS parameter on the "SET FILE" on page 51 command.

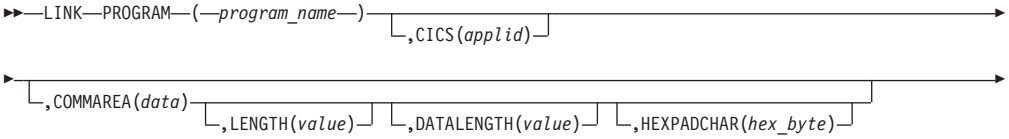
LINK

Function

To execute a program in the CICS region.

Syntax

LINK



Description

You can use the LINK command to link to and execute a program in the target CICS region.

Note that the batch request utility does not link directly to the requested program. As with all other commands, the LINK command is passed to a CICS BAC request server, which in turn links to the requested program. Therefore, the linked-to program runs as part of the same EXCI mirror transaction that is running the CICS BAC request server.

The PROGRAM parameter is the only required parameter: all the others are optional.

The data returned from the DPL program is written to the CBKPRINT data set, up to a maximum of 256 bytes.

Parameters

CICS(*applid***)**
Specifies the applid of the CICS region to which the CICS BAC batch request utility is to send the distributed program link command. If you omit this option, the command is routed to the default CICS region specified by the most recent DEFAULT command.

COMMAREA(*commarea_data***)**
Specifies the data to be passed as the communications area to the DPL program in CICS. The data can be up to a maximum of 256 bytes, with no restrictions on the contents, except you cannot have an unmatched parenthesis within the outer parentheses that enclose the commarea data. For example, COMMAREA(abc(def) is not valid, but COMMAREA(abc(123)def) is valid.

If you specify a commarea, you can also specify LENGTH, DATALENGTH, and HEXPADCHAR (see below).

DATALENGTH(*value***)**
Specifies the length of the data to be passed in the commarea. If you omit this option, the length defaults to the length of the actual data specified on the COMMAREA parameter. Note that:

- The maximum value you can specify is 256 bytes.
- If you specify a value that is greater than the length of the COMMAREA data field, the commarea data is padded by the character specified on the HEXPADCHAR option, or by nulls, up the length specified.
- If you specify a value that is less than the actual data length on the COMMAREA parameter, the CICS BAC batch request utility returns an error.

You can specify DATALENGTH only if you also specify a COMMAREA with data.

HEXPADCHAR(*hex_byte***)**
Specifies the hexadecimal character that you want to use to pad the data up to the specified data length, or the maximum number of characters if data length

is not specified. For example, if you want to pad with spaces, specify HEXPADCHAR(40). If you omit this option the pad character defaults to a null value.

You can specify HEXPADCHAR only if you also specify a COMMAREA with data.

LENGTH(*hex_byte***)**
Specifies the length of the data to be returned from the CICS application program to the CICS BAC request server. If you omit this option, the length defaults to the length of the actual data specified on the COMMAREA parameter. The maximum value you can specify for return data is 32500.

You can specify LENGTH only if you also specify a COMMAREA with data.

If you specify a length less than 256 bytes, it also controls the number of bytes written to the CBKPRINT print data set. If you specify a length greater than 256, only the first 256 bytes are written to the data set

NOTACTIVE({CONTINUE | TERMINATE})
Specifies the action you want the CICS BAC batch request utility to take if the target CICS region is not available, or if CICS BAC is not started in the region, as follows:

CONTINUE
Specify CONTINUE if you want the CICS BAC batch request utility to ignore the command and continue processing with the next command in the input data set.

TERMINATE
Specify TERMINATE if you want the CICS BAC batch request utility to terminate because of the not active condition.

PROGRAM(progname)
Specifies the 1-8 character name of the CICS program to be linked to.

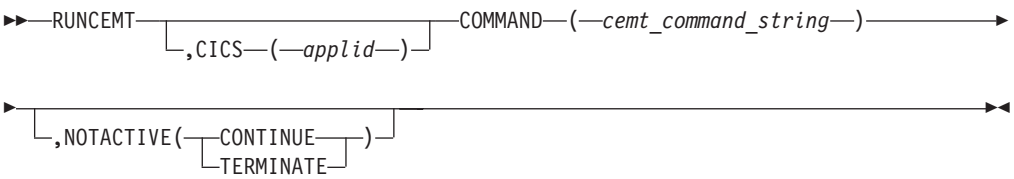
RUNCEMT

Function

To issue a CEMT command in the target CICS region.

Syntax

RUNCEMT



Description

You can use the RUNCEMT command to issue a CEMT command in a CICS region.

The CICS BAC batch request utility writes up to 256 bytes of the data returned by the CICS region from the CEMT command to the print data set, CBKPRINT.

Parameters

CICS(*applid***)**
Specifies the applid of the CICS region to which the CICS BAC batch request utility is to send the CEMT command. If you omit this option, the batch request utility routes the command to the default CICS region specified by the most recent DEFAULT command.

COMMAND(*cemt_command_string***)**
You can specify up to 256 characters to make up the CEMT command string. Up to 256 bytes of data returned from the CEMT command are written to the CBKPRINT data set. Enter the command string exactly as you would enter it from a 3270 terminal. For example, RUNCEMT COMMAND(CEMT I FILE(FILEA)).

NOTACTIVE({CONTINUE | TERMINATE})
Specifies the action you want the CICS BAC batch request utility to take if the target CICS region is not available, or if the CICS BAC startup procedure has not been run in the region, as follows:

CONTINUE
Specify CONTINUE if you want the CICS BAC batch request utility to ignore the command and carry on processing the next command in the input data set.

TERMINATE
Specify TERMINATE if you want the CICS BAC batch request utility to terminate because of the failed command.

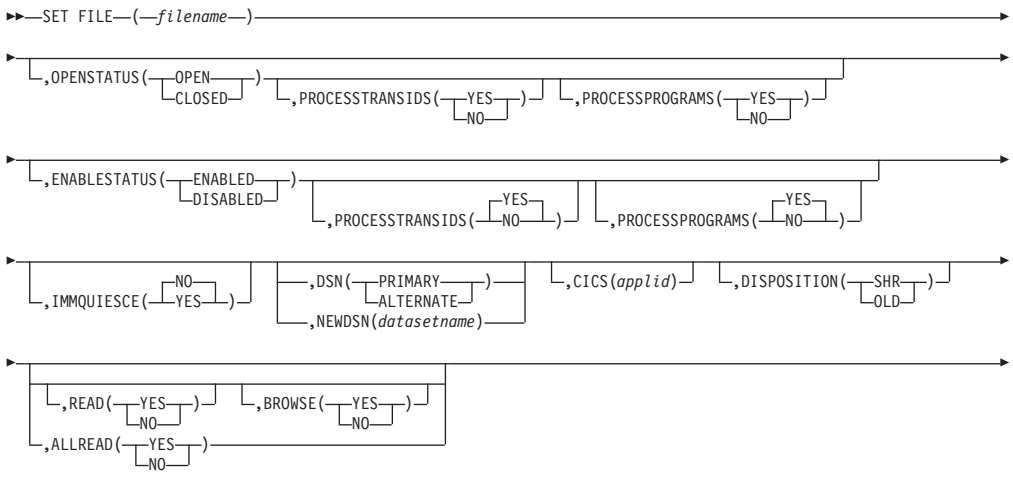
SET FILE

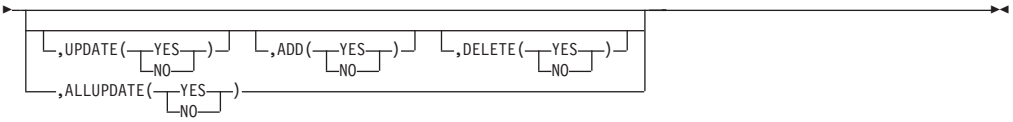
Function

To set the file options and process a file.

Syntax

SET FILE





Note: Although all the options in the above syntax diagram are shown as optional, you must specify at least either the OPENSTATUS parameter or the ENABLESTATUS parameter. For example, the minimum command could be:

```
SET FILE(FILEA),OPENSTATUS(CLOSED)
or
SET FILE(FILEA),ENABLESTATUS(ENABLED)
```

Description

The SET FILE command enables you to open, close, enable, or disable a VSAM file in a CICS region. You can also specify whether you want any transactions and programs that are in the control file record for the specified file to be processed, and you can vary the data set name that CICS is to use.

In this discussion of the SET FILE command, we assume that the files described in the various scenarios are dynamically allocated by CICS. It does not apply to those files that are defined by a DD statement in the CICS JCL, which are therefore allocated by MVS at job step initiation and remain allocated to CICS even when closed. We also stress that in addition to the points mentioned, that whether or not a file can be accessed through a batch job also depends on other factors not covered in this brief discussion. For example, the file disposition (OLD or SHR), the OPEN mode (INPUT or OUTPUT), and type of access (read or update) also affect the success or failure of the file control operation. When using CICS BAC commands, you should ensure that none of these other factors prevent a successful operation of the command you are using.

The discussion does not take into account the various file transition states, such as in the process of closing or being disabled (indicated by the CLOSING and DISABLING attributes). These have no effect on the issues being discussed.

In CICS, the open status of a VSAM file can either be OPEN or CLOSED, and this is indicated by the OPENSTATUS attribute in the file entry in the CICS file control table. If CICS closes a file, it is also de-allocated so that it can be processed by batch jobs. A file also has an enabled status, indicated by the ENABLESTATUS attribute, which can be either ENABLED, DISABLED, or UNENABLED. The enabled status determines whether or not a CICS application program request to a VSAM file can be attempted. If a file is disabled or unenabled, CICS fails the application program request without even attempting to send the request to VSAM. If the file is enabled, CICS issues the VSAM request on behalf of the application program.

The following are the most common combinations of these two states:

- OPEN, ENABLED
- CLOSED, ENABLED
- OPEN, DISABLED
- CLOSED, DISABLED
- CLOSED, UNENABLED

In the first case (OPEN, ENABLED), the file is open and available to CICS application programs and if a batch job tries to access the file, it generally fails.

In the second case (CLOSED, ENABLED), the file is closed and not allocated, but it is available to CICS programs. In this state, if your application program issues a file access request, CICS attempts to allocate then open the file. If CICS cannot allocate and open the file, it is most probably because it is allocated to a batch job, and the application program request fails. Also in this case, as long as the file is closed, it is available for access by batch jobs.

In the last three cases in the above list (OPEN, DISABLED; CLOSED, DISABLED; and CLOSED, UNENABLED), CICS fails all file control requests without even attempting them because the file is not enabled. The key difference in these three cases is that in one the file is open and in the other two it is closed. In the open case, it is allocated to CICS so that batch jobs that attempt to access the file will probably fail. In the closed cases , it is not allocated to CICS, and is available for access by batch jobs.

Note that when a file is closed to CICS, and therefore de-allocated, the file can even be deleted by a batch job.

For the most part, the DISABLED and UNENABLED states are the same. Both states prevent an application program from accessing the file. However, there is one important difference that you must take into consideration when changing the state of a file.

The only way that a file becomes unenabled is when you close the file without altering its enabled status. You can do this using the CICS BAC batch request utility, or by using the EXEC CICS SET FILE command or the CEMT SET FILE transaction. For example, if FILEA is currently open and enabled, and you issue the CEMT command SET FILE(FILEA) CLOSED, the resulting file attributes are set to CLOSED and UNENABLED. However, if the file is currently open and disabled, the same command results in the file state being set to CLOSED and DISABLED.

The difference between a disabled and an unenabled file is in how the file state is set when a closed file is subsequently opened. If a file is currently closed and disabled, and a request is made through the CICS BAC batch request utility, CICS system programming interface (SPI), or CICS CEMT command to open the file without altering its enable status attribute, the file is opened (assuming there is nothing to keep it from being opened), but its enable status remains DISABLED. On the other hand, if a file is unenabled, an open request that does not also specify an enabled status results in the file state being set to OPEN and ENABLED.

If you issue a request that alters the enable status of a file explicitly, the file ENABLESTATUS attribute is changed regardless of whether the current state is ENABLED, DISABLED, or UNENABLED.

When you define CICS BAC batch request utility SET FILE commands, it is very important to consider the final effects of various state changes. Not doing so can result in an unwanted state for a file. For example, if the state of FILEA is CLOSED, DISABLED, simply issuing a command to open the file will result in FILEA being OPEN, DISABLED – a state in which a CICS application program cannot access the file.

You should also be aware that in various situations CICS itself attempts to change the state of a file on behalf of an application program, and as specified by CICS file

resource definitions. With this in mind, consider what could happen if a file is set by CICS BAC startup processing to the CLOSED, DISABLED state:

- If you define the initial state of the file as enabled, but to be opened at first reference instead of during CICS startup, CICS installs the file resource definition with the file set to the FIRSTREF and ENABLED state. This is then changed by your CICS BAC startup processing to the CLOSED, DISABLED state. When the first reference to the file is made by an application program, CICS alters the OPENSTATUS attribute to OPEN, but leaves the file enabled status unchanged. The result is that the application program file request fails because of the DISABLED status.
- If you define a file to CICS to be opened at CICS startup, CICS attaches the CSFU transaction to open the file *after* CICS BAC startup processing, by which time CICS BAC has set the file to the CLOSED and DISABLED state. The CICS CSFU transaction sets the file status to OPEN without altering its enable status, resulting in the same OPEN and DISABLED state described above.

CICS BAC batch request utility SET FILE (and other SET commands) are implemented by CICS SPI commands, therefore the results of such commands are the same as if you issued an EXEC CICS SET FILE or CEMT SET FILE command. For example, if you use the batch request utility to close a currently OPEN, ENABLED file (without altering the ENABLESTATUS) it results in a file state of CLOSED, UNENABLED. Thus, you can use the CEMT transaction or the CICS SPI through the CECI transaction to determine in advance what the effects of various batch request utility commands will be.

RLS-mode files: CICS BAC includes support for VSAM files that are defined to CICS with VSAM record-level sharing (RLS) enabled (that is, RLSACCESS(YES) is specified). There is special processing associated with RLS-mode files for which you issue SET FILE commands. For a information about how CICS BAC processes SET FILE commands for RLS files, see “CICS BAC support for record-level sharing (RLS)” on page 119.

Parameters

CICS(*applid*)
Specifies the applid of the CICS region to which this command is to be routed. If you omit this option, the batch request utility routes the command to the default CICS region specified by the most recent DEFAULT command.

DISPOSITION({SHR | OLD})
Sets the value of the file allocation disposition with CICS, which specifies how CICS allocates the file. The values you can specify are:

SHR Specify this value if you want the data set to be allocated by CICS as DISP(SHR) the next time CICS opens the file.

OLD Specify this value if you want CICS to allocate the file as DISP(OLD) the next time CICS opens the file.

DSN({PRIMARY | ALTERNATE})
Enables you to change the data set name associated with the CICS file definition. The values you can specify are:

PRIMARY
Specify this value if you want to change the data set name associated with the CICS file definition to the primary data set name defined in the CICS BAC control file object record.

ALTERNATE
Specify this value if you want to change the data set name associated with the CICS file definition to the alternate data set name in the CICS BAC control file object record.

Note:

1. The CICS BAC request server can process the DSN option only if the specified file is dynamically allocated by CICS. CICS BAC cannot process the DSN option for a file that is explicitly defined in the CICS region startup JCL and therefore allocated to CICS by MVS. If you attempt to alter the data set name for a file that is allocated by MVS, the SET FILE command fails and the batch request utility job step terminates immediately with return code 12.
2. Also, the CICS BAC request server can process the DSN option only if the specified file is currently closed and disabled, or if the DSN parameter is specified as part of a command to CLOSE and DISABLE the file. If the file is *not* already closed, *or* CLOSE and DISABLE are not specified on the command, the command fails and the batch request utility job step is terminated.
3. If the CICS region is unavailable or CICS BAC is not active in the CICS region, the batch request utility itself sets the data set name in the CICS region control file. In this case, it does so without regard to the last known state of the file, or whether or not the file was explicitly defined in the CICS region JCL.
4. The DSN parameter is mutually exclusive with the NEWDSN parameter—you cannot specify both options. If you omit this and the NEWDSN option, the data set name remains as defined in the CICS region.

ENABLESTATUS({ENABLED | DISABLED})
Specifies whether the file is to be enabled or disabled to CICS.

FILE(*filename*)
Specifies the CICS file control 8–character file name.

IMMQUIESCE({NO | YES})
Specifies the type of quiesce command CICS BAC issues when it determines that it must issue a quiesce for a file that references an RLS data set. This is the equivalent of the IMMQUIESCED parameter on the EXEC CICS SET DSNAME(*name*) SPI command. See "CICS BAC support for record-level sharing (RLS)" on page 119 for more information.

NEWDSN(*data-value*)
Specifies a new, fully qualified, 44–character data set name to be associated with the CICS file definition.

Note:

1. The CICS BAC request server can process the NEWDSN option only if the specified file is currently closed and disabled, or if the NEWDSN parameter is specified as part of a command to CLOSE and DISABLE the file. If the file is *not* already closed, *or* CLOSE and DISABLE are not specified on the command, the command fails and the batch request utility job step is terminated.
2. If the CICS region is unavailable or CICS BAC is not active in the CICS region, the batch request utility itself sets the data set name in the CICS region control file. In this case, it does so without regard to the last known state of the file.

3. The NEWDSN parameter is mutually exclusive with the DSN parameter—you cannot specify both options. If you omit this and the DSN option, the data set name remains as defined in the CICS region.

OPENSTATUS({OPEN | CLOSED})
Specifies whether the file is to be opened or closed in the CICS region.

PROCESSPROGRAMS({YES | NO})
Specifies whether any program specified in the CICS BAC control file object record for the file being processed is also to be processed, depending on the requested status of the file.

YES Specifying YES means that the program status is set as follows:

- If the command specifies OPENSTATUS(OPEN) *and* ENABLESTATUS(ENABLED), the associated programs are ENABLED. The programs are enabled *after* the file status is set.
- If the command specifies OPENSTATUS(CLOSED) *or* ENABLESTATUS(DISABLED), the associated programs are DISABLED. The programs are disabled *before* the file status is set

NO Specifies that the status of associated programs is not to be changed.

PROCESSTRANSIDS({YES | NO})
Specifies whether any transaction specified in the CICS BAC control file object records for the file being processed is also to be processed, depending on the requested status of the file, as follows:

YES Specifying YES means that the transaction status is set as follows:

- If the command specifies OPENSTATUS(OPEN) *and* ENABLESTATUS(ENABLED), the associated transactions are ENABLED.
- If the command specifies OPENSTATUS(CLOSED) *or* ENABLESTATUS(DISABLED), the associated transactions are DISABLED.

NO Specifies that the status of associated transaction IDs is not to be changed.

ALLREAD({YES | NO})
Sets the value of both the file READ and BROWSE attributes (see below).

You cannot specify ALLREAD with either READ or BROWSE.

ALLUPDATE({YES | NO})
Sets the value of the file ADD, DELETE, and UPDATE attributes (see below).

You cannot specify ALLUPDATE with any of the ADD, UPDATE, or DELETE options.

ADD({YES | NO})
Sets the value of the file ADD attribute, which specifies whether records can be added to the file.

BROWSE({YES | NO})
Sets the value of the file BROWSE attribute, which specifies whether records can be retrieved sequentially from the file.

DELETE({YES | NO})
Sets the value of the file DELETE attribute, which specifies whether records can be deleted from the file.

READ({YES | NO})
Sets the value of the file READ attribute, which specifies whether records on this file can be read.

UPDATE({YES | NO})
Sets the value of the file UPDATE attribute, which specifies whether records on this file can be updated.

The following options all relate to the file operations as defined by the CICS file resource definition.

SET GROUP & SET LIST

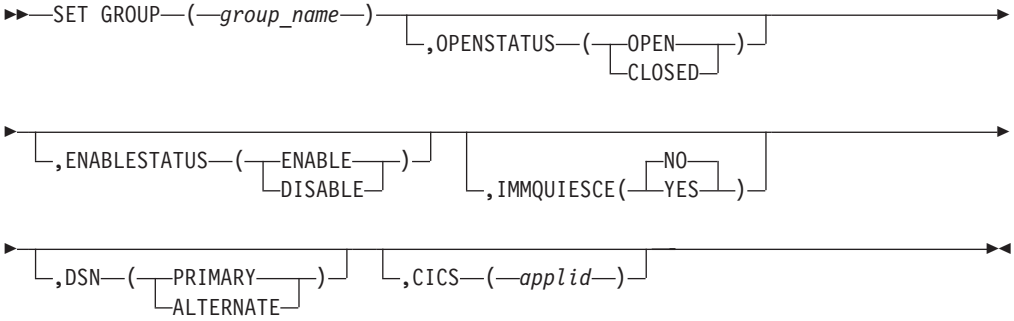
Function

The SET GROUP command enables you to set the attributes of all the objects, files, TD queues, transactions, and programs, in the specified group.

The SET LIST command operates on a list of groups, all of which are processed as in the SET GROUP command.

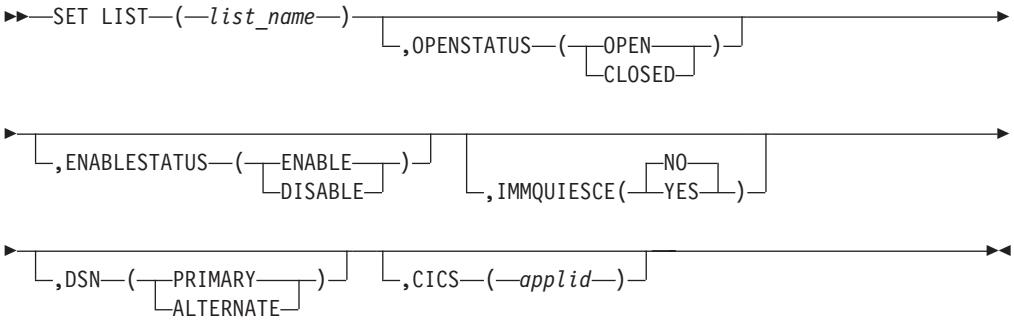
Syntax

SET GROUP



Syntax

SET LIST



Note: Although all the options in the SET GROUP and SET LIST syntax diagrams are shown as optional, you must specify at least either the OPENSTATUS parameter or the ENABLESTATUS parameter. For example, as a minimum, the command could be:

```
SET GROUP(GROUPX),OPENSTATUS(CLOSED)
or
SET LIST(LISTY),ENABLESTATUS(ENABLED)
```

Description

The SET GROUP and SET LIST commands apply the specified attributes, as appropriate, to each object in the group, or each object in all the groups in the list, as follows:

- DSN applies only to files
- OPENSTATUS applies to files and TD queues
- ENABLESTATUS applies to files, TDQueues, programs, and transactions

Parameters

CICS(*applid*)
Specifies the applid of the CICS region to which this command is to be routed. If you omit this option, the batch request utility routes the command to the default CICS region specified by the most recent DEFAULT command.

DSN({PRIMARY | ALTERNATE})
Specifies which data set the OPENSTATUS or ENABLESTATUS parameter is to be applied, using the information defined in the CICS BAC control file. The values you can specify are:

PRIMARY
Specify this value if you want to change the data set name associated with the CICS file definition to the primary data set name defined in the CICS BAC control file object record.

ALTERNATE
Specify this value if you want to change the data set name associated with the CICS file definition to the alternate data set name defined in the CICS BAC control file object record.

ENABLESTATUS({ENABLED | DISABLED})
Specifies whether the objects in the group or list are to be enabled or disabled to CICS.

OPENSTATUS({OPEN | CLOSED})
Specifies whether all the files and TD queues in the group or list are to be opened or closed in the CICS region.

IMMQUIESCE({NO | YES})
Specifies the type of quiesce command CICS BAC issues when it determines that it must issue a quiesce for a file that references an RLS data set. This is the equivalent of the IMMQUIESCED parameter on the EXEC CICS SET DSNAME(*name*) SPI command. See “CICS BAC support for record-level sharing (RLS)” on page 119 for more information.

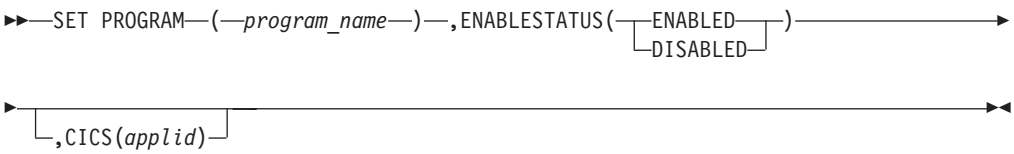
SET PROGRAM

Function

To set the executable status of a program.

Syntax

SET PROGRAM



Parameters

CICS(*applid*)
Specifies the applid of the CICS region to which this command is to be routed. If you omit this option, the command is routed to the default CICS region specified by the most recent DEFAULT command.

ENABLESTATUS({ENABLED | DISABLED})
Specifies whether the program is to be enabled or disabled in the target CICS region.

PROGRAM(progname)
Specifies the 1-8 character CICS program name that you want to enable or disable.

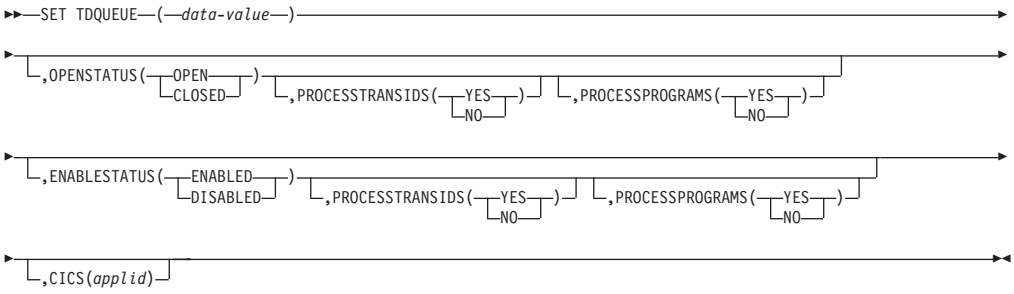
SET TDQUEUE

Function

To process a transient data (TD) queue.

Syntax

SET TDQUEUE



Note: Although all the options in the syntax diagram are shown as optional, you must specify at least either the OPENSTATUS parameter or the ENABLESTATUS parameter. For example, as a minimum, the command could be:

```
SET TDQUEUE(ANYQ),OPENSTATUS(CLOSED)
```

Parameters

CICS(*applid*)
Specifies the applid of the CICS region to which this command is to be routed. If you omit this option, the command is routed to the default CICS region specified by the most recent DEFAULT command.

ENABLESTATUS({ENABLED | DISABLED})
Specifies whether the TD queue is to be enabled or disabled to CICS.

OPENSTATUS({OPEN | CLOSED})
Specifies whether the TD queue is to be closed or opened in the CICS region.

PROCESSPROGRAMS({YES | NO})
Specifies whether any program specified in the CICS BAC control file is to be processed, depending on the status of the TD queue, as follows:

- If the command specifies OPENSTATUS(OPEN) *and* ENABLESTATUS(ENABLED), the associated programs are ENABLED.
- If the command specifies OPENSTATUS(CLOSED) *or* ENABLESTATUS(DISABLE), the associated programs are DISABLED.

PROCESSTRANSIDS({YES | NO})
Specifies whether any transaction specified in the CICS BAC control file is to be processed, depending on the status of the TD queue, as follows:

- If the command specifies OPENSTATUS(OPEN) *and* ENABLESTATUS(ENABLED), the associated transactions are ENABLED.
- If the command specifies OPENSTATUS(CLOSED) *or* ENABLESTATUS(DISABLE) is specified, the associated transactions are DISABLED.

TDQUEUE(*queuename*)
Specifies the CICS 4–character transient data queue name.

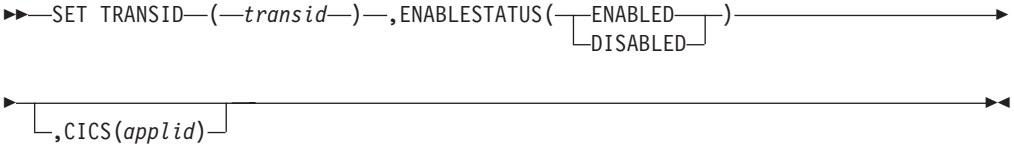
SET TRANSID

Function

To set the executable status of a transaction.

Syntax

SET TRANSID



Parameters

CICS(*applid*)
Specifies the applid of the CICS region to which this command is to be routed. If you omit this option, the command is routed to the default CICS region specified by the most recent DEFAULT command.

ENABLESTATUS({ENABLED | DISABLED})
Specifies whether the specified transaction is to be enabled or disabled in the target CICS region.

TRANSID(transid)
Specifies the 1-4 character CICS transaction ID that you want to enable or disable in the target CICS region.

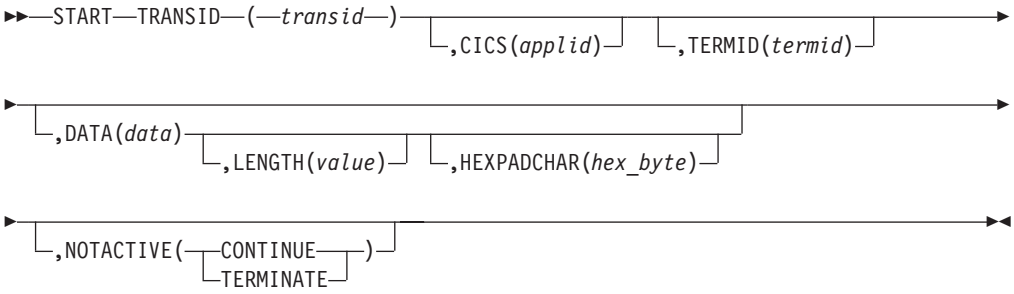
START

Function

To start a transaction.

Syntax

START



Description

The CICS BAC batch request utility issues message CBKxx5009I if the request executes successfully in the target CICS region. However, be aware that this simply means that the EXEC CICS START TRANSID(*name*) command was executed and returned a normal response, at which point the next command, if any exists, is processed. It does not necessarily mean that the specified transaction ran successfully. It could fail, for example, if the program that the transaction is to execute is disabled. If you think this is a possibility, you might want to precede the CICS BAC START request with a SET PROGAM(*progname*) ENABLESTATUS(ENABLED).

Parameters

CICS(*applid*)
Specifies the applid of the CICS region to which this command is to be routed. If you omit this option, the command is routed to the default CICS region specified by the most recent DEFAULTcommand.

DATA(data_string)
Specifies the data you want to pass to the transaction in CICS. You can specify up to a maximum of 256 bytes, with no restrictions on the contents, except that you cannot have an unmatched parenthesis within the outer parentheses that enclose the data. For example, DATA(abc(def) is not valid, but DATA(abc(123)) is valid.

If you specify the DATA parameter, you can also specify the LENGTH parameter.

HEXPADCHAR(*hex_byte*)
Specifies the hexadecimal character that you want to use to pad the data up to the data length as specified by the LENGTH parameter. For example, if you want to pad with spaces, specify HEXPADCHAR(40). If you omit this option the pad character defaults to a null value.

You can specify HEXPADCHAR only if you also specify both the DATA and LENGTH parameters

LENGTH(*hex_byte***)**
Specifies the length of the data to be passed to the CICS transaction being started. If you omit this option, the length defaults to the length of the actual data specified on the DATA operand. The maximum value you can specify is 256

You can specify LENGTH only if you also specify the DATA parameter.

NOTACTIVE({CONTINUE | TERMINATE})
Specifies the action you want the CICS BAC batch request utility to take if the target CICS region is not available, or if the CICS BAC startup procedure has not been run in the region, as follows:

CONTINUE
Specify CONTINUE if you want the CICS BAC batch request utility to ignore the command and continue on processing the next command in the input data set.

TERMINATE
Specify TERMINATE if you want the CICS BAC batch request utility to terminate because of the failed command.

TERMID(termid)
Specifies the 1-4 character CICS terminal ID name that you want the started transaction to be associated with. Note that the CICS BAC request server checks to see that the terminal identified by *termid* is defined to the CICS region and if it is not, it rejects the request.

TRANSID(transid)
Specifies the 1-4 character CICS transaction ID name you want to start asynchronously in the target CICS region.

Note: The CICS BAC request server can verify only that the EXEC CICS START command itself successfully executes, not that the transaction specified on the START command successfully completes, or even starts.

Chapter 5. CICS BAC workstation administration client

This chapter describes the CICS BAC workstation administration client.

The workstation administration client is only one of three tools CICS BAC provides to enable you to perform CICS BAC administration tasks. You can choose to use the workstation client exclusively in your environment. Alternatively, you can use the workstation client in conjunction with either or both of the other two tools (the CICS BAC ISPF administration interface and the file maintenance utility) or you can decide not to use the client at all.

You use the workstation administration client to create and maintain information about files, programs, transactions, and TD queues in the VSAM KSDS control files. The client is a Windows-based application. During the CICS BAC installation process, a data set is created that contains the workstation administration client program modules. The DDNAME for the data set is SCBKDWLD. Before you can install the client, you must download the appropriate module for the language you choose to the workstation (using FTP or a similar download method) and perform the client installation process. For more information on the languages supported by the client, and a list of the client modules in the SCBKDWLD data set, see Appendix B, "CICS BAC national language support (NLS)," on page 189. For more information about the client installation data set on the host, see the *CICS BAC Program Directory*. When you have downloaded the client module to the workstation, you can perform the client installation process. You can find instructions for installing, implementing, and using the client in the *CICS BAC Workstation User's Guide* and also in the online help feature integrated into the client.

Client functions

About this task

You can use the workstation administration client to perform almost all the maintenance functions for a CICS BAC VSAM KSDS control file. You can use the client to create and modify the following objects:

- Region properties
- Object default values
- File
- TD queue
- Program
- Transaction
- Group
- List

After you have you created one of these objects, you should use the client functions to upload it to a CICS region control file on the host, where it resides permanently. You download an object from a CICS region control file to the client for making modifications, and upload it when you finish. With the workstation administration client, you can also:

- Browse objects in the control file
- Perform CICS INQUIRE functions
- Delete objects from the control file.

Security requirements

All client functions that require a host communications server to perform an action (for example, download a group, upload a list, or download file records) require you to provide a host user ID and password, even if security checking is not active on the host. If security checking is active on the host for the communication server to which you are connecting, your user ID must have update authority for the CICS region control file that you are trying to access. If you do not have the required authorization, the request fails with a security violation. Your user ID does not require any other type of authority, such as TSO or CICS.

If active, client security checking is performed by the communication server processing your request using standard z/OS System Authorization Facility (SAF) checks. Thus you can use on the host any security manager that conforms to the standard SAF interface.

Data integrity of the control file

The CICS BAC control file on the host mainframe is the permanent storage location for all resource and default properties records that you are using to control access to CICS resources. When you use the workstation administration client to create and maintain control records, you should consider the existence of these objects on the workstation as temporary. That is, the mainframe components of CICS BAC do not use the copies residing on the workstation during any of its processing, and because there are no locks on the host control records when the records are downloaded to a workstation, they could be made obsolete by another user modifying the same object from a different workstation.

Because records on the control file are not locked when downloaded, when you create or edit objects on the workstation, you are recommended to upload them to the control file on the host and delete them from the workstation. Note that you can make deletion after upload an automatic function by using the appropriate option on the workstation administration client preferences panel. When you need to edit an object again, download it from the host, perform the required editing, and then upload it, assuring its integrity in the control file.

Chapter 6. CICS BAC ISPF administration interface

This chapter describes the CICS BAC ISPF administration interface, which provides an alternative method you can use to perform the functions provided by the CICS BAC workstation administration client, or the file maintenance utility. You can use the ISPF interface to create and maintain information about files, programs, transactions, and TD queues in a CICS region VSAM KSDS control file. The CICS BAC installation process creates the data sets that contain the ISPF interface, comprising REXX EXECs, panels, and messages. The ISPF interface currently supports only US English. For more information about the ISPF administration interface installation data sets, see the *CICS BAC Program Directory*.

For information about implementing the ISPF administration interface, see "Setting up and starting the ISPF administration interface" on page 35.

Information about using the ISPF administration interface is provided in the integrated online help facility and you can display this at any time while using the ISPF interface by pressing HELP. Message explanations for messages displayed by the ISPF administration interface are also provided in the integrated online help facility. Press HELP when a message is displayed for more information about the messages and the action that you should take.

ISPF administration interface functions

You can use the ISPF administration interface instead of the CICS BAC workstation administration client for all the maintenance functions of a CICS region VSAM KSDS control file. You can use the ISPF administration interface to create and modify the following objects:

- Region properties
- Object default values
- File
- TD queue
- Program
- Transaction
- Group
- List

In contrast with the workstation administration client, when you use the the ISPF administration interface, you are operating directly on the target CICS control file. This means that new or modified objects are saved immediately in the target CICS control file when you use the ISPF administration interface save function to save an object. In the case of the workstation administration client, you operate on a downloaded copy of an object and when you save your changes you are saving only a local copy, and the target CICS control file is not updated until you upload the object to the CICS region control file on the host through the communication server.

With the ISPF administration client, you can also:

- Browse objects in the control file
- Perform CICS INQUIRE functions

- Delete objects from the control file.

Security

The ISPF administration interface performs a security check on all the requests that you make to access data in a CICS region control file (for example, update a CICS region's properties, display a list of the file objects in a control file, or save a modified application group). The security checking is performed by the ISPF administration interface using standard z/OS System Authorization Facility (SAF) calls. This means that you can use any security manager that conforms to the standard SAF interface. See Chapter 9, "CICS BAC security," on page 123 for more information about CICS BAC security checking.

CICS BAC ISPF administration interface security is based on the resource name classifications described in topic "CICS BAC resource names" on page 124, where *accesstype* is ADMIN, and *objectType* and *objectName* are those associated with access type ADMIN. See "Administrative object types and object names" on page 125 for more information.

Data integrity considerations

The CICS BAC control file on the MVS system is the permanent storage location for all resource and default properties records that you use to control access to CICS resources through CICS BAC. Unlike the CICS BAC workstation administration client, the ISPF administration interface does not store object records in temporary files until you explicitly upload the object record back to the host. The ISPF administration interface has no process that is synonymous with the workstation administration client's upload and download process. Instead, when you select for display a particular object record, the record is read immediately from the target region control file. When you save a particular object record, it is immediately written to the target region VSAM KSDS control file.

The ISPF administration interface does not lock records on a CIC region control file when they are read by the interface. This means that, when you edit an object record and save it using the ISPF interface, you should be aware that the record could have been have changed since you originally read the object, possibly by another TSO user operating on the same CICS region control file. Also, there is no indication that an object you read from a control file has been previously downloaded from the control file to a workstation using the CICS BAC workstation administration client and still exists on the workstation. In this situation, a record updated using the ISPF administration interface could be overwritten by another changed version uploaded from the workstation.

Chapter 7. CICS BAC file maintenance utility

This chapter describes the CICS BAC file maintenance utility that you can use as an alternative to the workstation administration client or the ISPF administration interface, which are discussed in Chapter 5, "CICS BAC workstation administration client," on page 63 (see also the *CICS Batch Application Control for z/OS Workstation User's Guide*) and Chapter 6, "CICS BAC ISPF administration interface," on page 65. You can run the CICS BAC file maintenance utility as a batch job to create and maintain information in a CICS BAC VSAM KSDS control file.

File maintenance utility functions

The CICS BAC file maintenance utility supports all the functions provided by the workstation administration client and the ISPF administration interface, allowing you to issue add, update, list and delete commands that operate on the contents of the CICS BAC VSAM KSDS control file.

You can use the utility to create, modify, list, and delete the following types of object:

- File
- TD queue
- Program
- Transaction
- Application Group
- Application List

You can also use the utility to update and list CICS region objects, but you can't add or delete them.

The command syntax for the file maintenance utility supports the use of a generic character, allowing you to specify generic names in commands (see "Generic character usage" on page 74 for details).

File maintenance utility processing

The file maintenance utility needs to know the fully-qualified data set name of the control file of the CICS region for which it is processing file maintenance commands. This is so that the utility can allocate and access the appropriate control file in the event that CICS BAC is not active in the owning CICS region. In the normal course of events, the file maintenance utility reads and updates the control file through the CICS BAC request server task in the CICS region. For each CICS region, the corresponding control file data set name is held in the control file table, defined in a partitioned data set member named CBKCFTBL, as described in "The control file table" on page 23. Thus, to access CICS BAC control file for the CICS region for which it is processing commands, the file maintenance utility needs one of the following:

- The applid of the CICS region so that it can look up the data set name in the CBKCFTBL table. With this information, the file maintenance utility can dynamically allocate the CICS region control file to the job if CICS BAC is not active in the CICS region.
- The fully qualified data set name of the CICS BAC control file for the region. With this information, the file maintenance utility can look up the CICS applid

in the CBKCFTBL table, and can then communicate with the CICS region to determine whether CICS BAC is active in the region

You provide this information in the file maintenance utility JCL; see Figure 13 on page 72.

The following is a brief outline of the processing steps performed by the file maintenance utility:

Check for runtime parameters
The utility performs PARM string processing, checking for runtime parameters that can override the file maintenance utility program defaults.

Check for control file DD statement
As described above, if you do not provide an APPLID runtime parameter, the utility needs the fully qualified data set name of the CICS BAC control file.

Read commands from the input data set
If the runtime parameters are valid, the utility opens the CBKIN data set containing file maintenance commands and performs syntax checks on the commands.

Process valid commands
If the commands are valid and there are no syntax errors, the utility processes the commands sequentially. As each command is processed, the return code is checked against the maximum return code for that command and if it is less than, or equal to, the maximum return code value for the command, processing continues with the next command; otherwise processing is terminated. Any successful file activity up to the point of termination is preserved.

Print report
On completion of all command processing, the file maintenance utility produces a report.

File maintenance utility runtime parameters

There are several runtime parameters that you can pass to the file maintenance utility at its startup, all of which are optional. You can specify these parameters using any combination of the following methods:

- Allow the file maintenance utility to use its built-in defaults.
- Define your own values in CBKMAINT, which can be a member of either a CBKPARMS data set, or a data set in the MVS logical parmlib concatenation.
- Define you own values in the PARM string of the EXEC JCL statement.

The file maintenance utility searches for startup parameters, and processes them, in the following way:

1. **Default values**: The file maintenance utility first sets the default values for each parameter and continues at step 2.
2. **CBKPARMS data set**: The file maintenance utility searches for a CBKMAINT member in the CBKPARMS data set, if present. If the utility finds a CBKMAINT member it uses any runtime parameters in the member to override the defaults. If the member is empty, the defaults are not changed. When the file maintenance utility has finished processing the CBKMAINT member, it continues at step 4.
3. **Logical parmlib concatenation**: If the file maintenance utility cannot find a CBKMAINT member in CBKPARMS, it searches for a CBKMAINT member in the MVS logical parmlib concatenation. If the utility finds a CBKMAINT

member it uses any runtime parameters in the member to override the defaults. If the member is empty, the defaults are not changed.

4. **PARM string**: The file maintenance utility reads any parameters specified in the PARM string of the JCL EXEC statement, and uses these to override default or CBKMAINT parameter values.

Note: If you want to use only default values, or specify parameters in the PARM string only, do one of the following:

- Include an empty CBKMAINT member in the CBKPARMS data set. The file maintenance utility skips to step 4 above when it finds an empty CBKMAINT member.
- Authorize the file maintenance utility to have READ access to all the data sets in the MVS logical parmlib concatenation. The file maintenance utility continues with step 4 if it cannot find a CBKMAINT member. If the file maintenance utility does not have at least READ access to the logical parmlib concatenation, the job step fails with a 913 abend.

The runtime parameters enable you to:

- Specify the CICS region applid
- Specify the message severity level for producing system dumps
- Specify the language used by the file maintenance utility for messages written to the message log
- Specify trace settings.

The runtime parameters are described in detail as follows:

APPLID(name)
Specifies the CICS region applid. If CICS BAC is not active in the specified CICS region, the file maintenance utility can use the applid to locate the region control file table (CBKCFTBL) entry and obtain the fully-qualified name of the CICS region control file, which the utility can then update directly.

If you omit this option, you must specify the CBKCNTL DD statement in the batch job step of the file maintenance utility. The file maintenance utility uses the data set name specified on the CBKCNTL DD statement to perform a reverse lookup of the control file table (CBKCFTBL) to find the corresponding CICS region applid.

If you omit both APPLID and the CBKCNTL DD statement, the job step terminates with an error message (see message CBK*xx*7010 on page "CBKxx7010E" on page 156).

CICSMIRRORTRANS({CSMI | *transid*})
Specifies the name of the CICS mirror transaction to be specified on EXCI requests issued by the file maintenance utility. When the EXCI request is executed in the target CICS region, it runs under this transaction ID as an MRO mirror transaction.

CSMI This is the normal CICS EXCI default mirror transaction ID.

transid Specifies the 4-character name that you want CICS to use as the name of the MRO mirror transaction when it executes the EXCI requests it receives from the file maintenance utility. To avoid confusion with other mirror transactions that might be running in your CICS regions, you might want to choose a unique ID to identify CICS BAC file

maintenance utility mirror transactions. Define the appropriate transaction ID in the CICS region for a CICS mirror transaction, ensuring the following:

- The program associated with the transaction ID must be DFHMIRS
- The profile for the transaction ID must be DFHCICSA.

You might consider using the CICS-supplied CSMI transaction definition as a basis for this transaction ID.

LANGUAGE({ENU | JPN})
Specifies the language used by the file maintenance utility for messages written to the message log (DD name CBKPRINT). See Appendix B, "CICS BAC national language support (NLS)," on page 189 for more information. The values you can specify are:

ENU US English

JPN Japanese

Note: Specify the language parameter only on the PARM string of the JCL EXEC statement for the file maintenance utility. The parameter is ignored if it is specified in the CBKMAINT member in either the CBKPARMS data set or the logical parmlib concatenation.

MAXDOWNLOADPAGES(256|*number*)
Specifies the maximum number of 4KB pages the file maintenance utility is to allocate as a buffer for downloading data from the target CICS region.

number
Specifies the number of pages in the range 1 through 999. The default is 256.

Note that if the allocated buffer is too small for the command being processed, the command fails and the file maintenance utility issues message CBK*xx*7310. For example, if you issue a LIST command with only the generic character as the identifier, the volume of data returned could be quite large if the region control file contains multiple object records of the specified resource type.

MSGLEVEL(16 | value)
Specifies when the file maintenance utility is to produce a dump, based on the return code severity. The default return code value is 16.

When the file maintenance utility is terminating, its final step is to compare the value of the return code with the value of the MSGLEVEL option. If the return code is greater than or equal to MSGLEVEL, the job step is terminated with abend code U3502 and takes a dump. Thus by default, any return code of 16 or greater causes the utility to abend and produce a dump.

MSGLEVEL is intended primarily for support and diagnosis purposes. For example, if Technical Support need a dump when you have a condition where the utility is terminating with return code 8, you can specify EXEC PGM=CBKFMAIN,PARM='MSGLEVEL(8)', and run the job again to force an abend and a dump.

TRACEOPTIONS({C0 | options})
Specifies the types of trace entries you want CICS BAC to create. There are eight trace entry types, each controlled by a single bit in the TRACEOPTIONS hexadecimal byte flag. Specify this option only at the direction of a Technical Support representative.

C0 The hexadecimal value C0 is the default trace setting.

options

Specify, in hexadecimal form, the options requested by Technical Support. For more details of the hexadecimal byte flag, see the TRACEOPTIONS parameter under "Communication server runtime parameters" on page 27

TRACESIZE({1000 | number-of-entries})

Specifies the size of the trace table in terms of the number of entries to be traced for diagnostic purposes. Change this value only at the direction of Technical Support.

1000 The default trace size.

number

Specify the trace size as requested by Technical Support.

File maintenance utility JCL

A file maintenance utility job step requires the following statements in the JCL:

- An EXEC statement for program CBKFMAIN (note the program name must be CBKFMAIN).
- A DD statement for the CBKPARMS data set that contains:
  - The control file table member, CBKCFTBL, that identifies the control file data set name for each CICS region
  - The parameter member, CBKMAINT, that contains the file maintenance utility runtime parameters

  Alternatively, you can include the CBKCFTBL and CBKMAINT members in a data set that is in the MVS logical parmlib concatenation, in which case you do not need the CBKPARMS DD statement. However, if you put the CBKCFTBL member and the optional CBKMAINT parameter member in the MVS logical parmlib concatenation, authorize the file maintenance utility to have READ access to all the data sets in the MVS logical parmlib concatenation. The file maintenance utility searches for CBKCFTBL and CBKMAINT in the logical parmlib concatenation if either of these members is not found in the CBKPARMS data set.
- A STEPLIB DD statement that references the SCBKLOAD and the SDFHEXCI data set containing the load modules required by the file maintenance utility if the data sets are not in the MVS linklist.
- A DD statement for the CBKIN data set, from which the CICS BAC file maintenance utility reads the commands it is to execute.
- A DD statement for the CBKPRINT data set, to which the CICS BAC file maintenance utility writes its output messages.

The JCL in Figure 13 on page 72 shows a sample job to run the file maintenance utility.

```
//CBKFMU EXEC PGM=CBKFMAIN,PARM='APPLID(CICSPROD)'
//STEPLIB  DD DISP=SHR,DSN=hlq.SCBKLOAD
//         DD DISP=SHR,DSN=hlq.SDFHEXCI
//CBKPARMS DD DISP=SHR,DSN=hlq.CBKPARMS
//CBKPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//*****************************************************************
//* Un-comment the following DD statement if you need to add a
//* SYSMDUMP DD statement. You might need to modify this statement
//* to meet your installation's requirements.
//*****************************************************************
//*SYSMDUMP DD DISP=(,CATLG),DSN=hlq.sysmdump.dataset,
//* UNIT=SYSDA,SPACE=(CYL,(50,25)),
//* DCB=(DSORG=PS,RECFM=FBS,LRECL=4160,
//* BLKSIZE=24960 //CBKPRINT DD SYSOUT=*
//CBKIN DD *
ADD FILE(FILEA),RLSOPTIONS(TERMINATE),RLSRETCODE(12)
```

Figure 13. Sample JCL to run the file maintenance utility

Note:

1. Change the characters *hlq* to the high-level qualifiers for these data sets:
   - SCBKLOAD is the name of the CICS BAC load library
   - SDFHEXCI is the name of the CICS TS EXCI load library
   - CBKPARMS is the name of your CICS BAC parmlib.
2. Also change *hlq.sysmdump.dataset* to the name of your own dump data set.
3. If you put the CBKCFTBL member and the optional CBKMAINT parameter member in the MVS logical parmlib concatenation, authorize the batch request utility to have READ access to all the data sets in the MVS logical parmlib concatenation.
4. See "The CBKPARMS data set" on page 24 for information about defining the CBKPARMS data set.

There is some sample JCL for a file maintenance utility job shown in member CBKFMJCL, which you can find in the SCBKSAMP library. You can copy this and modify it for your own use.

File maintenance utility commands

About this task

The section covers the following topics:

- Command format
- Syntax notation for file maintenance utility commands
- The commands

The following sections describe all the commands that you can use to maintain a CICS BAC VSAM control file using the file maintenance utility in a batch job.

Command format

About this task

The basic format of a CICS BAC file maintenance utility input command is as follows:

```
COMMAND NAME Parameter1(value1),Parameter2(value2),...,Parametern(valuen)
```

The rules for coding commands are as follows:

- The command line is restricted to column positions 1 to 71 inclusive
- You indicate that a parameter string continues on the next line by specifying any non-blank character in column 72.
- The command name must be the first non-blank word on the line, and it must be followed by one or more spaces.
- You follow the command name with one or more parameter(*value*) strings (or none).
- You can abbreviate parameters and enumerated parameter values (such as "OPEN" and "CLOSED") to their shortest unique string. However, CICS BAC does not guarantee that an abbreviation that is valid in this release will be valid in a future release. Therefore, you should consider using abbreviations that are not the shortest abbreviation possible for a parameter.
- If you specify more than one parameter string they must be separated by a comma.
- Some parameters are optional, and others are required.
- If you omit an optional parameter, that parameter is not used in the command.
- You pass your commands to the CICS BAC file maintenance utility in a data set with the DDNAME CBKIN.
- The order in which command parameters are shown in the syntax diagrams in this chapter does not imply a required sequence. You can order the parameters within a command in any sequence.

The above rules are explained in further detail, as appropriate, in the description of each command. The commands provided by CICS BAC are:

- "UPDATE TDQUEUE" on page 112 (see page "UPDATE TDQUEUE" on page 112)
- "UPDATE TRANSID" on page 113 (see page "UPDATE TRANSID" on page 113)

Syntax notation for file maintenance utility commands

The syntax for each file maintenance utility input command is presented in the form of a diagram, commonly referred to as a **railroad diagram**. The syntax, which is interpreted by following the dashed lines and arrows from left to right, and top to bottom, is described in the following table.

Symbol	Meaning
`>>-----A-----><`	A required parameter that you must provide.

`>>-----A-----><`
`     +-B-+`
`     +-C-+`
	A set of alternative parameters, one of which you must provide

`     +-A-+`
` >>-----------><`
`     +-B-+`
`     +-C-+`
	A set of optional parameters, where A is the default. You may provide only one of these options.

`>>-----------><`
`     +-A-+`
`     +-B-+`
	A set of optional parameters. None are required, and there is no default. You may provide only one of these.

`>>-- Name +-><`
 	
`Name:`
`>>--A----B---->`
	Denotes a named section provided elsewhere in the command syntax.

`Punctuation and upper case characters`	Code exactly as documented.
`Lowercase, italicized characters`	Requires you to code your own text to replace these fields.

Generic character usage: The command syntax for the file maintenance utility supports the use of a generic character, which you can use in the following commands:

- In the DELETE *objecttype*(*objname*) command, *objname* can contain generic characters
- On any of the REMOVE*xxxx* parameters of the various objects.
  - FILE and TDQUEUE objects have the REMOVETRANSIDS and REMOVEPROGRAMS parameters
  - APPGROUP object has the REMOVEFILES, REMOVETDQUEUES, REMOVETRANSIDS and REMOVEPROGRAMS parameters.

  All these named parameters accept generic characters in the name operand.

The generic character is an * (asterisk), where an * matches none or one or more characters. Thus * by itself means all names, so to remove all the files from an application group, use REMOVEFILES(*). Operand ab* matches anything beginning with the characters ab, such as abc, abcde, and so on.

Omitting parameters from commands:
Except for the REPLACE parameter on an ADD *resource* command and the ADDIFNEW parameter on an UPDATE *resource* command, none of the other

parameters have defaults. If you omit a parameter from an ADD *resource* or UPDATE *resource* command, the file maintenance utility takes the value for a missing parameter as follows:

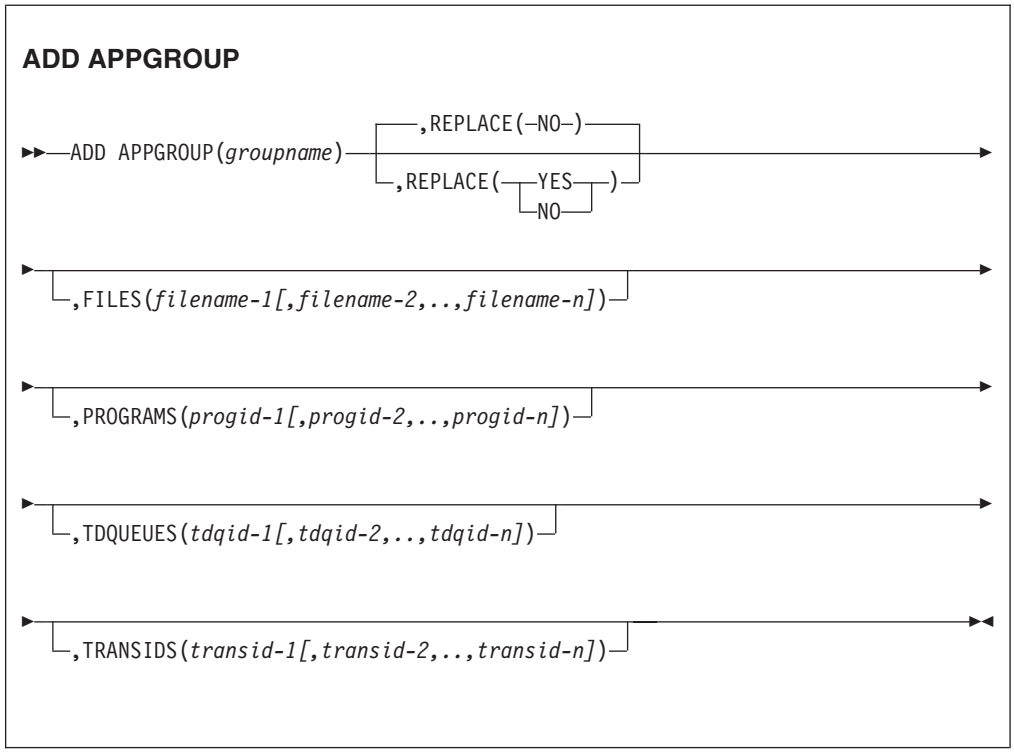
- If you omit a parameter from an ADD FILE, ADD PROGRAM, ADD TDQUEUE, or ADD TRANSID command, the file maintenance utility derives the missing parameter value from the $Default record on the CICS control file for the object type. There is no action for ADD GROUP and ADD LIST commands because these objects do not have a $Default record.
- If you omit a parameter from an UPDATE command, the value in the existing record is unchanged.

ADD APPGROUP

Function

Add a new application group object to the target CICS region control file.

Syntax



Description

When you have created CICS region definitions and related resource object definitions, you can associate some of them with a specific application, such as a payroll application. You give the application group a name that is identifies it with the application, and add objects that are used by the application. You can also remove objects that are no longer applicable to the application.

Note: A resource object does not necessarily have to be defined in the control file to be included in an application group. You can add a resource to a group that is not defined in the control file, and it will be processed as part of a SET GROUP

command if the CICS region undefined object processing options permit it. See the UNDEFINED*xxxx* set of parameters on the "UPDATE REGION" on page 100 command on page "UPDATE REGION" on page 100.

Parameters

Note: When adding file names, program names, TD queue IDs, transaction IDs to a group, the total number of all objects cannot exceed 512.

APPGROUP(*groupname***)**
Specifies the name of the application group that is to contain the names of the resources that make up the application group.

FILES(*filename-1***[,***filename-2***,..,***filename-n***])**
Specifies the name of one or more files to be added to the application group. File names must be valid CICS file names up to a maximum of 8 characters.

If you enter more than one file name, separate the names with a comma or a blank.

PROGRAMS(*progid-1***[,***progid-2***,..,***progid-n***])**
Specifies the name of one or more programs to be added to the application group. Program names must be valid CICS program names up to a maximum of 8 characters.

If you enter more than one program name, separate the names with a comma or a blank.

REPLACE({NO|YES})
Specifies the action the file maintenance utility is to take if an application group object record already exists for the application group name you are trying to add.

NO If there is already an application group object in the CICS region control file for this application group name, reject the ADD APPGROUP command.

YES If there is already an application group object in the CICS region control file for this application group name, replace it with the information in this ADD APPGROUP command.

TDQUEUES(*tdqid-1***[,***tdqid-2***,..,***tdqid-n***])**
Specifies the name of one or more TD queues to be added to the application group. TD queue IDs must be valid CICS TD queue IDs up to a maximum of 4 characters.

If you enter more than one TD queue ID, separate the IDs with a comma or a blank.

TRANSIDS(*transid-1***[,***transid-2***,..,***transid-n***])**
Specifies the name of one or more transaction IDs to be added to the application group. Transaction IDs must be valid CICS transaction IDs up to a maximum of 4 characters.

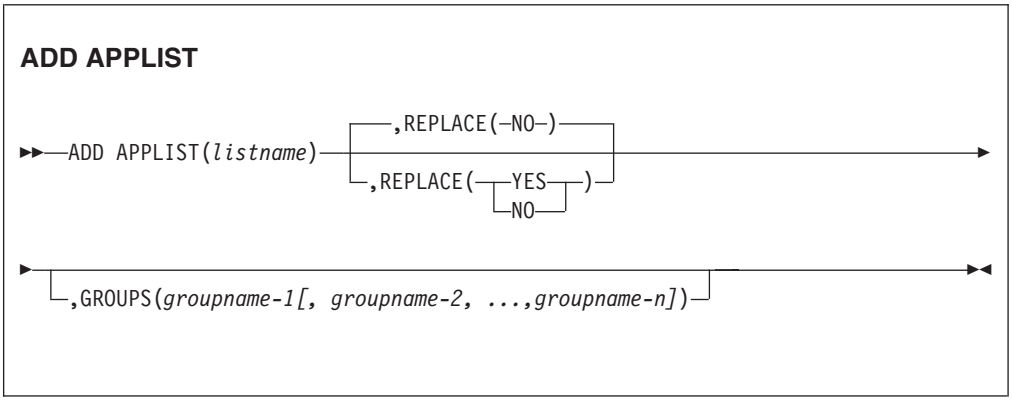
If you enter more than one transaction ID, separate the IDs with a comma or a blank.

ADD APPLIST

Function

Add a new application list object to the target CICS region control file.

Syntax



Description

If you have multiple application groups in a CICS region control file, you can combine them into an application list that can be processed as a single object during batch request utility job steps.

Parameters

APPLIST(*listname***)**
Specifies the name of the application list that is to contain one or more application group names.

GROUPS(*groupname-1***[, *groupname-2*, ...,*groupname-n*])**
Specifies the names of the application groups that are to comprise the application list. If you specify more than one group name, separate the names with a comma or a blank.

The maximum number of group names you can add is 512.

REPLACE({NO|YES})
Specifies the action the file maintenance utility is to take if an application list object record already exists for the application list name you are trying to add.

NO If there is already an application list object in the CICS region control file for this application list name, reject the ADD APPLIST command.

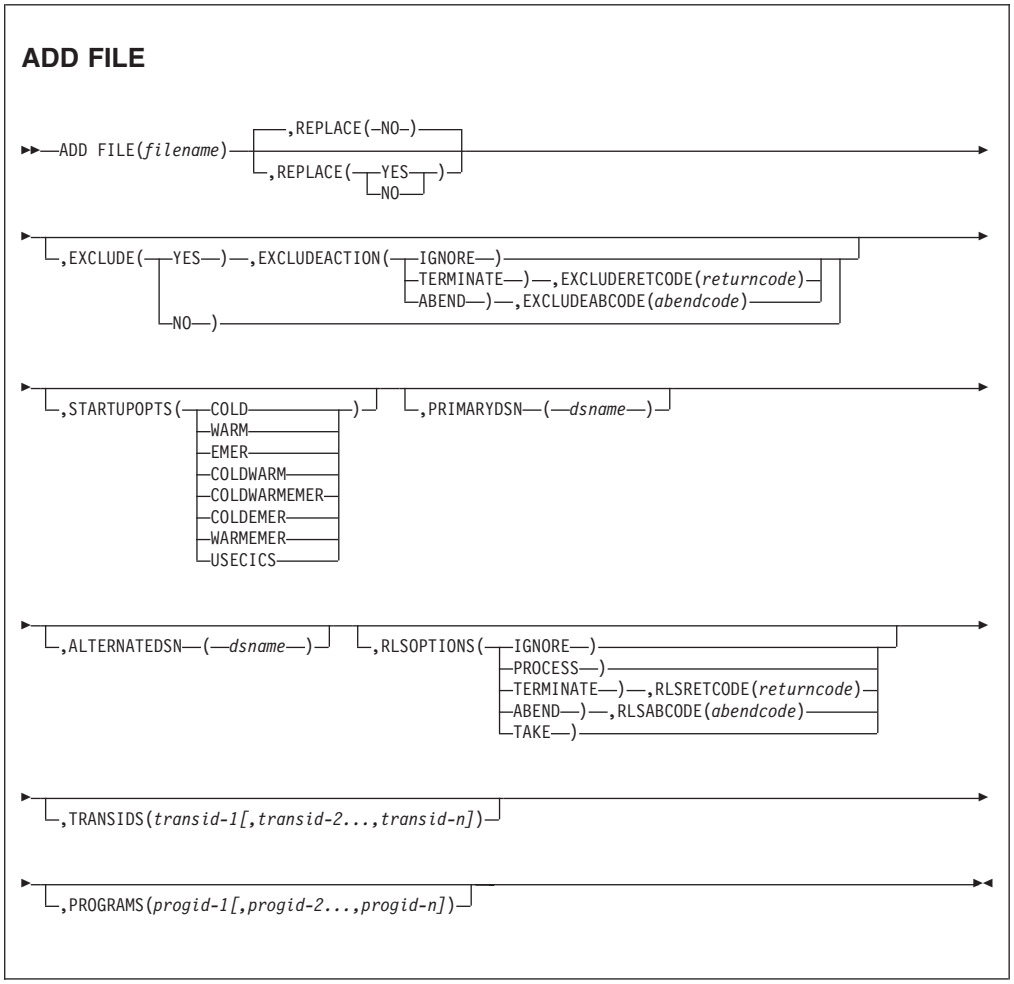
YES If there is already an application list object in the CICS region control file for this application list name, replace it with the information in this ADD APPLIST command.

ADD FILE

Function

Add a new file object to the target CICS region control file.

Syntax



Description

You can use the ADD FILE command to add a file object record to a CICS region control file. If a record for the specified file name already exists in the region control file, the action taken by the file maintenance utility is determined by the REPLACE parameter.

Parameters

ALTERNATEDSN(*datasetname***)**
Specifies the fully-qualified data set name CICS BAC is to use when a batch request utility job step invokes a SET FILE command that specifies DSN(ALTERNATE) parameter.

EXCLUDE({NO | YES})
Specifies that CICS BAC is to exclude this file from all processing, including batch request utility processing and CICS state monitoring.

NO CICS BAC is not to exclude this file.

YES CICS BAC is to exclude this file from all processing. If you specify YES, also specify the action that CICS BAC is to take if it encounters a batch request utility command for this file (see the EXCLUDEACTION parameter).

EXCLUDEABCODE(*abendcode***)**
Specifies the numeric abend code CICS BAC is to use if you specify EXCLUDEACTION(ABEND). You can specify an abend code value in the range 1 through 4095.

EXCLUDEACTION(IGNORE|TERMINATE|ABEND)
Specifies the action CICS BAC is to take if you specify EXCLUDE(YES). You can choose one of the following three options:

IGNORE
CICS BAC is to ignore the command as if it was not present and continue processing with the next command.

TERMINATE
CICS BAC is not to process any further commands in the job step, and the batch request utility job step is to terminate with the specified return code (see the EXCLUDERETCODE parameter).

ABEND
CICS BAC is not to process any further commands in the job step, and the batch request utility job step is to abend with the specified abend code (see the EXCLUDEABCODE parameter).

EXCLUDERETCODE(*returncode***)**
Specifies the numeric return code CICS BAC is to use if you specify EXCLUDEACTION(TERMINATE). You can specify a return code value in the range 1 through 4095.

FILE(*filename***)**
Specifies the 8-character file name that you want to add to a CICS region control file.

PRIMARYDSN(*datasetname***)**
Specifies the fully-qualified data set name CICS BAC is to use when a batch request utility job step invokes a SET FILE command that specifies DSN(PRIMARY) parameter.

PROGRAMS(*progid-1***[,***progid-2...,progid-n***])**
Specifies one or more programs to be added to the list of programs in the file object record in the CICS region control file. Program names must be valid CICS application program names up to a maximum of 8 characters.

If you enter more than one program name, separate the names with a comma or a blank. The maximum number of program names you can add is 512.

These are programs to be processed by the batch request utility when it processes a SET FILE command for this file. If the command sets the file state to OPENSTATUS(OPEN) and ENABLESTATUS(ENABLED), the specified programs are set to ENABLESTATUS(ENABLED) *after* the file state has been successfully set. If the batch request utility command sets the file state to OPENSTATUS(CLOSED) or ENABLESTATUS(DISABLED), the programs are set to ENABLESTATUS(DISABLED) *before* the file state is set.

For more information, see the PROCESSPROGRAMS parameter on the SET FILE command on page “SET FILE” on page 51.

REPLACE({NO|YES})
Specifies the action the file maintenance utility is to take if a file object record already exists for the file name you are trying to add.

NO If there is already a file object in the CICS region control file for this file name, reject the ADD FILE command.

YES If there is already a file object in the CICS region control file for this file name, replace it with the information in this ADD FILE command.

RLSABCODE(*abendcode***)**
Specifies the numeric abend code CICS BAC is to use if you specify RLSOPTION(ABEND). You can specify an abend code value in the range 1 through 4095.

RLSOPTIONS({IGNORE|PROCESS|TERMINATE|ABEND|TAKE})
Specifies the default action to be taken if the CICS BAC request server receives a command to open or close the file, and the file is defined with RLSACCESS(YES). You can specify one of the following options:

IGNORE
CICS BAC is to ignore open and close commands for this file. However, programs and transaction IDs associated with the file object are to be processed as if the command was processed.

PROCESS
CICS BAC is to process the command. If the command is to close the file, CICS BAC is to quiesce the data set associated with the file (if the file is not already quiesced) before closing the file. If the command is to open the file, CICS BAC is to unquiesce the data set associated with the file (if it is currently quiesced) before opening the file.

TERMINATE
CICS BAC is to terminate the job step with the return code specified on the RLSRETCODE parameter.

ABEND
CICS BAC is to abend the job step with the abend code specified on the RLSABCODE parameter.

TAKE CICS BAC is to take the action specified in the CICS region settings.

RLSRETCODE(*abendcode***)**
Specifies the numeric return code CICS BAC is to use if you specify RLSOPTION(TERMINATE). You can specify a return code value in the range 1 through 4095.

STARTUPOPTS({COLD|WARM|EMER|COLDWARM|COLDWARMEMER|COLDEMER|WARMEMER|USECICS})
Specifies the types of CICS startups during which CICS BAC should set the state of this file to its last requested state, including its open status, enable status, data set name, and access options. If you specify any of the startup types, CICS BAC resets this file to its last requested state at the end of that type of CICS region initialization. The options are as follows:

COLD CICS BAC is to restore the last requested state of the file during a cold start of the CICS region.

WARM
CICS BAC is to restore the last requested state of the file during a warm start of the CICS region.

EMER CICS BAC is to restore the last requested state of the file during an emergency restart of the CICS region.

COLDWARM
CICS BAC is to restore the last requested state of the file during cold and warm starts of the CICS region.

COLDWARMEMER
CICS BAC is to restore the last requested state of the file during cold, warm, and emergency starts of the CICS region.

COLDEMER
CICS BAC is to restore the last requested state of the file during cold and emergency starts of the CICS region.

WARMEMER
CICS BAC is to restore the last requested state of the file during warm and emergency starts of the CICS region.

USECICS
Depending on the type of startup, CICS BAC is to use the CICS region default startup property (as defined by the region properties record for the CICS region) to determine whether or not it should reset a file to its last requested state.

TRANSIDS(*transid-1***[,***transid-2...,transid-n***])**
Specifies one or more transactions to be added to the list of transaction in the file object record in the CICS region control file. Transaction IDs must be valid CICS transaction IDs up to a maximum of 4 characters.

If you enter more than one transactionID, separate the IDs with a comma or a blank. The maximum number of transaction IDs you can add is 512.

These are transactions to be processed by the batch request utility when it processes a SET FILE command for this file. If the command sets the file state to OPENSTATUS(OPEN) and ENABLESTATUS(ENABLED), the specified transactions are set to ENABLESTATUS(ENABLED) *after* the file state has been successfully set. If the batch request utility command sets the file state to OPENSTATUS(CLOSED) or ENABLESTATUS(DISABLED), the transactions are set to ENABLESTATUS(DISABLED) *before* the file state is set.

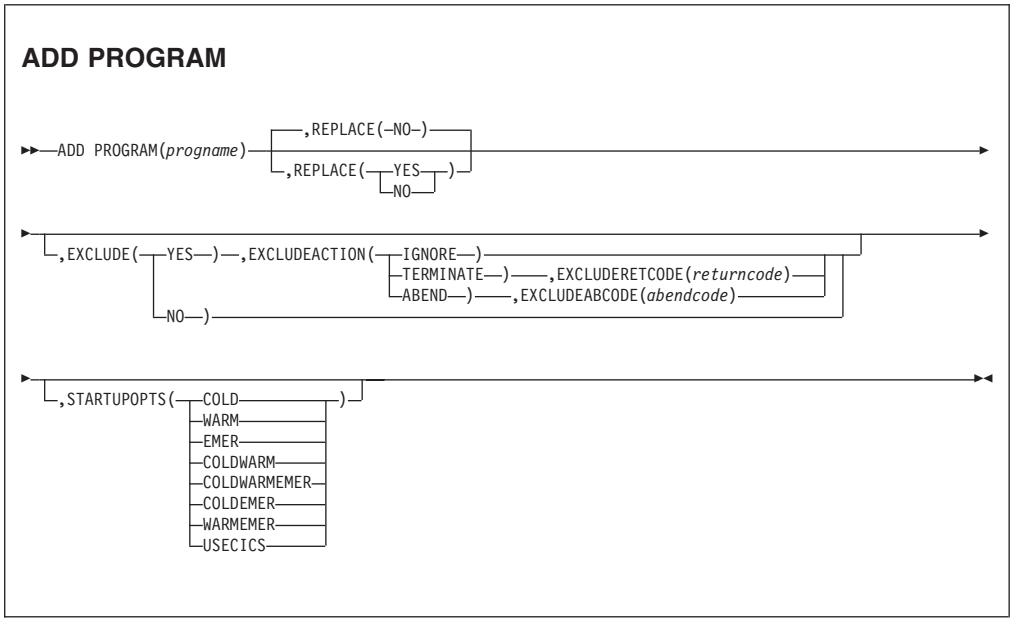
For more information, see the PROCESSTRANSIDS parameter on the SET file command on page “SET FILE” on page 51).

ADD PROGRAM

Function

Add a new program object to the target CICS region control file.

Syntax



Description

You can define exclude and startup properties for a program object. These properties determine how program objects are to be processed by CICS BAC during CICS region initialization.

Parameters

EXCLUDE({NO|YES})
Specifies that CICS BAC is to exclude this program from all processing, including batch request utility processing and CICS state monitoring.

NO CICS BAC is not to exclude this program.

YES CICS BAC is to exclude this program from all processing. If you specify YES, also specify the action that CICS BAC is to take if it encounters a batch request utility command for this program (see the EXCLUDEACTION parameter).

EXCLUDEABCODE(*abendcode*)
Specifies the numeric abend code CICS BAC is to use if you specify EXCLUDEACTION(ABEND). You can specify an abend code value in the range 1 through 4095.

EXCLUDEACTION({IGNORE|TERMINATE|ABEND})
Specifies the action CICS BAC is to take if you specify EXCLUDE(YES). You can choose one of the following three options:

IGNORE
CICS BAC is to ignore the command as if it was not present and continue processing with the next command.

TERMINATE
CICS BAC is not to process any further commands in the job step, and the batch request utility job step is to terminate with the specified return code (see the EXCLUDERETCODE parameter).

ABEND
CICS BAC is not to process any further commands in the job step, and the batch request utility job step is to abend with the specified abend code (see the EXCLUDEABCODE parameter).

EXCLUDERETCODE(*returncode***)**
Specifies the numeric return code CICS BAC is to use if you specify EXCLUDEACTION(TERMINATE). You can specify a return code value in the range 1 through 4095.

PROGRAM(*progname***)**
Specifies the name of the program object you are adding to the CICS region control file.

REPLACE({NO|YES})
Specifies the action the file maintenance utility is to take if a program object record already exists for the program name you are trying to add.

NO If there is already a program object in the CICS region control file for this program name, reject the ADD PROGRAM command.

YES If there is already a program object in the CICS region control file for this program name, replace it with the information in this ADD PROGRAM command.

STARTUPOPTS({COLD|WARM|EMER|COLDWARM|COLDWARMEMER|COLDEMER|WARMEMER|USECICS})
Specifies the types of CICS startups during which CICS BAC should set the state of this program to its last requested enabled state. If you specify any of the startup types, CICS BAC resets this program to its last requested state at the end of that type of CICS region initialization. The options are as follows:

COLD CICS BAC is to restore the last requested state of the program during a cold start of the CICS region.

WARM
CICS BAC is to restore the last requested state of the program during a warm start of the CICS region.

EMER CICS BAC is to restore the last requested state of the program during an emergency restart of the CICS region.

COLDWARM
CICS BAC is to restore the last requested state of the program during cold and warm starts of the CICS region.

COLDWARMEMER
CICS BAC is to restore the last requested state of the program during cold, warm, and emergency starts of the CICS region.

COLDEMER
CICS BAC is to restore the last requested state of the program during cold and emergency starts of the CICS region.

WARMEMER
CICS BAC is to restore the last requested state of the program during warm and emergency starts of the CICS region.

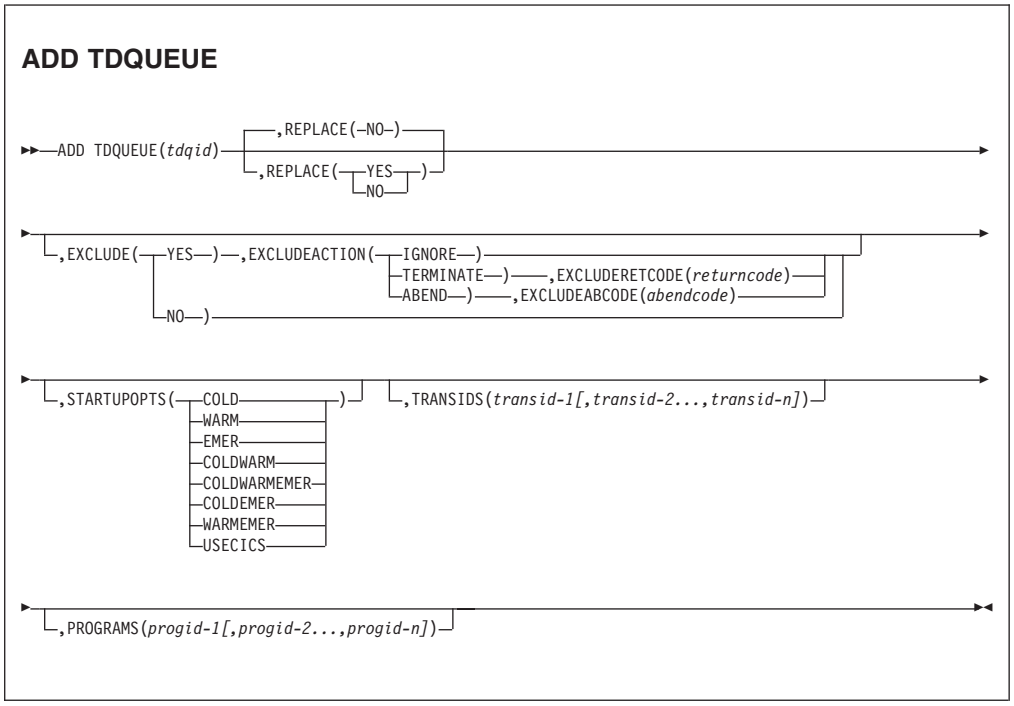
USECICS
Depending on the type of startup, CICS BAC is to use the CICS region default startup property (as defined by the region properties record for the CICS region) to determine whether or not it should reset a program to its last requested state.

ADD TDQUEUE

Function

Add a new TD queue object to the target CICS region control file.

Syntax



Description

You can define a set of customized properties for a TD queue object. These properties determine how TD queue objects are to be processed by CICS BAC during CICS region initialization.

Parameters

EXCLUDE({NO | YES})
Specifies that CICS BAC is to exclude this TD queue from all processing, including batch request utility processing and CICS state monitoring.

NO CICS BAC is not to exclude this TD queue.

YES CICS BAC is to exclude this TD queue from all processing. If you specify YES, also specify the action that CICS BAC is to take if it encounters a batch request utility command for this TD queue (see the EXCLUDEACTION parameter).

EXCLUDEABCODE(*abendcode*)
Specifies the numeric abend code CICS BAC is to use if you specify EXCLUDEACTION(ABEND). You can specify an abend code value in the range 1 through 4095.

EXCLUDEACTION({IGNORE|TERMINATE|ABEND})
Specifies the action CICS BAC is to take if you specify EXCLUDE(YES). You can choose one of the following three options:

IGNORE
CICS BAC is to ignore the command as if it was not present and continue processing with the next command.

TERMINATE
CICS BAC is not to process any further commands in the job step, and the batch request utility job step is to terminate with the specified return code (see the EXCLUDERETCODE parameter).

ABEND
CICS BAC is not to process any further commands in the job step, and the batch request utility job step is to abend with the specified abend code (see the EXCLUDEABCODE parameter).

EXCLUDERETCODE(*returncode***)**
Specifies the numeric return code CICS BAC is to use if you specify EXCLUDEACTION(TERMINATE). You can specify a return code value in the range 1 through 4095.

PROGRAMS(*progid-1***[,***progid-2...,progid-n***])**
Specifies one or more programs to be added to the list of programs in the TD queue object record in the CICS region control file. Program names must be valid CICS application program names up to a maximum of 8 characters.

If you enter more than one program name, separate the names with a comma or a blank. The maximum number of program names you can add is 512.

These are programs to be processed by the batch request utility when it processes a SET TDQUEUE command for this TD queue. If the command sets the TD queue state to OPENSTATUS(OPEN) and ENABLESTATUS(ENABLED), the specified programs are set to ENABLESTATUS(ENABLED). If the batch request utility command sets the TD queue state to OPENSTATUS(CLOSED) or ENABLESTATUS(DISABLED), the programs are set to ENABLESTATUS(DISABLED).

For more information, see the PROCESSPROGRAMS parameter on the SET TDQUEUE command on page “SET TDQUEUE” on page 59.

REPLACE({NO|YES})
Specifies the action the file maintenance utility is to take if a TD queue object record already exists for the TD queue name you are trying to add.

NO If there is already a TD queue object in the CICS region control file for this TD queue name, reject the ADD TDQUEUE command.

YES If there is already a TD queue object in the CICS region control file for this TD queue name, replace it with the information in this ADD TDQUEUE command.

STARTUPOPTS({COLD|WARM|EMER|COLDWARM|COLDWARMEMER|COLDEMER|WARMEMER|USECICS})
Specifies the types of CICS startups during which CICS BAC should set the state of this TD queue to its last requested state, including its open status and enable status. If you specify any of the startup types, CICS BAC resets this TD queue to its last requested state at the end of that type of CICS region initialization. The options are as follows:

COLD CICS BAC is to restore the last requested state of the TD queue during a cold start of the CICS region.

WARM
CICS BAC is to restore the last requested state of the TD queue during a warm start of the CICS region.

EMER CICS BAC is to restore the last requested state of the TD queue during an emergency restart of the CICS region.

COLDWARM
CICS BAC is to restore the last requested state of the TD queue during cold and warm starts of the CICS region.

COLDWARMEMER
CICS BAC is to restore the last requested state of the TD queue during cold, warm, and emergency starts of the CICS region.

COLDEMER
CICS BAC is to restore the last requested state of the TD queue during cold and emergency starts of the CICS region.

WARMEMER
CICS BAC is to restore the last requested state of the TD queue during warm and emergency starts of the CICS region.

USECICS
Depending on the type of startup, CICS BAC is to use the CICS region default startup property (as defined by the region properties record for the CICS region) to determine whether or not it should reset a TD queue to its last requested state.

TDQUEUE(*tdqid***)**
Specifies the name of the TD queue object that you are adding to the CICS region control file.

TRANSIDS(*transid-1***[,***transid-2...,transid-n***])**
Specifies one or more transactions to be added to the list of transaction in the file object record in the CICS region control file. Transaction IDs must be valid CICS transaction IDs up to a maximum of 4 characters.

If you enter more than one transaction ID, separate the IDs with a comma or a blank. The maximum number of transaction IDs you can add is 512.

These are transactions to be processed by the batch request utility when it processes a SET TDQUEUE command for this TD queue. If the command sets the TD queue state to OPENSTATUS(OPEN) and ENABLESTATUS(ENABLED), the specified transactions are set to ENABLESTATUS(ENABLED). If the batch request utility command sets the TD queue state to OPENSTATUS(CLOSED) or ENABLESTATUS(DISABLED), the transactions are set to ENABLESTATUS(DISABLED).

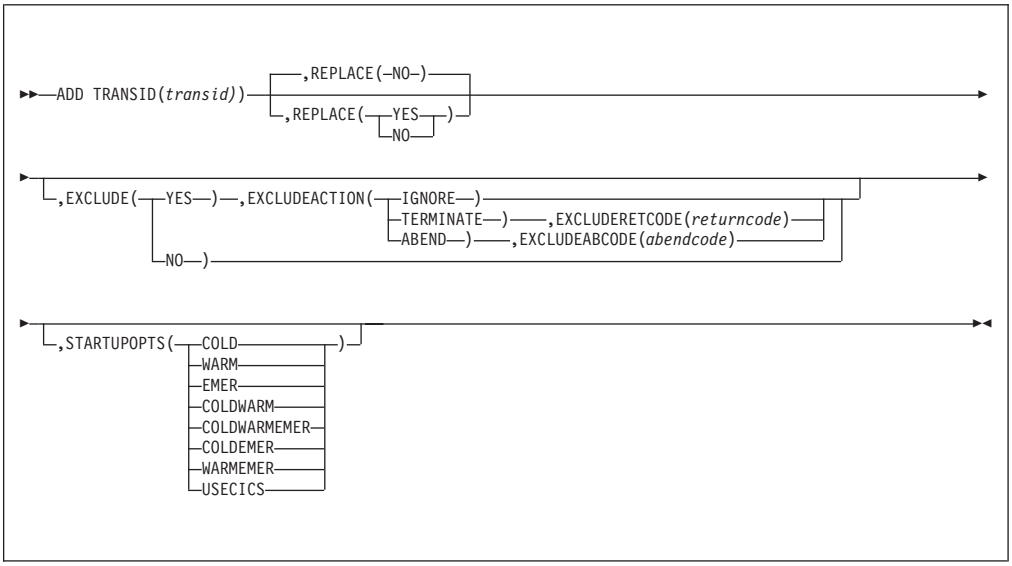
For more information, see the PROCESSTRANSIDS parameter on the SET TDQUEUE command on page “SET TDQUEUE” on page 59.

ADD TRANSID

Function

Add a new transaction ID object to the target CICS region control file.

Syntax



Description

You can define a set of customized properties for a transaction ID object. These properties determine how transaction ID objects are to be processed by CICS BAC during CICS region initialization.

Parameters

EXCLUDE({NO | YES})
Specifies that CICS BAC is to exclude this transaction from all processing, including batch request utility processing and CICS state monitoring.

NO CICS BAC is not to exclude this transaction.

YES CICS BAC is to exclude this transaction from all processing. If you specify YES, also specify the action that CICS BAC is to take if it encounters a batch request utility command for this transaction (see the EXCLUDEACTION parameter).

EXCLUDEABCODE(*abendcode*)
Specifies the numeric abend code CICS BAC is to use if you specify EXCLUDEACTION(ABEND). You can specify an abend code value in the range 1 through 4095.

EXCLUDEACTION({IGNORE|TERMINATE|ABEND})
Specifies the action CICS BAC is to take if you specify EXCLUDE(YES). You can choose one of the following three options:

IGNORE
CICS BAC is to ignore the command as if it was not present and continue processing with the next command.

TERMINATE
CICS BAC is not to process any further commands in the job step, and the batch request utility job step is to terminate with the specified return code (see the EXCLUDERETCODE parameter).

ABEND
CICS BAC is not to process any further commands in the job step, and

the batch request utility job step is to abend with the specified abend code (see the EXCLUDEABCODE parameter).

EXCLUDERETCODE(*returncode***)**
Specifies the numeric return code CICS BAC is to use if you specify EXCLUDEACTION(TERMINATE). You can specify a return code value in the range 1 through 4095.

REPLACE({NO|YES})
Specifies the action the file maintenance utility is to take if a transaction object record already exists for the transaction name you are trying to add.

NO If there is already a transaction object in the CICS region control file for this transaction name, reject the ADD TRANSACTION command.

YES If there is already a transaction object in the CICS region control file for this transaction name, replace it with the information in this ADD TRANSACTION command.

STARTUPOPTS({COLD|WARM|EMER|COLDWARM|COLDWARMEMER|COLDEMER|WARMEMER|USECICS})
Specifies the types of CICS startups during which CICS BAC should set the state of this transaction to its last requested status. If you specify any of the startup types, CICS BAC resets this transaction to its last requested state at the end of that type of CICS region initialization. The options are as follows:

COLD CICS BAC is to restore the last requested state of the transaction during a cold start of the CICS region.

WARM
CICS BAC is to restore the last requested state of the transaction during a warm start of the CICS region.

EMER CICS BAC is to restore the last requested state of the transaction during an emergency restart of the CICS region.

COLDWARM
CICS BAC is to restore the last requested state of the transaction during cold and warm starts of the CICS region.

COLDWARMEMER
CICS BAC is to restore the last requested state of the transaction during cold, warm, and emergency starts of the CICS region.

COLDEMER
CICS BAC is to restore the last requested state of the transaction during cold and emergency starts of the CICS region.

WARMEMER
CICS BAC is to restore the last requested state of the transaction during warm and emergency starts of the CICS region.

USECICS
Depending on the type of startup, CICS BAC is to use the CICS region default startup property (as defined by the region properties record for the CICS region) to determine whether or not it should reset a transaction to its last requested state.

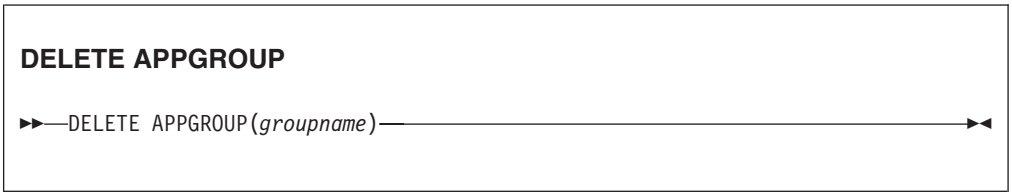
TRANSID(*transid***)**
Specifies the name of the transaction object that you are adding to the CICS region control file.

DELETE APPGROUP

Function

Delete an application group object from the target CICS region control file.

Syntax



Parameters

APPGROUP(*groupname***)**
Specifies the name of an application group object you want to remove from the target CICS region control file.

DELETE APPLIST

Function

Delete an application list object from the target CICS region control file.

Syntax

Parameters

APPLIST(*listname***)**
Specifies the name of an application list object you want to remove from the CICS region control file.

DELETE FILE

Function

Delete a file object from the target CICS region control file.

Syntax

Parameters

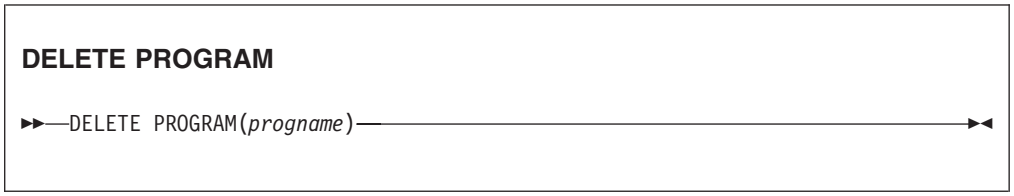
FILE(*filename***)**
Specifies the name of a file object you want to remove from the CICS region control file.

DELETE PROGRAM

Function

Delete a program object from the target CICS region control file.

Syntax



Parameters

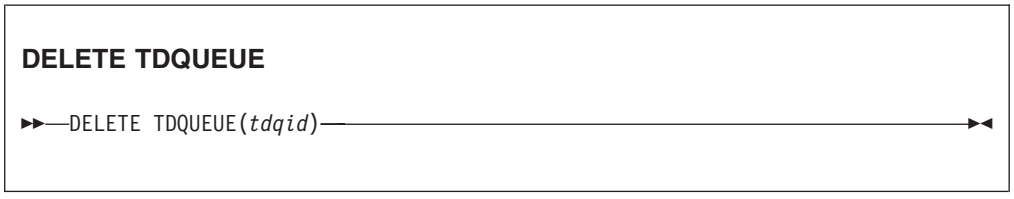
PROGRAM(*progname***)**
Specifies the name of a program object you want to remove from the CICS region control file.

DELETE TDQUEUE

Function

Delete a TD queue object from the target CICS region control file.

Syntax



Parameters

TDQUEUE(*tdqid***)**
Specifies the name of a TD queue object that you want to remove from the CICS region control file.

DELETE TRANSID

Function

Delete a transaction object from the target CICS region control file.

Syntax

DELETE TRANSID

Parameters

TRANSID(*transid***)**
Specifies the name of a transaction object that you want to remove from the CICS region control file.

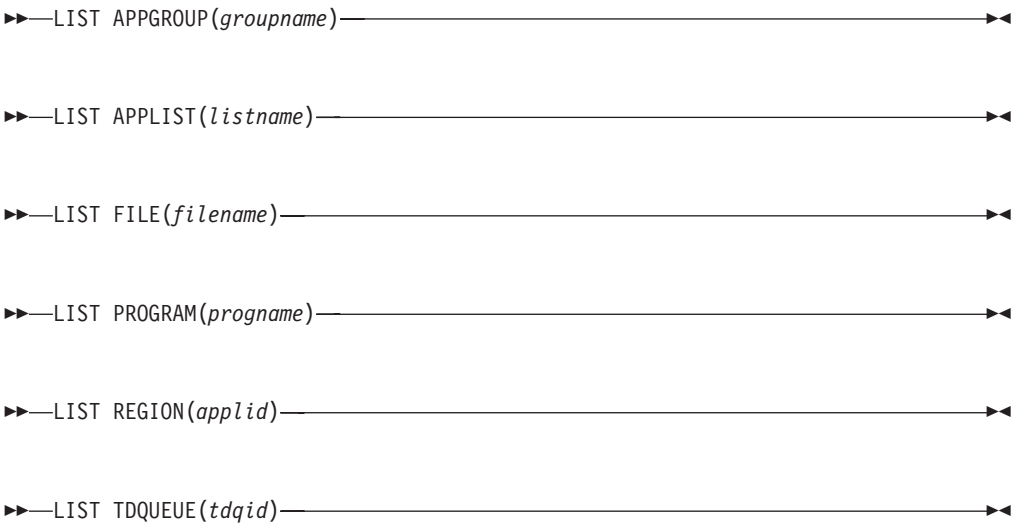
LIST

Function

List one or more object records of a specified object type from the target CICS region control file.

Syntax

LIST commands



```
►►—LIST TRANSID(transid)———————————————————————————————►◄
```

Description

You can use the LIST command to list any, or all, object records of a specified resource type from a target CICS region control file. You can use the generic asterisk (*) character to list more than one object record of the specified resource type or, if you use the generic character on its own, list all the object records of the specified type. For example:

```
LIST REGION(*)        to list the CICS region properties from its control file
LIST FILE(*)          to list all file object reecords from the control file.
LIST APPGROUP(TEST*)  to list all application groups beginning with the string TEST
LIST TDQUEUE(TDQ1)    to list the TD queue named TDQ1.
```

The results of the LIST command are written to CBKPRINT.

Parameters

APPGROUP(*name***| *)**
APPLIST(*listname***| *)**
FILE*filename***| *)**
PROGRAM*progname***| *)**
REGION(*applid***| *)**
TDQUEUE(*tdqid***| *)**
TRANSID(*transid***| *)**

Specifies the name of the object type for which you want to list the specified object record or records. If you specify only the generic asterisk character, the file maintenance utility lists from the target CICS region control file all the object records for the named object type.

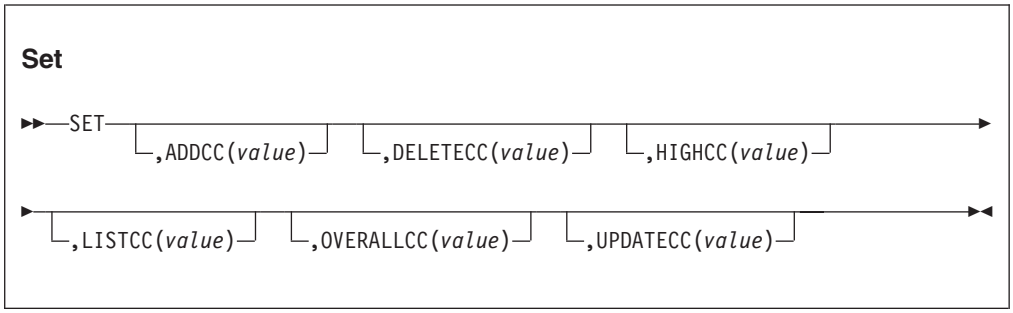
For an explanation of the various field names and data returned by the LIST command, see the corresponding parameters on the ADD and UPDATE commands.

SET

Function

Set the file maintenance utility command processor condition code.

Syntax



Description

The SET command enables you to specify a return code (RC) for individual commands that allows file maintenance utility command processing to continue instead of terminating. You can specify a different condition code value for each of the file maintenance utility commands–ADD, DELETE, and UPDATE. The default permitted condition code for all the commands is 0 (zero) thus, by default any command returning a non-zero return causes termination of the file maintenance utility.

The SET command enables you to raise the maximum allowed condition code to a value that does not cause the utility to terminate. For example, the DELETE command sets return code 8 if the object you specify for deletion does not exist, in which case, the default action is that command processing terminates and no more commands are processed (see CBKxx7302E). However, if you use DELETECC to set a value equal to, or greater than, the object-not-found return code (8 in this case), command processing continues. Each *xxxx*CC command corresponds to the command that it affects.

For example, if there is no FILE object named TEST in the CICS BAC control file and you specify DELETE FILE(TEST), the command return code is 8. By default, command processing terminates and the job step ends. However, if you are prepared to accept this one rejection only and want subsequent commands to be processed, specify SET DELETECC(8) command *before* the DELETE command.

If you subsequently want any commands to revert to default command processing and terminate, for example, when an object is not found, issue a SET DELETECC(0) command. Thus, after each command is processed, and a non-zero return code is issued, a check is made first against the corresponding xxxxCC value. If the return code is less than, or equal to, the xxxxCC value, command processing continues.

Instead of specifying an individual command condition code, you can use the OVERALLCC parameter to specify a 'catchall' condition code. If you have not specified an individual *xxxx*CC value for a particular command, and the command gives a non-zero return code, the command processor compares the return code with any OVERALLCC value that you have specified. If the command return code is less than, or equal to, the OVERALLCC value, command processing continues.

After each command is processed, the return code from the command is compared to the highest return code raised so far. If the latest command return code is higher then the highest return code reported so far, the latest RC is set as the new highest value. You can reset the highest RC value to 0 using the SET HIGHCC(0) command.

Parameters

ADDCC(*value***)**
Specifies the value of the condition code for an ADD command up to which command processing is to continue. If an ADD command raises a return code *above* the specified value, the command processor does not process any further commands and the file maintenance utility job step terminates and displays the highest return code raised during the job step. Note that the highest return code displayed at job step termination might have been set by a HIGHCC parameter.

DELETECC(*value*)
Specifies the value of the condition code for a DELETE command up to which command processing is to continue. If a DELETE command raises a return code *above* the specified value, the command processor does not process any further commands and the file maintenance utility job step terminates and displays the highest return code raised during the job step. Note that the highest return code displayed at job step termination might have been set by a HIGHCC parameter.

HIGHCC(*value*)
Specifies a new highest condition code. This replaces the highest condition code raised up to the point when file maintenance utility processes the HIGHCC parameter.

LISTCC(*value*)
Specifies the value of the condition code for a LIST command up to which command processing is to continue. If a LIST command raises a return code *above* the specified value, the command processor does not process any further commands and the file maintenance utility job step terminates and displays the highest return code raised during the job step. Note that the highest return code displayed at job step termination might have been set by a HIGHCC parameter.

OVERALLCC(*value*)
Specifies, for those commands for which you have not specified an individual xxxxCC parameter, the value of the condition code up to which command processing is to continue. If any command raises a non-zero return code and you have not specified an individual *xxxx*CC parameter for the command, the command processor compares the return code with the OVERALLCC value before it terminates command processing. Thus you can use OVERALLCC as a substitute for an individual command parameter.

UPDATECC(*value*)
Specifies the level of the condition code for an UPDATE command up to which command processing is to continue. If an UPDATE command raises a return code *above* the specified value, the command processor does not process any further commands and the file maintenance utility job step terminates and displays the highest return code raised during the job step. Note that the highest return code displayed at job step termination might have been set by a HIGHCC parameter.

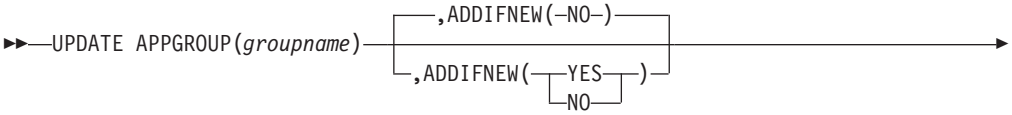
UPDATE APPGROUP

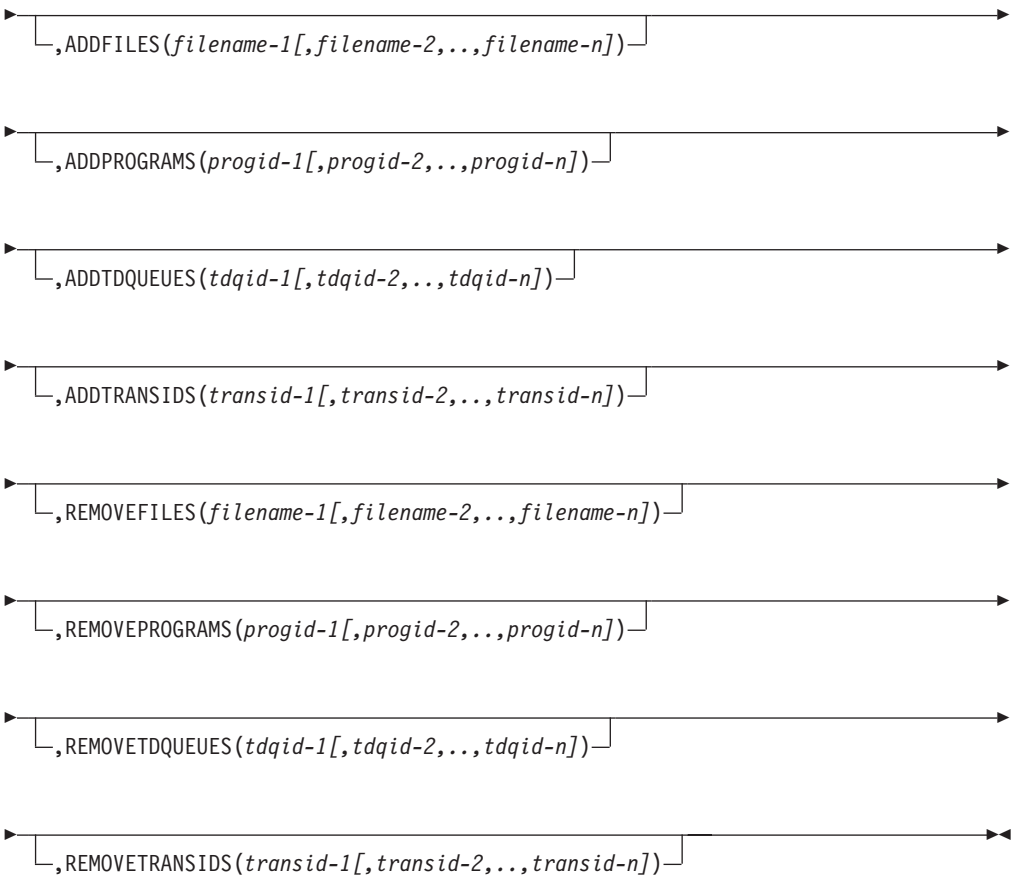
Function

Update an existing application group object record in the target CICS region control file.

Syntax

UPDATE APPGROUP





Description

You can use the UPDATE APPGROUP command to modify an application group object record in a CICS region control file. If a record for the specified application group that you are trying to update does not exist in the region control file, the action taken by the file maintenance utility is determined by the ADDIFNEW parameter.

Parameters

ADDFILES(*filename-1***[,***filename-2***,..,***filename-n***])**
Specifies the name of one or more files to be added to the application group. File names must be valid CICS file names up to a maximum of 8 characters.

ADDIFNEW({NO|YES})
Specifies whether or not a new application group object is to be added if the application group named on the UPDATE APPGROUP command does not exist in the CICS region control file.

ADDPROGRAMS(*progid-1***[,***progid-2***,..,***progid-n***])**
Specifies the name of one or more programs to be added to the application group. Program names must be valid CICS application program names up to a maximum of 8 characters.

ADDTDQUEUES(*tdqid-1***[,***tdqid-2***,..,***tdqid-n***])**
Specifies the ID of one or more TD queues to be added to the application group. TD queue IDs must be valid CICS TD queue IDs up to a maximum of 4 characters.

ADDTRANSIDS(*transid-1***[,***transid-2***,..,***transid-n***])**
Specifies ID of one or more transaction IDs to be added to the application group. Transaction IDs must be valid CICS transaction IDs up to a maximum of 4 characters.

REMOVEFILES(*filename-1***[,***filename-2***,..,***filename-n***])**
Specifies the name of one or more files to be removed from the application group. File names must be valid CICS file names up to a maximum of 8 characters.

REMOVEPROGRAMS(*progid-1***[,***progid-2***,..,***progid-n***])**
Specifies the name of one or more programs to be removed from the application group. Program names must be valid CICS application program names up to a maximum of 8 characters.

REMOVETDQUEUES(*tdqid-1***[,***tdqid-2***,..,***tdqid-n***])**
Specifies the ID of one or more TD queues to be removed from the application group. TD queue IDs must be valid CICS TD queue IDs up to a maximum of 4 characters.

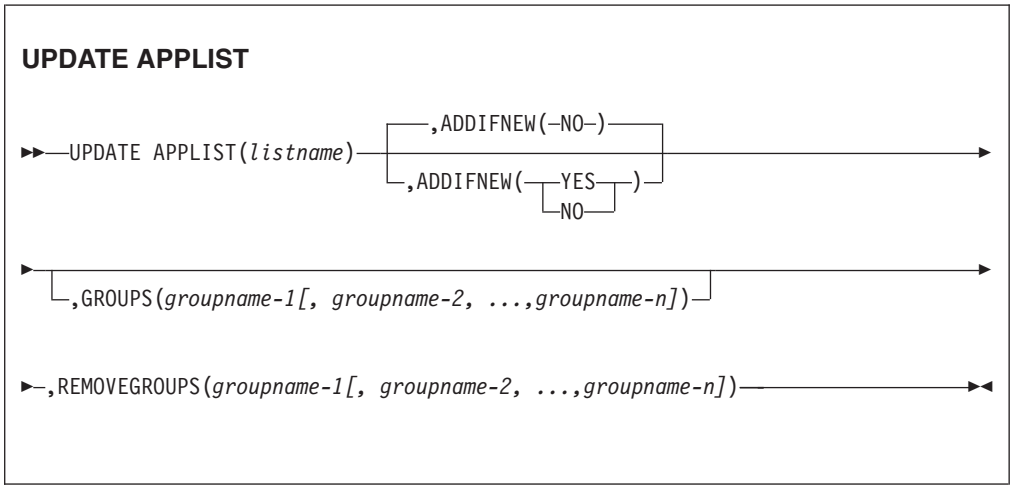
REMOVETRANSIDS(*transid-1***[,***transid-2***,..,***transid-n***])**
Specifies ID of one or more transaction IDs to be removed from the application group. Transaction IDs must be valid CICS transaction IDs up to a maximum of 4 characters.

UPDATE APPLIST

Function

Update an application list object record in the target CICS region control file.

Syntax



Description

You can use the UPDATE APPLIST command to modify an application list object record in a CICS region control file. If a record for the specified application list that

you are trying to update does not exist in the region control file, the action taken by the file maintenance utility is determined by the ADDIFNEW parameter.

Parameters

ADDIFNEW({NO|YES})
Specifies whether or not a new application list object is to be added if the application list named on the UPDATE APPLIST command does not exist in the CICS region control file.

GROUPS(*group-1***[,***group-2***,...group-***n***])**
Specifies names of application groups to be added to the application list. If you specify more than one group, separate the group names with a comma or a blank.

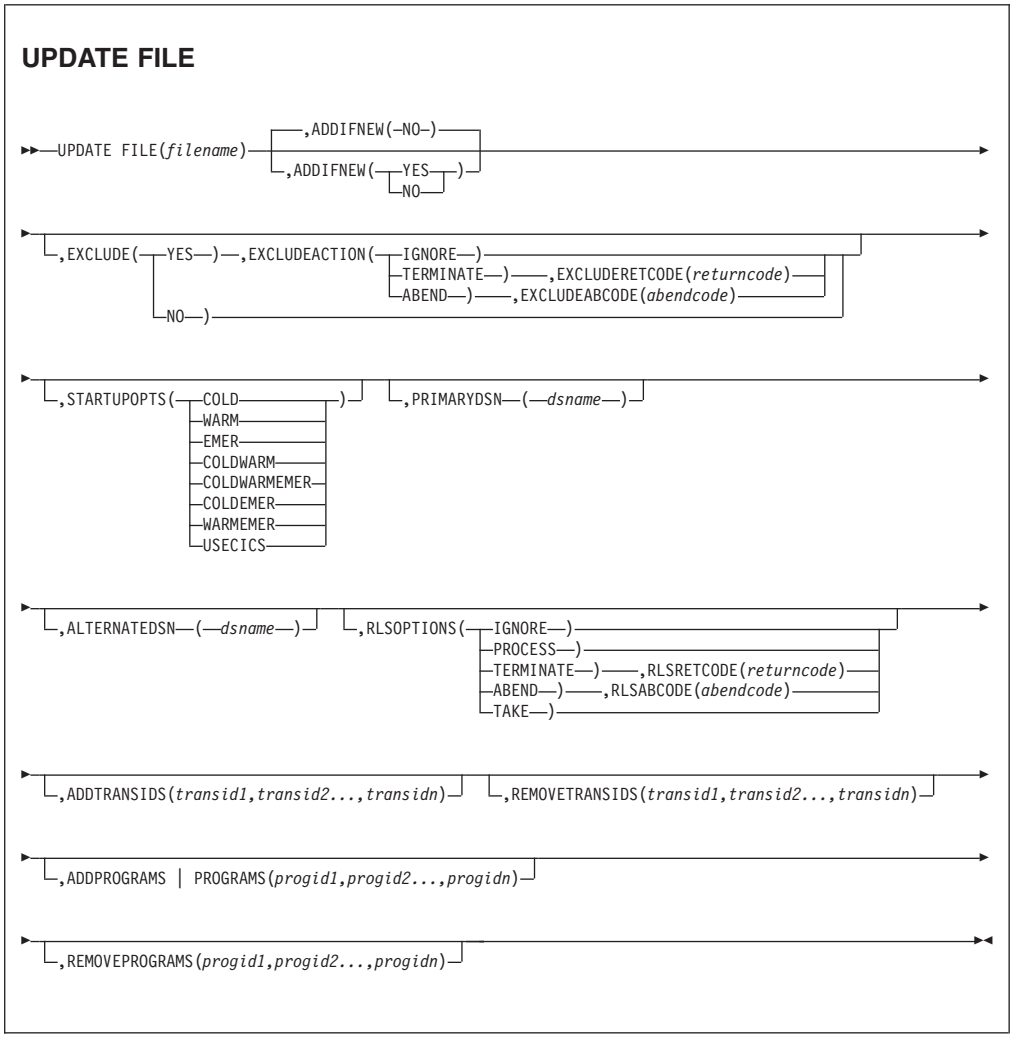
REMOVEGROUPS(*group-1***[,***group-2***,...group-***n***])**
Specifies names of groups to be removed from the application list. If you specify more than one group, separate the group names with a comma or a blank.

UPDATE FILE

Function

Update an existing file object in the CICS region control file.

Syntax



Description

You can use the UPDATE FILE command to modify a file object record in a CICS region control file. If a record for the specified file that you are trying to update does not exist in the region control file, the action taken by the file maintenance utility is determined by the ADDIFNEW parameter.

Parameters

The parameters on the UPDATE FILE command are mostly the same as on those the ADD file command, with the differences described below. For details of the parameters that are common, see the "ADD FILE" on page 77 command.

ADDIFNEW({NO|YES})
Specifies whether or not a new file object is to be added if the file named on the UPDATE FILE command does not exist in the CICS region control file.

ADDPROGRAMS(*progid1,progid2...,progidn*
Specifies one or more programs to be added to the list of programs in the file object record in the CICS region control file.

If you enter more than one program name, separate the names with a comma or a blank.

Note: This is the equivalent of the PROGRAMS parameter on the ADD FILE command.

ADDTRANSIDS*transid-1*,*transid-2...*,*transid-n*
Specifies one or more programs to be added to the list of programs in the file object record in the CICS region control file.

If you enter more than one program name, separate the names with a comma or a blank.

Note: This is the equivalent of the TRANSIDS parameter on the ADD FILE command.

REMOVEPROGRAMS(*progid-1*,*progid-2...*,*progid-n***])**
Specifies one or more programs to be removed from the list of programs in the file object record in the CICS region control file.

If you enter more than one program name, separate the names with a comma or a blank.

For more information, see the PROCESSPROGRAMS parameter on the SET file command on page “SET FILE” on page 51).

REMOVETRANSIDS(*transid1*,*transid2...*,*transidn***])**
Specifies one or more transactions to be removed from the list of transactions in the file object record in the CICS region control file.

If you enter more than one transaction ID, separate the IDs with a comma or a blank.

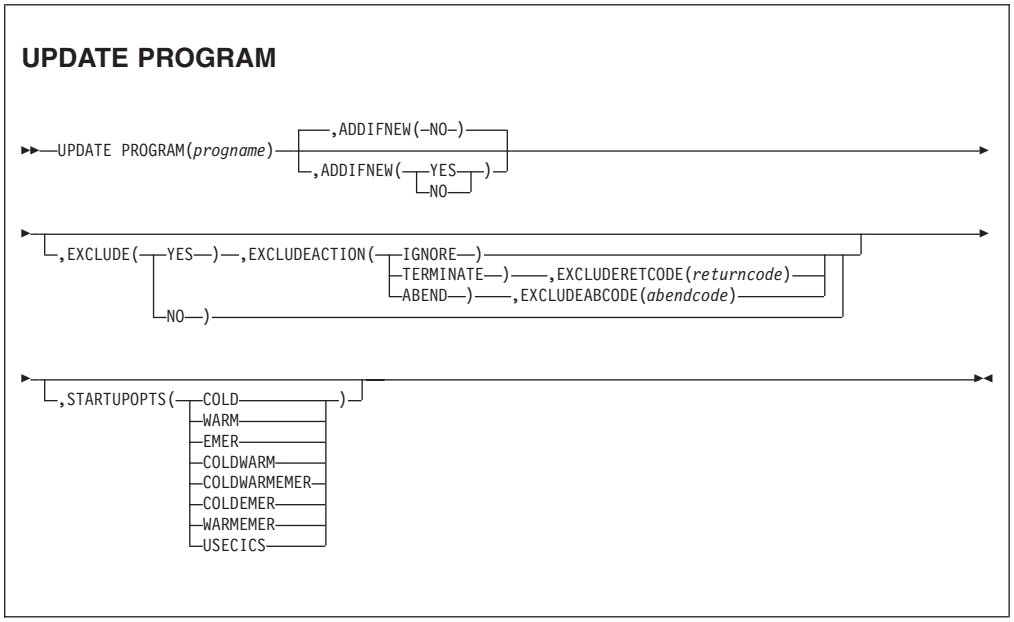
For more information, see the PROCESSTRANSIDS parameter on the SET file command on page “SET FILE” on page 51).

UPDATE PROGRAM

Function

Update an existing program object in the CICS region control file.

Syntax



Description

You can use the UPDATE PROGRAM command to modify a program object record in a CICS region control file. If a record for the specified program that you are trying to update does not exist in the region control file, the action taken by the file maintenance utility is determined by the ADDIFNEW parameter.

Parameters

The parameters on the UPDATE PROGRAM command are the same as on those the ADD PROGRAM command, with the exception of the ADDIFNEW parameter instead of REPLACE. For details of the parameters that are common, see the "ADD PROGRAM" on page 81 command.

ADDIFNEW({NO|YES})
Specifies whether or not a new program object is to be added if the program named on the UPDATE PROGRAM command does not exist in the CICS region control file.

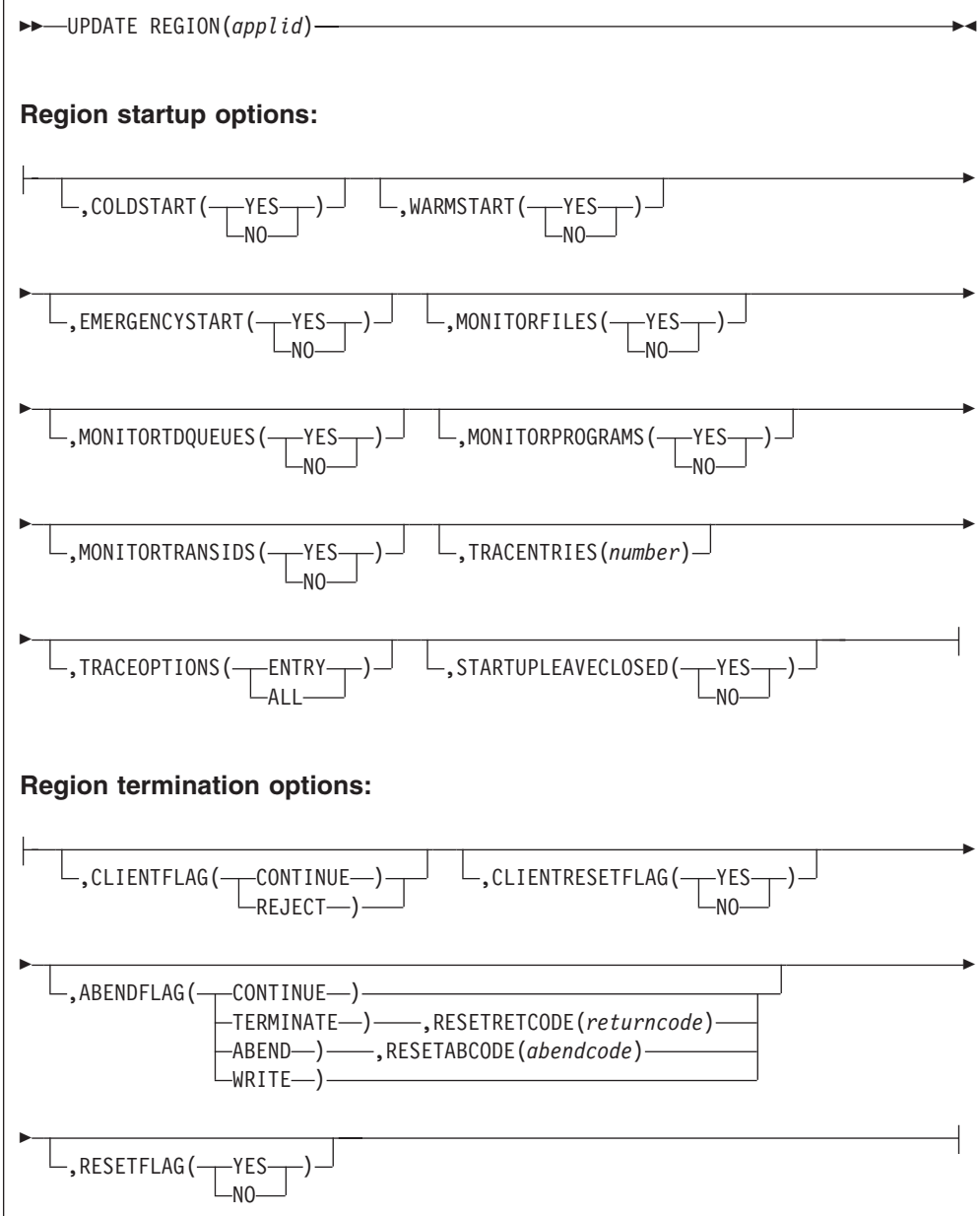
UPDATE REGION

Function

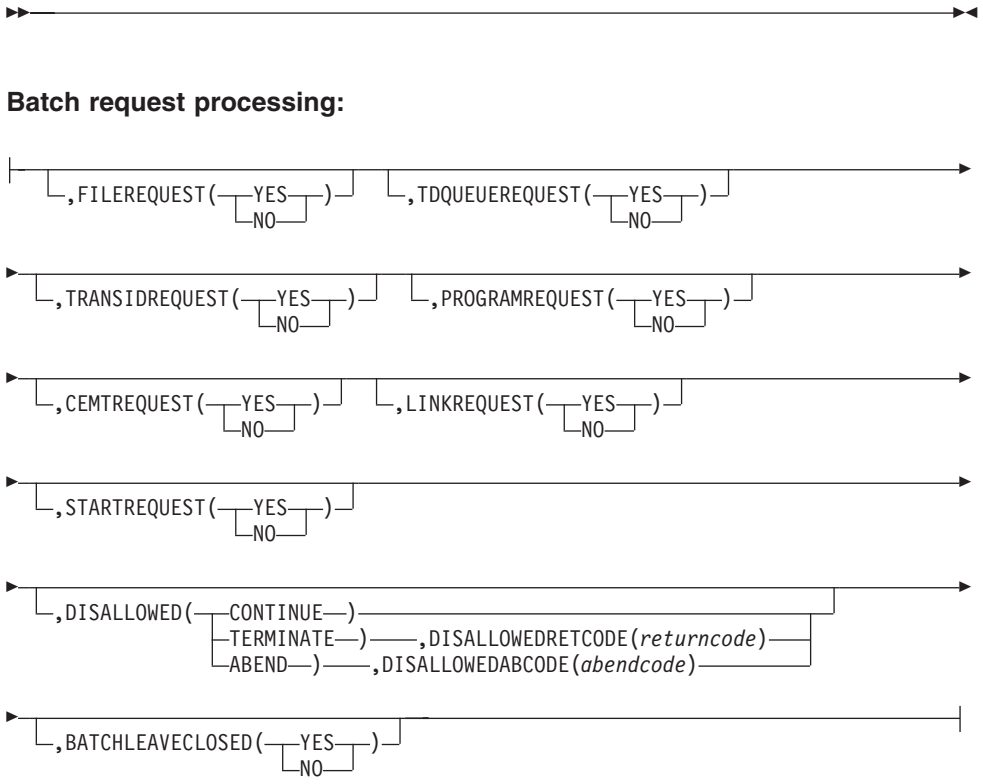
Update an existing CICS region object in the CICS region control file.

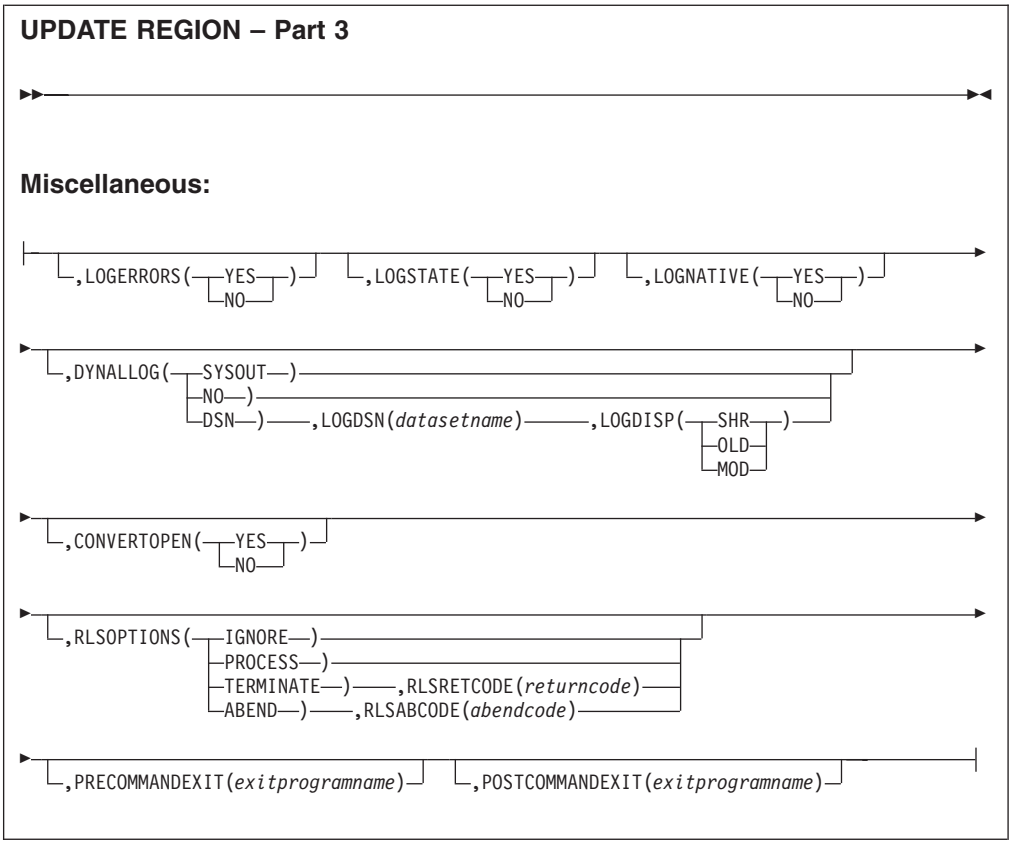
Syntax

UPDATE REGION – Part 1



UPDATE REGION – Part 2





Description

You can use the UPDATE REGION command to modify a CICS region object record in a region control file. If a record for the specified CICS region that you are trying to update does not exist in the region control file, the file maintenance utility rejects the command as an error with error message CBK*xx*7206E.

Note that you cannot specify an ADDIFNEW parameter for the CICS region object record as you can with other resource types. For information about the CICS region control file, see "Defining and initializing a CICS BAC control file" on page 11.

Parameters

ABENDFLAG({CONTINUE|TERMINATE|ABEND|WRITE})
Specifies the action to be taken by the CICS BAC batch request utility if the CICS region is not available and the last time the CICS region was shut down it did not go through normal shutdown processing.

CONTINUE
The CICS BAC batch request utility is to continue processing as if the CICS region had shut down normally.

ABEND
The CICS BAC batch request utility job step is to terminate abnormally with the abend code you specify on the RESETABCODE parameter.

TERMINATE
The CICS BAC batch request utility job step is to terminate with the non-zero return code you specify on the RESETRETCODE parameter.

WRITE
The CICS BAC batch request utility is to issue a write-to-operator-with-reply (WTOR) message to the MVS system console and wait for a reply to determine what further action it should take.

BATCHLEAVECLOSED({YES|NO})
Specifies the action CICS BAC is to take during batch request utility processing if it receives a SET FILE command to set the file open status and enable status to OPEN and ENABLED and the specified file is currently set to CLOSED and ENABLED.

YES Specifies that CICS BAC is to ignore the SET FILE OPENSTATUS and ENABLESTATUS parameters and leave the file status as CLOSED and ENABLED.

NO Specifies that CICS BAC is honor the SET FILE OPENSTATUS and ENABLESTATUS parameters and change the file status to OPEN and ENABLED.

CEMTREQUEST({YES|NO})
Specifies whether CICS BAC is to allow batch request utility RUNCEMT commands to be processed in this CICS region. If you specify NO, the action taken by CICS BAC is determined by the DISALLOWED parameter.

CLIENTFLAG({CONTINUE|REJECT})
Specifies the action CICS BAC is to take for workstation administration client requests (for example, upload and download requests) if the CICS region is not available because of abnormal termination:

CONTINUE
CICS BAC is to continue processing requests as if the region had shut down normally.

REJECT
CICS BAC is to ignore requests and send a message to the workstation administration client stating that the request failed.

CLIENTRESETFLAG({YES|NO})
Specifies whether the region shutdown flag should be reset to normal in addition to the action CICS BAC takes if you specify the CLIENTFLAG parameter.

COLDSTART({YES | NO})
Specifies whether a cold start of the CICS region should include CICS BAC resource processing.

YES If CICS performs a cold start, at the end of CICS region initialization process CICS BAC is to set the state of resources defined in the region control file to their last requested state, unless specified otherwise by an individual resource object definition.

NO If CICS performs a cold start, CICS BAC does not set the state of resources defined in the region control file to their last requested state, unless specified otherwise by an individual resource object definition.

See also the STARTUPOPTS parameter on the definitions of the CICS BAC FILE, PROGRAM, TDQUEUE, and TRANSID resource objects.

CONVERTOPEN({YES|NO})
Specifies the action that CICS BAC is to take for batch request utility SET FILE and SET TDQUEUE commands if a command specifies OPENSTATUS(OPEN) parameter but the ENABLESTATUS parameter is not also specified.

YES Specifies that CICS BAC is to convert SET FILE and SET TDQUEUE OPENSTATUS(OPEN) commands to OPENSTATUS(OPEN),ENABLESTATUS(ENABLED) even if the if the command does not explicitly contain the ENABLESTATUS parameter.

NO Do not convert commands that specify OPENSTATUS without ENABLESTATUS.

CREATERECORD({YES|NO})
Specifies whether CICS BAC is to create a resource object record in the CICS region control file for an undefined resource and for which the batch request utility processes a command. These are undefined resources controlled by the UNDEFINED*resource*(YES) parameters.

YES Specifies that CICS BAC is to create a resource object record in the CICS region control file for an undefined resource that it is allowed to process. Creating a record for an undefined resource enables CICS BAC to set the object to its last known state during CICS BAC region startup processing, and avoids an undefined object condition occurring again for the resource.

See the UNDEFINED and UNDEFINED*resource* parameters for more information.

NO Specifies that CICS BAC is not to create a resource object record in the CICS region control file for an undefined resource, even though you have specified that CICS BAC is allowed to process batch request utility commands for such resources.

DISALLOWED({CONTINUE|TERMINATE|ABEND})
Specifies the action CICS BAC is to take if it receives a disallowed command (that is you have specified NO on one of the *command*REQUEST parameters, such as the FILEREQUEST(NO) parameter, for example). The disallowed actions are:

CONTINUE
CICS BAC is to ignore only the disallowed command and carry on processing the next command from the batch request utility.

TERMINATE
CICS BAC is to terminate the job step with the return code specified on the DISALLOWEDRETCODE parameter.

ABEND
CICS BAC is to terminate the job step abnormally with the abend code specified on the DISALLOWEDABCODE parameter.

Terminate with abend code. CICS BAC will abend the job step with the abend code entered in this text field.

DISALLOWEDABCODE(*number*)
Specifies the abend code CICS BAC is to use when it abends the batch request utility job step because it has received a disallowed command and you have specified DISALLOWED(ABEND).

number
Specifies an abend code in the range 1 through 4095.

DISALLOWEDRETCODE(*number*)
Specifies the job step return code CICS BAC is to use when it terminates the batch request utility job step because it has received a disallowed command and you have specified DISALLOWED(TERMINATE).

number
Specifies a return code in the range 1 through 4095.

EMERGENCYSTART({YES | NO})
Specifies whether an emergency start of the CICS region should include CICS BAC resource processing.

YES If CICS performs an emergency start, at the end of CICS region initialization process CICS BAC is to set the state of resources defined in the region control file to their last requested state, unless specified otherwise by an individual resource object definition.

NO If CICS performs an emergency start, CICS BAC does not set the state of resources defined in the region control file to their last requested state, unless specified otherwise by an individual resource object definition.

See also the STARTUPOPTS parameter on the definitions of the CICS BAC FILE, PROGRAM, TDQUEUE, and TRANSID resource objects.

FILEREQUEST({YES|NO})
Specifies whether CICS BAC is to allow batch request utility SET FILE commands to be processed in this CICS region. If you specify NO, the action taken by CICS BAC is determined by the DISALLOWED parameter.

LINKREQUEST({YES|NO})
Specifies whether CICS BAC is to allow batch request utility LINK commands to be processed in this CICS region. If you specify NO, the action taken by CICS BAC is determined by the DISALLOWED parameter.

DYNALLOG({SYSOUT|DSN|NO})
Specifies whether the CICS BAC audit log, DD name CBKLOG, should be dynamically allocated at CICS BAC startup if it is not present in the CICS region JCL, or in the batch request utility JCL if the request is being serviced directly because the CICS region is not available. If it dynamically allocates the audit log data set, CICS BAC uses CBKLOG as the DD name.

SYSOUT
Specifies that CICS BAC is to dynamically allocate the audit log to the SYSOUT class instead of a data set.

DSN Specifies that CICS BAC is to dynamically allocate the audit log to the data set named on the LOGDSN parameter, with the attributes specified on the LOGDISP parameter.

NO Specifies that CICS BAC is not to dynamically allocate the audit log. If a CBKLOG DD statement is not present in the CICS region JCL, no audit log messages are written, regardless of the value of the LOGERRORS, LOGSTATE, or LOGNATIVE parameters at the time the audit log record needs to be written. This also applies to the batch request utility JCL if the commands are being serviced directly to the CICS region control file because the CICS region is not available.

LOGDISP({SHR|OLD|MOD})
Specifies the disposition of the audit log data set named on the LOGDSN parameter. The dispositions you can specify are the standard MVS DISP parameter values for data sets (SHR, OLD, and MOD).

LOGDSN(*datasetname*)
Specifies the fully-qualified 44-character data set name that CICS BAC is to dynamically allocate as the audit log data set. Dynamic allocation is performed either:

- By CICS BAC if it is running in the CICS region, or
- By the batch request utility if is processing commands directly because the CICS region is not available.

The DD name used to dynamically allocate the audit log is CBKLOG. See "The audit log data set" on page 25 for more information.

LOGERRORS({YES|NO})
Specifies whether you want CICS BAC to write audit log records for any error conditions or failed requests. If you specify YES, log records are written to the audit log if one is currently allocated.

LOGNATIVE({YES|NO})
Specifies whether you want CICS BAC batch request utility to write audit log records for any error conditions or failed requests when it is updating the CICS region control file directly because the CICS region is not available. If you specify YES, log records are written to the audit log if one is currently allocated to the batch request utility.

LOGSTATE({YES|NO})
Specifies whether you want CICS BAC to write audit log records for any changes to the state of resources by other than CICS BAC commands. If you specify YES, log records are written to the audit log if one is currently allocated.

MONITORFILES({YES|NO})
Specifies whether you want the CICS BAC state monitor to track file status changes for the region. Note that all state changes resulting from a CICS BAC command are recorded, regardless of the value of this parameter.

YES Any change in the state of a file occurring while CICS BAC is active in the region is recorded in the relevant file object record in the CICS control file record, if the file is defined.

NO Any change in the state of a file is not recorded unless the state change originated from a CICS BAC command from the batch request utility or from the CICS BAC request server callable API.

MONITORPROGRAMS({YES|NO})
Specifies whether you want the CICS BAC state monitor to track program status changes for the region. Note that all state changes resulting from a CICS BAC command are recorded, regardless of the value of this parameter.

YES Any change in the state of a program occurring while CICS BAC is active in the region is recorded in the relevant program object record in the CICS control file record, if the program is defined.

NO Any change in the state of a program is not recorded unless the state change originated from a CICS BAC command from the batch request utility or from the CICS BAC request server callable API.

MONITORTDQUEUES({YES|NO})
Specifies whether you want the CICS BAC state monitor to track TD queue status changes for the region. Note that all state changes resulting from a CICS BAC command are recorded, regardless of the value of this parameter.

YES Any change in the state of a TD queue that occurs while CICS BAC is active in the region is recorded in the relevant TD queue object record in the CICS control file record, if the TD queue is defined.

NO Any change in the state of a TD queue is not recorded unless the state

change originated from a CICS BAC command from the batch request utility or from the CICS BAC request server callable API.

MONITORTRANSIDS({YES|NO})
Specifies whether you want the CICS BAC state monitor to track transaction ID status changes for the region. Note that all state changes resulting from a CICS BAC command are recorded, regardless of the value of this parameter.

YES Any change in the state of a transaction ID that occurs while CICS BAC is active in the region is recorded in the relevant transaction ID object record in the CICS control file record, if the transaction ID is defined.

NO Any change in the state of a transaction ID is not recorded unless the state change originated from a CICS BAC command from the batch request utility or from the CICS BAC request server callable API.

POSTCOMMANDEXIT(*exitprogname*)
Specifies the name of your exit program that is to be invoked immediately after the CICS BAC request server has processed a batch request utility command in the CICS region. Note that commands must be processed in the CICS region for this post-command exit to be invoked. Also, commands issued through the CICS BAC request server callable API can also drive the exit.

PRECOMMANDEXIT(*exitprogname*)
Specifies the name of your exit program that is to be invoked immediately before the CICS BAC request server processes a batch request utility command in the CICS region. Note that commands must be processed in the CICS region for this pre-command exit to be invoked. Also, commands issued through the CICS BAC request server callable API can also drive the exit.

PROGRAMREQUEST({YES|NO})
Specifies whether CICS BAC is to allow batch request utility SET PROGRAM commands to be processed in this CICS region. If you specify NO, the action taken by CICS BAC is determined by the DISALLOWED parameter.

REGION(*applid*)
Specifies the name of the CICS region object record that you are updating.

RESETABCODE(*number*)
Specifies the abend code to be used by the batch request utility if:

- The CICS region is not available and the last time the CICS region was shut down it did not go through normal shutdown processing and
- You specify ABENDFLAG(ABEND) in the CICS region properties record.

number
Specifies an abend code in the range 1 through 4095.

RESETFLAG({YES|NO})
Specifies whether the CICS region shutdown flag is to be reset to normal in addition to the action CICS BAC takes if you specify the ABENDFLAG parameter.

RESETRETCODE(*number*)
Specifies the return code to be used by the batch request utility if:

- The CICS region is not available and the last time the CICS region was shut down it did not go through normal shutdown processing and
- You specify ABENDFLAG(TERMINATE).

number
Specifies a return code in the range 1 through 4095.

RLSOPTIONS({PROCESS|IGNORE|TERMINATE|ABEND})
Specifies what action CICS BAC is to take when the CICS BAC request server receives a command to open or close a file defined to CICS with RLSACCESS(YES).

PROCESS
Specifies that CICS BAC is to process the command. If the command is to close the file, CICS BAC issues the appropriate commands to quiesce the data set associated with the file (if the file is not currently quiesced) before closing the file. If the command is to open the file, CICS BAC issues the appropriate commands to unquiesce the data set associated with the file (if it is currently quiesced) before opening the file.

IGNORE
Specifies that CICS BAC is to ignore the command. However, programs and transaction IDs associated with the file are processed as if the SET FILE command was processed.

TERMINATE
Specifies that CICS BAC is to terminate the job step with the return code specified on the RLSRETCODE parameter.

ABEND
Specifies that CICS BAC is to abnormally terminate the job step with the abend code specified on the RLSABCODE parameter

RLSRETCODE(*number*)
Specifies the numeric return code CICS BAC is to use if you specify RLSOPTIONS(TERMINATE). You can specify a return code value in the range 1 through 4095.

RLSABCODE(*number*)
Specifies the numeric abend code CICS BAC is to use if you specify RLSOPTIONS(ABEND). You can specify a return code value in the range 1 through 4095.

STARTREQUEST({YES|NO})
Specifies whether CICS BAC is to allow batch request utility START commands to be processed in this CICS region. If you specify NO, the action taken by CICS BAC is determined by the DISALLOWED parameter.

STARTUPLEAVECLOSED({YES|NO})
Specifies how you want CICS BAC to handle status changes for files during CICS region startup. To reduce region initialization time, CICS allows you to define a file with the CLOSED ENABLED attributes. This means that the file is not allocated and opened until the first time it is referenced by an application program. This avoids the overhead of opening files during CICS initialization, and reduces CICS region initialization time.

YES CICS BAC is to allow a file to remain in the CLOSED ENABLED state during startup processing even if, during CICS region initialization, it finds that the last requested status for the file is OPEN ENABLED.

NO CICS BAC is to override the CLOSED ENABLED attributes if the last requested state in the CICS BAC region control file is OPEN ENABLED and open the file.

SYSCREATE({YES|NO})
Specifies whether CICS BAC is to create a resource object record in the CICS region control file for an undefined resource and for which the CICS BAC state

change monitor has detected a state change. This parameter applies to undefined resources controlled by the UNDEFINED*resource*(YES) parameters.

YES Specifies that CICS BAC is to create a resource object record in the CICS region control file for an undefined resource for which the state change monitor has detected a state change. Creating a record for an undefined resource enables CICS BAC to set correctly the state of an object during CICS BAC region startup processing, and avoids an undefined object condition occurring again for the resource.

See the UNDEFINED and UNDEFINED*resource* parameters for more information.

NO Specifies that CICS BAC is not to create a resource object record in the CICS region control file for an undefined resource.

TDQUEUEREQUEST({YES|NO})
Specifies whether CICS BAC is to allow batch request utility SET TDQUEUE commands to be processed in this CICS region. If you specify NO, the action taken by CICS BAC is determined by the DISALLOWED parameter.

TRACEENTRIES(0|*number*)
Specifies the number of trace table entries CICS BAC is to allocate for diagnostic tracing. A single entry occupies 128 bytes in CICS private storage above 16MB.

0 Zero turns off CICS BAC internal trace.

number
Number can be a value in the range 1 through 9999.

TRACEOPTIONS({ENTRY|ALL})
Specifies the level of CICS BAC internal tracing activity you want in the CICS region. CICS BAC has its own trace facility for diagnostic purposes and the following options specify what you want CICS BAC to trace:

ENTRY
Trace CICS BAC module entry and exit points only.

ALL Trace all CICS BAC trace points.

TRANSIDREQUEST({YES|NO})
Specifies whether CICS BAC is to allow batch request utility SET TRANSID commands to be processed in this CICS region. If you specify NO, the action taken by CICS BAC is determined by the DISALLOWED parameter.

UNDEFINED({CONTINUE|TERMINATE|ABEND})
Specifies the action you want CICS BAC to take if an undefined object condition occurs and you have specified that batch request utility commands for such resources are not to be processed (see the UNDEFINED*resource* parameters).

In some situations, a batch request utility job step can contain a command to change the state of a resource, even though the resource is not yet defined to CICS BAC in the CICS region control file, causing an undefined object condition. Similarly, if the CICS BAC state change monitor is active and tracking resource state changes in the CICS region, it could detect a state change request for an undefined resource.

If you have specified NO on an UNDEFINED*resource* parameter, specify the action CICS BAC is to take as follows:

CONTINUE
Specifies that CICS BAC is to ignore the batch request utility command for the undefined resource and continue processing with the next command.

TERMINATE
Specifies that CICS BAC is to terminate the batch request utility job step with the return code you specify on the UNDEFINEDRETCODE parameter.

ABEND
Specifies that CICS BAC is to terminate abnormally the batch request utility job step with the abend code you specify on the UNDEFINEDABCODE parameter.

UNDEFINEDABCODE({YES|NO})
Specifies the abend code CICS BAC is to use when it abends the batch request utility job step because it has received a command for an undefined object and you have specified UNDEFINED(ABEND).

number
Specifies an abend code in the range 1 through 4095.

UNDEFINEDFILE({YES|NO})
UNDEFINEDPROGRAM({YES|NO})
UNDEFINEDTDQUEUE({YES|NO})
UNDEFINEDTRANSID({YES|NO})
Specifies the action you want CICS BAC to take if the batch request utility receives a command for a file, program, TD queue, or transaction ID resource that is not defined in the CICS region control file (see also the UNDEFINED parameter).

YES Specifies that CICS BAC batch request utility can process commands for file, program, TD queue, or transaction ID resources if they are not defined in the CICS region control file.

NO Specifies that CICS BAC batch request utility is not to process commands for resources if they are not defined in the CICS region control file. The action taken by the batch request utility in this case is determined by what you specify on the UNDEFINED parameter.

UNDEFINEDRETCODE({YES|NO})
Specifies the return code CICS BAC is to use when it terminates the batch request utility job step because it has received a command for an undefined object and you have specified UNDEFINED(TERMINATE).

number
Specifies a return code in the range 1 through 4095.

WARMSTART({YES | NO})
Specifies whether a warm start of the CICS region should include CICS BAC resource processing.

YES If CICS performs a warm start, at the end of CICS region initialization process CICS BAC is to set the state of resources defined in the region control file to their last requested state, unless specified otherwise by an individual resource object definition.

NO If CICS performs a warm start, CICS BAC does not set the state of resources defined in the region control file to their last requested state, unless specified otherwise by an individual resource object definition.

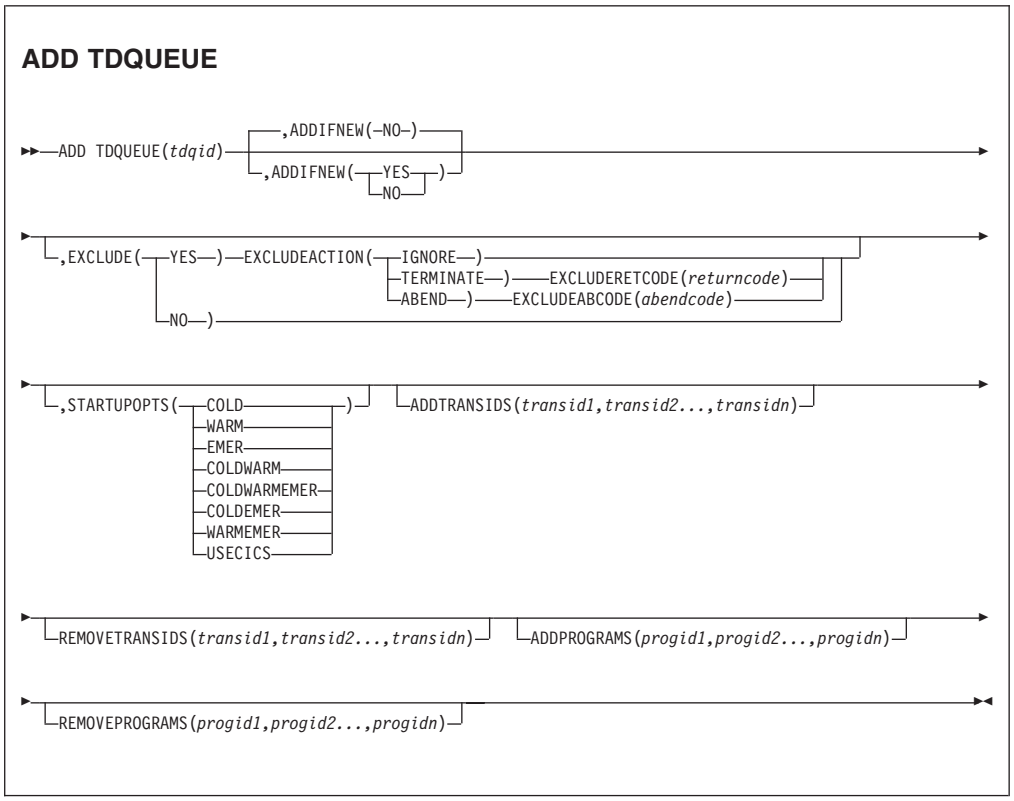
See also the STARTUPOPTS parameter on the definitions of the CICS BAC FILE, PROGRAM, TDQUEUE, and TRANSID resource objects.

UPDATE TDQUEUE

Function

Update an existing TD queue object in the CICS region control file.

Syntax



Description

You can use the UPDATE TDQUEUE command to modify a TD queue object record in a CICS region control file. If a record for the specified TD queue that you are trying to update does not exist in the region control file, the action taken by the file maintenance utility is determined by the ADDIFNEW parameter.

Parameters

The parameters on the UPDATE TDQUEUE command are mostly the same as on those the ADD TDQUEUE command, with the differences described below. For details of the parameters that are common, see the “ADD TDQUEUE” on page 84 command.

ADDIFNEW({NO|YES})
Specifies whether or not a new TD queue object is to be added if the TD queue named on the UPDATE TDQUEUE command does not exist in the CICS region control file.

ADDPROGRAMS(*progid1,progid2...,progidn*

Specifies one or more programs to be added to the list of programs in the TD queue object record in the CICS region control file.

If you enter more than one program name, separate the names with a comma or a blank.

Note: This is the equivalent of the PROGRAMS parameter on the ADD TDQUEUE command.

ADDTRANSIDS*transid-1,transid-2...,transid-n*

Specifies one or more transactions to be added to the list of transactions in the TD queue object record in the CICS region control file.

If you enter more than one transaction ID, separate the IDs with a comma or a blank.

Note: This is the equivalent of the TRANSIDS parameter on the ADD TDQUEUE command.

REMOVEPROGRAMS(*progid-1,progid-2...,progid-n***])**

Specifies one or more programs to be removed from the list of programs in the file object record in the CICS region control file.

If you enter more than one program name, separate the names with a comma or a blank.

For more information, see the PROCESSPROGRAMS parameter on the SET file command on page “SET FILE” on page 51).

REMOVETRANSIDS(*transid1,transid2...,transidn***])**

Specifies one or more transactions to be removed from the list of transactions in the file object record in the CICS region control file.

If you enter more than one transaction ID, separate the IDs with a comma or a blank.

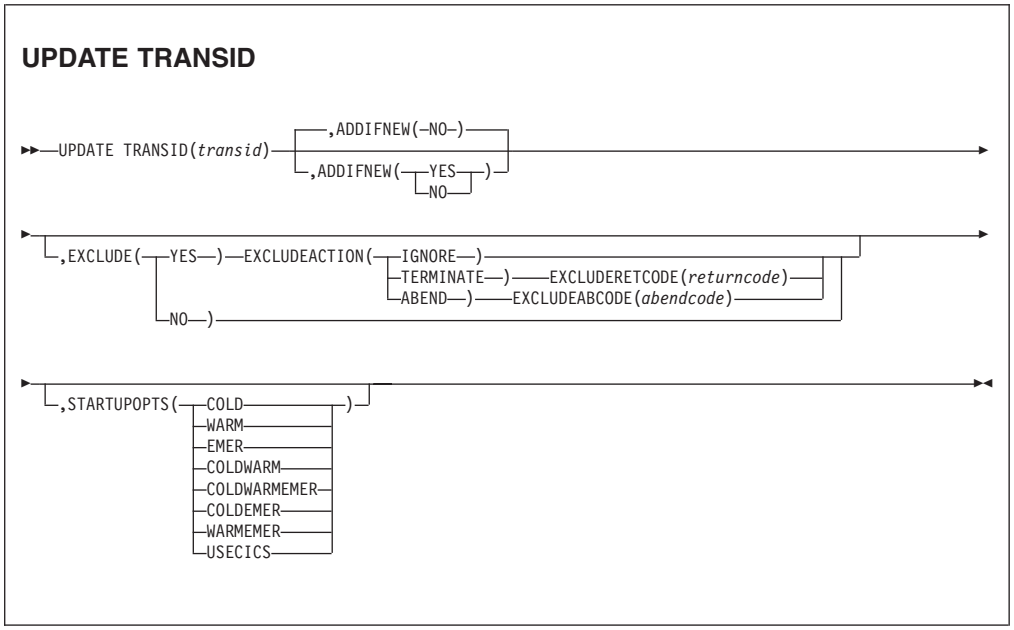
For more information, see the PROCESSTRANSIDS parameter on the SET file command on page “SET FILE” on page 51).

UPDATE TRANSID

Function

Update an existing transaction object in the CICS region control file.

Syntax



Description

You can use the UPDATE TRANSID command to modify a transaction object record in a CICS region control file. If a record for the specified transaction that you are trying to update does not exist in the region control file, the action taken by the file maintenance utility is determined by the ADDIFNEW parameter.

Parameters

The parameters on the UPDATE TRANSID command are the same as on those the ADD TRANSID command, with the exception of the ADDIFNEW parameter instead of REPLACE. For details of the parameters that are common, see the "ADD PROGRAM" on page 81 command.

ADDIFNEW({NO|YES})
Specifies whether or not a new transaction object is to be added if the transaction named on the UPDATE TRANSID command does not exist in the CICS region control file.

Chapter 8. Additional facilities

This chapter describes the following additional facilities that are provided by CICS BAC:

- "CICS BAC request server callable API"
- "CICS BAC request server user exits"
- "CICS group and alias function" on page 117
- "CICS BAC support for record-level sharing (RLS)" on page 119

CICS BAC request server callable API

You can invoke the CICS BAC request server from a CICS application program using the EXEC CICS LINK command. You can issue any of the SET *resource object* commands, such as SET FILE. To call the CICS BAC CICS request server, link to program CBKCLINK and pass it the appropriate COMMAREA, as follows:

```
EXEC CICS LINK PROGRAM('CBKCLINK') COMMAREA(commarea) LENGTH(calength)
```

In the above example, *commarea* is a storage area containing the CICS BAC request, and *calength* is a halfword value containing the length of *commarea*. The storage area layout for the COMMAREA is mapped by SCBKSAMP member CBKAPI for assembler language programs. Comments in the SCBKSAMP members explain the various fields and their use.

When you call the CICS BAC CICS request server from an application program, you must obey the following rules:

- CICS BAC must be active in the CICS region.
- You can only issue one command at a time.
- Set to null values all reserved, unused, and filler storage areas in the COMMAREA.
- Set to null values all variables that are not used or are not relevant to the command being issued.
- Ensure you follow the CICS BAC batch request utility rules for accepted combinations of parameters.
- Ensure you follow the CICS BAC batch request utility rules for required and minimum parameters for a command.
- Each invocation is independent, even within the same CICS unit of work, and your application program cannot derive any value from a previous invocation.

CICS BAC provides a sample program for invoking the CICS BAC request server to close and disable a file and set it to read-only status in member CBKCAPI of the SCBKSAMP data set.

CICS BAC request server user exits

CICS BAC provides two user exit points for which you can provide your own user exit programs. These exit programs are driven during the processing of CICS BAC batch request utility commands in the CICS region:

- The first is a CICS pre-command exit program and this is called immediately before an EXEC CICS SET command is issued.

- The second is a CICS post-command exit program and this is called immediately after an EXEC CICS SET command is issued (except the CICS SET DSNAME command when processing an RLS data set), unless a severe error prevents the exit program from being called.

You specify the names of the exit programs in the CICS region properties record using the CICS BAC workstation administration client or the CICS BAC batch file maintenance utility. If you specify a program name for one or both of these exit points, the relevant exit point is enabled and active after CICS BAC has loaded the CICS region properties record. You specify the exit program names on the Miscellaneous(2) tab of the CICS region properties panel. The exits are called only when a CICS BAC command is processed by the CICS request server running in the region. Batch request commands that are serviced directly by the batch request utility itself do not result in the user exits being called. CICS BAC commands from an application program calling the CICS BAC request server through the callable API also drive the user exit programs. The user exit programs are not called during CICS BAC startup processing in the CICS region.

You can use the pre-command exit to suppress the execution of a command by setting the appropriate return code before returning to CICS BAC. If the command is not suppressed, it will be executed using the parameter values described by the COMMAREA passed to the exit program. You cannot alter the command by changing values in the COMMAREA. All changes to the COMMAREA to fields other than the return code fields and message text field are ignored. You can also perform in your exit program any other processing permitted by CICS in an application program.

You can use the post-command exit only to perform some additional processing of your own following the completion of a command. If your user exit program makes any changes to the COMMAREA passed to it, they are ignored.

CICS BAC calls user exit programs before and after the state of a CICS resource (file, transient data queue, program, or transaction ID) is changed. This can occur as a result of a CICS BAC command that results in the resource state being changed directly (as in a CICS BAC SET PROGRAM command), or indirectly, such as a transaction ID state being changed as a result of a request to change the open state or enable state of a file with which the transaction ID is associated. Resource state changes that result from a CICS BAC SET LIST or SET GROUP command also result in the user exits being called. Note that the exits are not called if the resource named on a SET command is already at the requested state. For example, if a command specifies that a file is to be closed, but the file is already closed, the state of the file is not changed and therefore the pre-command and post command user exits are not called.

The user exit programs are not called for the following events:

- Processing of a CICS BAC SET LIST command. However, the exits are called for each of the resources whose states are being changed as a result of the SET LIST command.
- Processing of a CICS BAC SET GROUP command. However, the exits are called for each of the resources whose states are being changed as a result of the SET GROUP command.

If specified, the CICS BAC user exit programs are driven by a standard EXEC CICS LINK command. A data area describing:

- The EXEC CICS SET command that is about to be processed for the pre-command exit, or
- The results of the command that has just been processed for the post-command exit,

is passed to the user exit program as a COMMAREA. To help you develop your user exit programs, the SCBKSAMP data set provides the following:

- A DSECT describing the layout of the COMMAREA in member CBKXPI
- A sample user exit program in member CBKUEXIT.

CICS group and alias function

The CICS group and alias function enables you to do the following:

- Specify an alias for a single CICS region, allowing you to route CICS BAC batch request commands to a different named region from that specified on the CICS(*applid*) parameter. You might want to do this, for example, where a region name has been changed and you don't want to change CBKIN commands.
- Specify one name only and have the relevant CICS BAC commands routed to more than one CICS region in a group.

These two facilities operate through the CICS parameter in the following way. In its basic and simplest form, the CICS(*applid*) specifies the VTAM applid of a target CICS region. However, you can also use the parameter to reference a PDS member of a data set that contains one or more target CICS region VTAM applids. Whether CICS BAC uses the CICS parameter value as a real region applid, or as the member of a data set, depends on the presence or absence in a data set of a PDS member name that matches the name of the specified CICS applid:

- If a member is found, all the applids in that member become target CICS regions for the appropriate CICS BAC command.
- If CICS BAC cannot find a matching member name, the parameter value itself is taken as the target CICS region applid.
- If you don't include a data set in the batch request JCL, or you don't specify the CICSGROUPDSN parameter, the parameter value itself is taken as the target CICS region applid.

You specify a data set in one of two ways:

- You can specify the PDS on the CBKGROUP DD statement in the batch request utility job step JCL. If CICS BAC detects a DD statement, it uses the data set associated with that DD statement.
- You can specify a batch request utility runtime parameter called CICSGROUPDSN to provide the group data set name. If you omit the CBKGROUP DD statement from the JCL, CICS BAC looks for the CICSGROUPDSN parameter using the same search method as for other batch request utility runtime parameters. The format of the CICSGROUPDSN parameter is as follows:

  `CICSGROUPDSN(datasetname)`

A CICS group data set is a PDS defined with the following attributes:

- Record length (LRECL)=80
- Record format (FB) for fixed-length blocked, respectively.
- Block size (BLKSIZE) as an exact multiple of 80.
- Data set organization (DSORG)=PO

- Directory blocks must be a minimum of 1.

You can define in a CICS group member one or more CICS applids, with only one applid in each record. Begin each applid in column one, and pad the names on the right with spaces if they are less than eight bytes. Anything you enter in columns 9 to 72 is ignored. A record containing only spaces, or with an asterisk in column one, is treated as a comment card and is ignored.

You can use a PDS member with only one applid defined to provide an alias for a single CICS region, or to redirect a command from one CICS region to another. For example, if you have hundreds of batch request utility job steps that contain the applid CICSA, and new naming standards have forced you to change the applid from CICSA to CICS1, you make the change by simply by adding a CICS group member named CICSA. In that member, you specify the new CICS applid of CICS1. Thus, whenever a batch request utility job step targets CICSA, CICS BAC locates member CICSA and redirects commands to region CICS1.

CICS BAC performs only one level of name resolution. That is, if a CICS group member exists, the applids in the member are the final targets. No check is performed on the applids in the member to see if they in turn point to another CICS group member.

When a batch request utility command specifies a CICS group member as its target, the utility issues the command in sequence to each of the CICS applids in the group as if you had specified separate identical commands for each region.

Following are some examples of commands issued based on the use of the CICS group member function. In these examples, the CICS group data set member named CICS1 contains applid records for CICSA, CICSB, and CICSC, in that order:

Example 1: These actual commands from CBKIN:

```
SET FILE(FILE1),OPENSTATUS(CLOSE),CICS(CICS1)
SET FILE(FILE2),OPENSTATUS(CLOSE),CICS(CICS1)
```

result in the following commands being issued:

```
SET FILE(FILE1),OPENSTATUS(CLOSE),CICS(CICSA)
SET FILE(FILE1),OPENSTATUS(CLOSE),CICS(CICSB)
SET FILE(FILE1),OPENSTATUS(CLOSE),CICS(CICSC)
SET FILE(FILE2),OPENSTATUS(CLOSE),CICS(CICSA)
SET FILE(FILE2),OPENSTATUS(CLOSE),CICS(CICSB)
SET FILE(FILE2),OPENSTATUS(CLOSE),CICS(CICSC)
```

The same six generated commands shown in the above example are also generated if the your batch request utility job step contains the following actual commands:

```
DEFAULT CICS(CICS1)
SET FILE(FILE1),OPENSTATUS(CLOSE)
SET FILE(FILE2),OPENSTATUS(CLOSE)
```

Example 2`: These actual commands from CBKIN:

```
DEFAULT CICS(CICS1)
SET FILE(FILE1),OPENSTATUS(CLOSE)
SET FILE(FILE2),OPENSTATUS(CLOSE),CICS(CICSTEST)
SET FILE(FILE3),OPENSTATUS(CLOSE)
```

result in the following commands being issued:

```
SET FILE(FILE1),OPENSTATUS(CLOSE),CICS(CICSA)
SET FILE(FILE1),OPENSTATUS(CLOSE),CICS(CICSB)
SET FILE(FILE1),OPENSTATUS(CLOSE),CICS(CICSC)
SET FILE(FILE2),OPENSTATUS(CLOSE),CICS(CICSTEST)
SET FILE(FILE1),OPENSTATUS(CLOSE),CICS(CICSA)
SET FILE(FILE2),OPENSTATUS(CLOSE),CICS(CICSB)
SET FILE(FILE3),OPENSTATUS(CLOSE),CICS(CICSC)
```

In this example, CICSTEST is a real applid that does not have a CICS group data set member of this name and is routed normally.

CICS BAC support for record-level sharing (RLS)

CICS BAC provides support for CICS files defined to be used in record-level sharing (RLS) mode, and when a batch request utility command that specifies a file that is currently defined as RLSACCESS(YES) in an active CICS region (in other words, the command will be serviced by the target CICS region). However, this support is not available if CICS BAC is not available in the target CICS region, or the region is unavailable and the command is therefore processed natively by the batch request utility, because CICS BAC cannot determine if the file is defined with RLSACCESS(YES).

CICS BAC RLS processing for a file depends on the CICS BAC RLS access options specified in the file object record in the CICS region control file:

- If the RLS option is IGNORE, CICS BAC does not perform any of the requested state changes for the file, but it does process all transaction IDs and programs associated with the file, just as if the file was processed as requested.
- If the RLS option is ABEND or TERMINATE, the batch request utility job step immediately abends or terminates with the requested abend or return code. In this case, transaction IDs and programs associated with the file are not processed.
- If the RLS option for the file is PROCESS, the CICS BAC RLS processing is performed as described in the following section.
- If the RLS option is TAKE, CICS BAC takes the RLS option specified in the CICS region properties record in the region control file.

Note: For details of these RLS options, see the “ADD FILE” on page 77 and “UPDATE FILE” on page 97 commands on pages “ADD FILE” on page 77 and “UPDATE FILE” on page 97 respectively.

CICS BAC request server RLS processing

When all the circumstances are such that the CICS BAC request server is enabled to perform its RLS processing, its actions are as follows:

- CICS BAC request server determines the name of the data set associated with the file
- CICS BAC request server determines the current quiesce state for the data set and:
  - If the data set is currently quiesced, and the request for the file is to open it, CICS BAC request server issues an EXEC CICS SET DSNAME(*dataset*) UNQUIESCE command to unquiesce the data set.
  - If the data set is currently unquiesced, and the request for the file is to close it, CICS BAC request server issues an EXEC CICS SET DSNAME(*dataset*)

QUIESCE command to quiesce the data set. If the CICS BAC command specified IMMQUIESCE(YES), an immediate quiesce option is also specified on the EXEC CICS SET command.

  - If neither of these is true, no quiesce or unquiesce command is issued.
  - If a quiesce or unquiesce command is issued, CICS BAC request server processing waits for the command to complete before continuing.
- Normal processing for the batch request utility command continues. For example, if the request is to close the file, CICS BAC request server issues a command to close the file.

Note that transaction IDs and programs associated with the files defined to be opened in RLS mode are processed in the same manner as all other file commands. That is, for close requests, transaction IDs and programs are disabled before the quiesce and close command. For open requests, transactions IDs and programs are enabled after the unquiesce and open commands.

When CICS BAC issues a quiesce command for a data set opened in RLS-mode, the request is passed to the SMSVSAM server by the CICS region in which the command is issued. SMSVSAM propagates the request to all CICS regions in the sysplex that currently have the data set unquiesced and open. This results in all files in those regions that refer to the data set being closed and the data set marked as quiesced. A data set cannot be opened in RLS mode by a CICS region if it is marked as quiesced. It can, however, be allocated and opened in non-RLS mode for batch processing. Once all the files are closed, control returns back to CICS BAC in the original CICS region, where command processing continues.

As a result of an unquiesce command issued by CICS BAC for a data set defined in RLS mode , the status of the data set throughout the sysplex is set to unquiesced, and the data set can subsequently be opened in RLS mode by any CICS region in the sysplex.

As already mentioned earlier, CICS BAC cannot perform its RLS processing for a SET FILE command if either the target CICS region, or the CICS BAC request server, is not active, because the batch request utility cannot determine whether or not the file is defined with RLSACCESS(YES). Therefore, if you know that files are defined in CICS regions to be opened in RLS mode, you should strongly consider using the CICS BAC CICS group feature to broadcast SET FILE commands that operate on those files. When you do this, the SET FILE command is sent in sequence to each of the CICS regions in the group. If CICS BAC is available in any of the target CICS regions named in the group member, it will detect that the file is defined as RLSACCESS(YES), and issue the appropriate quiesce or unquiesce command. Then, when the CICS BAC request server processes the SET FILE command in a subsequent region in the CICS group, it will recognize that the data set is already in the appropriate quiesce state and not issue another quiesce or unquiesce command.

CICS BAC also performs special processing for RLS-mode files during its region startup processing. If it detects that it needs to change the open or closed state of a file, and

- The file is defined as RLSACCESS(YES), and
- CICS BAC RLS processing is allowed for the file

CICS BAC first checks the current quiesce state of the associated data set and issue the corresponding quiesce or unquiesce command as necessary.

You should give special consideration to RLS-mode files with respect to associating transaction IDs and programs with such files. For normal (non-RLS-mode) files, the timing of the processing of associated transaction IDs and programs depends on the requested status for the file.

- If you are closing the file, transactions and programs are disabled *before* the close request is complete.
- If you are opening the file, transactions and programs are enabled *after* the open request is complete.

This is also true for RLS-mode files. However, one of the effects of the quiesce command issued by CICS BAC is that all files in all regions in the sysplex that refer to the data set being quiesced are subsequently closed by the CICS regions. In those regions, the associated transaction IDs and programs have not yet been disabled, which could result in NOTOPEN conditions being raised.

For example, consider the following illustration:

```
FILE1 is present and open in both in CICSA and CICSB
In both regions, FILE1 points to DATASET1
In CICS BAC, transaction ID TRN1 and program PROGRAM1 are both associated
   with FILE1CICS BAC
 CICS group data set member CICSGRP1 contains CICS region names
   CICSA and CICSB and both these regions are available.
```

In this scenario, when the following command is processed by the CICS BAC request server:

```
SET FILE(FILE1),OPENSTATUS(CLOSE),PROCESSTRAN(YES),PROCESSPROG(YES),
 CICS(CICSGRP1)
```

the following sequence of processing takes place:

1. In CICSA:
   a. CICS BAC issues SET commands to disable TRN1 and PROGRAM1
   b. CICS BAC detects that FILE1 has RLSACCESS(YES)
   c. CICS BAC issues a SET command to quiesce DATASET1, and on notification from CICSA, SMSVSAM propagates the quiesce request to CICSB.
2. In CICSB:
   a. CICS closes FILE1 as a result of the quiesce command for DATASET1
   b. SMSVSAM marks DATASET1 as quiesced.
3. In CICSA, CICS BAC closes FILE1
4. In CICSB:
   a. CICS BAC issues SET commands to disable TRN1 and PROGRAM1
   b. CICS BAC detects that FILE1 has RLSACCESS(YES), but its quiesce state is correct so it does not issue a quiesce command
   c. CICS BAC detects that FILE1 is currently closed, so it does not issue a close command.
5. Processing is complete.

A potential problem in this illustration is that the TRN1 and PROGRAM1 are not disabled in CICSB until *after* the file has already been closed. This problem can be eliminated by using application groups instead of associating transaction IDs and programs with RLS-mode files. For example, in this variation of our scenario, everything is the same as before except that TRN1 and PROGRAM1 are *not*

associated with FILE1. Instead, TRN1 and PROGRAM1 are defined as objects in the CICS BAC application group FILE1GRP. Thus, when the following commands are issued:

```
DEFAULT CICS(CICSGRP1)
SET GROUP(FILE1GRP),ENABLESTAT(DISABLED)
SET FILE(FILE1),OPENSTATUS(CLOSE)
```

the following sequence of processing takes place:

1. In CICSA, CICS BAC issues SET commands to disable TRN1 and PROGRAM1
2. In CICSB, CICS BAC issues SET commands to disable TRN1 and PROGRAM1.
3. In CICSA:
   a. CICS BAC detects that FILE1 has RLSACCESS(YES)
   b. CICS BAC issues a SET command to quiesce DATASET1.
4. In CICSB, FILE1 is closed as a result of the quiesce command for DATASET1
5. SMSVSAM marks DATASET1 as quiesced
6. 7. In CICSA, CICS BAC closes FILE1
7. In CICSB:
   a. CICS BAC detects that FILE1 has RLSACCESS(YES), but its quiesce state is correct so it does not issue a quiesce command
   b. CICS BAC detects that FILE1 is currently closed, so it does not issue a close command.
8. 10. Processing is complete.

Note that in the second illustration using an application group, TRN1 and PROGRAM1 are disabled in both regions *before* the file is quiesced and closed in either region. Similar results can be achieved with other variations of the above, including creating one group to disable the program and transaction and close the file. In this scenario, a second application group should be used to open the file and enable the program and transaction, with special care being taken when creating both groups to specify the objects in the appropriate order. That is, in the close group, specify all transaction IDs and programs before the file or files. Likewise, in the open group, specify the file or files before all transaction IDs and programs.

For more information about how CICS quiesces data sets in use in RLS mode within a sysplex, see the *CICS Recovery and Restart Guide* for *CICS Transaction Server for z/OS*.

Chapter 9. CICS BAC security

CICS BAC security provides complete security for all CICS BAC resources, with the flexibility that enables you to permit access as required by different users. User IDs can be set up with one level of authority in one CICS region and different levels of authority in other CICS regions. Additionally, you can restrict some users to CICS BAC administrative functions, such as the creation, modification, and deletion of CICS control file objects while other users are able only to submit jobs that execute commands against those objects. Additional granularity exists to further restrict users to certain types of objects, such as files and programs, or to specific objects, such as a payroll file.

CICS BAC security is implemented through the RACROUTE macro of the MVS system authorization facility (SAF) interface to route authorization requests to an external security manager (ESM), such as RACF. For convenience, this chapter uses RACF terminology when referring to security objects, such as profiles and resource classes, and refers to RACF as the ESM. If you are using a different ESM, substitute the corresponding terminology that is used by your ESM.

When you attempt to access a CICS BAC resource, CICS BAC builds a unique security resource name for the object you are trying to access. Using this security resource name, CICS BAC issues a RACROUTE call to verify your authority to access the resource, specifying the FACILITY general resource class. To differentiate CICS BAC resources from other resources within the FACILITY class, all CICS BAC resource names begin with $CBK (see "CICS BAC resource names" on page 124).

CICS BAC denies access to a resource if the RACROUTE return code indicates that the user is not authorized to access the resource. CICS BAC allows access if the RACROUTE return code indicates that the user is authorized to access the resource or if it indicates that the resource is not protected.

CICS BAC also supports CICS EXCI security; see "CICS BAC support for CICS EXCI security" on page 128.

Enabling CICS BAC resource security

By default, CICS BAC security mechanisms permit all users full access to CICS BAC resources unless you take steps to protect them by defining the required security profiles in the RACF database. This is because of the way the system authorization facility (SAF) responds if it cannot find the security profile for a specified resource. If SAF finds that a security profile is not found it neither grants nor refuses the access request, and in this case CICS BAC allows the request. Thus, to ensure security is fully enabled, define the required profiles to cover all your CICS BAC resources.

To ensure that no one can have access by default, define one generic profile to protect all resources, and then give specific access to resources as required. For example, use the following command to protect everything:

```
RDEFINE FACILITY $CBK.**  UACC(NONE)
```

With a universal access (UACC) of NONE, you deny access to all CICS BAC resource names that are not covered by a more explicit profile.

CICS BAC resource names

All CICS BAC resource names conform to the following format:

`$CBK.applid.accessType.objectType.objectName`

The prefix $CBK is a constant that identifies all CICS BAC resource names and distinguishes them from the resource names of other products.

The remaining parts of CICS BAC resource names are defined as follows:

applid Identifies the CICS region to which the resource belongs.

accessType
Indicates the type of access required for the resource and can be one of the following values:

ADMIN
Indicates that the resource resides in a VSAM KSDS control file and access is required for administrative purposes such as to read, edit, or delete the resource entry.

EXECUTE
Indicates a resource that is the subject of a batch request utility command that operates on the named resource.

objectType
The definition of *objectType* is dependent upon whether the value of *accessType* is ADMIN or EXECUTE, and is explained in "Administrative object types and object names" on page 125 and "Execution object types and object names" on page 126.

objectName
The definition of *objectName* is dependent upon whether the value of *accessType* is ADMIN or EXECUTE, and is explained in "Administrative object types and object names" on page 125 and "Execution object types and object names" on page 126.

Defining CICS BAC FACILITY class profiles

About this task

You use the RACF RDEFINE command to define FACILITY general resource class profiles, and the PERMIT command to grant and restrict access to resources based on a user ID or RACF group. If you do not have access to the RACF security database to create the required profiles yourself, ask your security administrator to create them for you using the information you provide, based on the information in this chapter. For more information about RACF security, see the *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683, and the *z/OS Security Server RACF Command Language Reference*, SA22-7687.

Using the information about the structure of CICS BAC resource names explained in the previous topic, you can define the required FACILITY class resource profiles for all your CICS BAC resources using the RACF RDEFINE command as follows:

`RDEFINE FACILITY $CBK.applid.accessType.objectType.objectName UACC(NONE)`

where *applid*, *accessType*, *objectType*, and*objectName* are as defined in "CICS BAC resource names."

When you have defined in this way all the CICS BAC resources to which you need to control access, the next step is to give specific user IDs or group IDs permission to access these resources. Typically, these will be the user IDs that are specified on a batch request utility job, a file maintenance utility job, the user IDs of users of the workstation administration client, or the TSO user of the ISPF administration interface. The UACC of NONE ensures that only the users specifically authorized are able to access your CICS BAC resources.

Examples of FACILITY class definitions

This section provides some illustrations of FACILITY class definitions, with two examples of different access levels for two different groups. The examples are based on a CICS region with applid CICSPROD, and you need to permit two groups of user different levels of access to objects in the CICS region control file.

- **Example 1**: The first group of users is the application development group of users, who are defined in the RACF group APPLDEV. This group is to be permitted to create, view, update, and delete any object for CICSPROD. However, they are not to be allowed to execute the objects through the batch request utility.
- **Example 2**: The second group of users is the production operations staff, who are in the RACF group PRODOPS. This group is to be permitted to view and execute any of the objects for CICSPROD, but they are not to be permitted to create, edit, or delete them.

The first step, for both examples, is to disallow all accesses within CICSPROD as a default by issuing the following RACF RDEFINE command:

```
RDEFINE FACILITY $CBK.CICSPROD.** UACC(NONE)
```

With all access to CICSPROD resources denied by default, you can enable the APPLDEV group requirements.

```
RDEFINE FACILITY $CBK.CICSPROD.ADMIN.** UACC(NONE)
```

This profile is specific to the ADMIN object type and takes precedence over the generic CICSPROD profile defined above. Using this profile you can grant the access required by the APPLDEV group using the following command.

```
PERMIT $CBK.CICSPROD.ADMIN.** CLASS(FACILITY) ID(APPLDEV) UACC(UPDATE)
```

Example 1:

Example 2:
To enable the PRODOPS group to execute commands, define the profile that is specific to the EXECUTE object type, which takes precedence over the generic CICSPROD profile defined earlier.

```
RDEFINE FACILITY $CBK.CICSPROD.EXECUTE.** UACC(NONE)
```

Using both this profile and the one defined in Example 1, you can now grant the PRODOPS group update access to the EXECUTE profile and read only access to the ADMIN profile:

```
PERMIT $CBK.CICSPROD.EXECUTE.** CLASS(FACILITY) ID(PRODOPS) UACC(UPDATE)
PERMIT $CBK.CICSPROD.ADMIN.** CLASS(FACILITY) ID(PRODOPS) UACC(READ)
```

Administrative object types and object names

CICS BAC generates administrative resource names when the workstation administration client, the file maintenance utility, or the ISPF administration interface attempts to access a record in a CICS region control file. Accesses that simply read the record require only a UACC of READ for the resource. Examples

of this are the LIST command in the file maintenance utility and the download and browse functions of the workstation administration client. Other accesses, such as insert, replace, or delete require a UACC of UPDATE for the resource. Examples of this are the ADD, UPDATE, and DELETE commands of the file maintenance utility and the upload and delete from host functions of the workstation administration client. The name generated for the security check is the name you need to define to RACF in the FACILITY general resource class.

The *objectType* and *objectName* fields for *accessType* ADMIN are derived according to the following table:

objectType	*objectName*
FILE	The name of the file object being accessed.
TDQUEUE	The name of the transient data queue object being accessed.
TRANSID	The name of the transaction object being accessed.
PROGRAM	The name of the program object being accessed.
APPGROUP	The name of the application group object being accessed.
APPLIST	Name of the application list object being accessed.
REGION	The literal PROPERTY for access to a CICS region object.

Some examples of the ADMIN access type resource names are as follows:

- **Example 1**: For a payroll file object called PAYROLL that is used in CICS region CICSP, define a profile in the FACILITY class as follows:

  ```
  RDEFINE FACILITY $CBK.CICSP.ADMIN.FILE.PAYROLL UACC(NONE)
  ```

  To read this file object, a user requires specific READ access for this profile. Alternatively, you could grant everyone access by changing the universal access to UACC(READ).
- **Example 2**: For an application group named DOCUMENT in CICS region CICSTEST, define a profile in the FACILITY class as follows:

  ```
  RDEFINE FACILITY $CBK.CICSTEST.ADMIN.APPGROUP.DOCUMENT UACC(NONE)
  ```

  To delete this application group object from the CICS region control file a user requires specific UPDATE access for this profile.
- **Example 3**: For a CICS region named CICST, define a profile in the FACILITY class as follows:

  ```
  RDEFINE FACILITY $CBK.CICST.ADMIN.REGION.PROPERTY UACC(NONE)
  ```

  To update the CICS region object, a user requires specific UPDATE access for this profile.

Execution object types and object names

CICS BAC generates execution resource names when the batch request utility attempts to execute a command. CICS BAC uses these resource names to ensure

that the user has the authority to execute the command. Note that a user requires ACCESS(UPDATE) to be able to execute a command against any of these resource names.

objectType	*objectName*
FILE	The name of the file object named on a SET command.
TDQUEUE	The name of the transient data queue object named on a SET command.
TRANSID	The name of the transaction object named on a SET command..
PROGRAM	The name of the program object named on a SET command.
APPGROUP	The name of the application group object named on a SET command.
APPLIST	The name of the application list object named on a SET command.
LINK	The name of the program object named on a LINK command.
RUN	The literal CEMT for a RUNCEMT command.
START	The name of the transaction named on a START command.

Some examples of the EXECUTE access type resource names are as follows:

- **Example 1**: For an accounts file object called ACCOUNTS that is used in CICS region CICSA, define a profile in the FACILITY class as follows:

  ```
  RDEFINE FACILITY $CBK.CICSA.EXECUTE.FILE.ACCOUNTS UACC(NONE)
  ```

  To issue a SET command on this file object, a user requires specific UPDATE access for this profile. For example:

  ```
  PERMIT $CBK.CICSA.EXECUTE.FILE.ACCOUNTS CLASS(FACILITY) ID(userid) ACCESS(UPDATE)
  ```

- **Example 2**: For a program named ACCT1 in CICS region CICSTEST, define a profile in the FACILITY class as follows:

  ```
  RDEFINE FACILITY $CBK.CICSTEST.EXECUTE.LINK.ACCT1 UACC(NONE)
  ```

  To issue a LINK command for this program a user requires specific UPDATE access for this profile. For example:

  ```
  PERMIT $CBK.CICSTEST.EXECUTE.LINK.ACCT1 CLASS(FACILITY) ID(userid) ACCESS(UPDATE)
  ```

- **Example 3**: To control access to the RUNCEMT command in CICS region CICSTEST, define a profile in the FACILITY class as follows:

  ```
  RDEFINE FACILITY $CBK.CICST.EXECUTE.RUN.CEMT UACC(NONE)
  ```

  To issue a RUNCEMT command in CICS region CICSTEST, a user requires specific UPDATE access for this profile. For example:

  ```
  PERMIT $CBK.CICSTEST.EXECUTE.RUN.CEMT CLASS(FACILITY) ID(userid) ACCESS(UPDATE)
  ```

CICS BAC support for CICS EXCI security

CICS BAC uses the external CICS interface (EXCI) provided by CICS whenever it determines that a request is to be serviced by the CICS BAC request server running in a target CICS region. This happens when CICS BAC is active in the target CICS region and a request from the workstation administration client, the batch request utility, the file maintenance utility, or the ISPF administration interface is passed over an EXCI link to the target CICS region.

In all cases, when CICS BAC issues an EXCI command, the user ID associated with the command is the user ID of the requestor, as follows:

- For a CICS BAC batch request utility request, or a CICS BAC file maintenance utility request, it is the user ID associated with the batch request utility job step, or the file maintenance utility job step, respectively.
- For a workstation administration client request, it is the user ID associated with the CICS BAC communication server to which the workstation administration client is connected.
- For an ISPF administration interface request, it is the TSO user ID of the user of the interface

When CICS processes an EXCI command, the type of security checking performed depends on the ATTACHSEC value provided on the generic EXCI CONNECTION definition used by CICS BAC. If you have specified ATTACHSEC(LOCAL), the CICS region performs link security checking against requests from CICS BAC. If you have specified ATTACHSEC(IDENTIFY), the CICS region performs user security checking against the user ID provided by the CICS BAC. In each case, CICS BAC uses the user ID associated with the origin of the request as described above. Therefore, you must ensure that the user ID associated with the CICS BAC EXCI request has sufficient authority with regard to transaction and resource security definitions in the target CICS region. This ensures that the CICS BAC request server is able service the request in the target CICS region.

CICS security does not apply, of course, when CICS BAC is not active in the target CICS region, or the target CICS region is not available. In this case, the request is serviced directly against the CICS region control file by the communication server, batch request utility, or file maintenance utility, as appropriate.

CICS BAC internal security checking as described the earlier part of this chapter is always performed *before* an EXCI request is issued.

See the *CICS External Interfaces Guide* for more information regarding EXCI security and the *CICS RACF Security Guide* for more information about CICS transaction and resource security checking.

Chapter 10. CICS BAC messages

CICS BAC produces three types of messages.

- Console messages to advise the system operator of execution progress, or request a decision
- Error messages reporting that an error has occurred
- Audit log messages written to the audit log if logging is active for the target CICS region.

This chapter explains the messages issued by the various CICS BAC components. The messages are grouped by the CICS BAC component that issues them.

Message identifier format

All CICS BAC message identifiers are ten characters long. The first three characters are always CBK, the product identifier for CICS BAC. The next two characters identify the program that generated the message. These program identifiers are not specified in the message documentation and instead are represented by "*xx*". The next four characters are a four digit number that identifies the message. The last character in the message identifier is the message severity code, as follows:

- **I – Information message**: No operator action is required.
- **R – Response required message**: An event has occurred that requires an operator response to a previous CICS BAC message displayed on the console. Operator action is required to enable CICS BAC to continue processing.
- **W – Warning message**: Unexpected results might have occurred. You should closely examine any relevant output. Operator action may be required.
- **E – Error message**: An unexpected error has occurred. CICS BAC processing could be negatively affected to the point that it can no longer perform its functions. Operator action is most likely required to correct the error and allow CICS BAC to function correctly.

Message information format

Each CICS BAC message in this chapter is presented in the following format:

Message identifier
Identifies the message in the format described above.

Message text
Provides the words and inserts that make up the message as displayed by CICS BAC.

Explanation
Describes the events that have caused the production of the message.

System action
Describes the action that has been, or is to be, taken by CICS BAC.

User response
Describes the recommended action that you should take.

Destination
Specifies the device or log to which the message is sent. This can be one of the following:

- **Console** – the operating system master console.
- **Message log** – the message log for the CICS BAC component that issues the message.
- **Terminal** – in the case of CICS BAC messages produced as a result of action taken by the CICS BAC transaction, the CICS terminal from which the transaction was invoked.
- **Audit log** – the data set or SYSOUT class allocated to the CBKLOG DD statement. Note that CBKLOG can be dynamically allocated by the CICS BAC component. See "The audit log data set" on page 25 for more information.

The following sections contain messages for:

CICS BAC communication server messages (CBK*xx*1000–1499)

The following messages are issued by the CICS BAC communication server:

CBKxx1001I CICS Batch Application Control for z/OS communication server initialization starting.

Explanation: The CICS BAC communication server has begun its initialization process.

System action: None.

User response: None.

Destination: Console

CBKxx1005I *cbkid* **CICS Batch Application Control for z/OS communication server initialization complete: MAB=***address*

Explanation: The CICS BAC communication server has successfully completed its initialization process.

System action: The communication server waits for command requests from the operator and for requests from a workstation administration client.

User response: None.

Destination: Console

CBKxx1015E *cbkid* **Empty or no control file table located. Communication server abending with abend code U1120.**

Explanation: An error has occurred while attempting to locate the control file table. Either it does not exist or the data set member does not contain any control file definitions.

System action: The CICS BAC communication server abends with a U1120 abend code.

User response: Add or update the PDS member that defines the control file table, then restart the communication server.

Destination: Console

CBKxx1030E Error detected in input parameter from *parmloc***. Processing terminating.**

Explanation: The communication server has detected an error in an input parameter. In the message text, *parmloc* specifies the location from which the parameter was read, and which can be one of the following values:

- MVS PARMLIB

- CBKPARMS member CBKSRVR
- EXEC PARM

This message is preceded by a message giving the details of the parameter error.

System action: The communication server terminates.

User response: Correct the parameter error and restart the communication server.

Destination: Console

CBKxx1050E *cbkid* **A server with this ID is already active. Initialization terminates.**

Explanation: Another communication server with the ID of *cbkid* is already active in another address space. A communication server ID cannot be active on more than one MVS image at a time.

System action: The communication server address space that is trying to start with a duplicate ID terminates.

User response: Check to see if the communication server which is already active is the same as the one being started. If so, use the one that is already active. If not, change the ID parameter for the communication server being started to give it a unique ID.

Destination: Console

CBKxx1090E Unable to load CICS EXCI module. CICS SDHEXCI load library may not be available: code1(*retcode***) code2(***rsncode***)**

Explanation: During startup processing, the communication server has attempted to load one of the required CICS EXCI modules to ensure the server has access to the load library. The load request has failed with the errors indicated by *retcode* and *rsncode*, which contain the contents of R15 and R1, respectively, as returned by the failed load attempt.

System action: The communication server terminates.

User response: Ensure the CICS SDFHEXCI load library is available to the communication server by placing it in the system linklist or in the communication server STEPLIB concatenation.

Destination: Console

CBKxx1098I *cbkid* **CICS Batch Application Control for z/OS server shutdown processing starting.**

Explanation: The communication server *cbkid* has received a command to shut down. It has begun the process of shutting down its address space.

System action: The communication server terminates.

User response: None.

Destination: Console

CBKxx1099I *cbkid* **CICS Batch Application Control for z/OS server shutdown complete.**

Explanation: The communication server *cbkid* has completed its process of shutting down.

System action: The communication server terminates.

User response: None

Destination: Console

CBKxx1150W *ddname* **member** *member* **not found.**

Explanation: The *member* member cannot be found in the *ddname* data set. This member provides the communication parameter values, and without this member the communication server uses all default values.

System action: Parameter processing continues without the indicated member.

User response: If the parameters are specified elsewhere, or if the default parameter values are sufficient, no action is needed. Otherwise, add the indicated member to the data set, specifying the needed parameter values.

Destination: Console

CBKxx1157W Server startup will continue. Server parameters with errors will be ignored.

Explanation: The communication server has accepted your reply GO to message CBK*xx*1156R. The communication server ignores the parameters with errors and continue its initialization process.

System action: The communication server continues to initialize.

User response: None.

Destination: Console

CBKxx1200I *cbkid* **The following startup parameters are in effect for this CICS BAC communication server:**

Explanation: The communication server *cbkid* has received a DISPLAYPARMS command. After displaying this message, the server issues one or more CBK*xx*1201I messages, where each message text contains the current parameter value for a communication server parameter. Note that the communication server also issues this message automatically during startup processing to display the parameters that are in effect.

System action: None.

User response: None.

Destination: Console

CBKxx1201I *cbkid loc parm*

Explanation: The communication server *cbkid* displays each parameter and its value pair *parm* immediately after it has displayed message CBK*xx*1200I. Each communication server parameter and its value is displayed in a separate CBK*xx*1201I message. The source *loc* from which the communication server obtained the parameter can be one of the following:

- **EXEC PARM** , indicating the parameter was taken from the PARM string of the JCL EXEC statement in the communication server startup procedure.
- **CBKPARMS** indicating the parameter was taken from the CBKPARMS data set member, CBKSRVR
- **PARMLIB** indicating the parameter was taken from the MVS logical parmlib concatenation
- **Default** indicating a default value.

System action: None.

User response: None.

Destination: Console

CBKxx1209I *cbkid* **End of CICS BAC communication server startup parameter list.**

Explanation: The communication server *cbkid* has finished displaying parameter and value pairs in response to a DISPLAYPARMS command (or during server initialization) and DISPLAYPARMS command processing is now complete.

System action: The DISPLAYPARMS command is complete.

User response: None.

Destination: Console

CBKxx1300I *cbkid* **CICS Region=***region***, DSN=***dsname***.**

Explanation: The communication server *cbkid* has received a DISPLAYCFT command and responds with one **CBK*xx*1300I** message for each entry in its control file table, CBKCFTBL. Each message indicates the CICS region *region* and its CICS BAC control file data set name *dsname* .

System action: None.

User response: None.

Destination: Console

CBKxx1301I *cbkid* **End of CICS control file data set name list.**

Explanation: The communication server *cbkid* has displayed, in one or more CBK*xx*1300I messages, all the entries in its control file table (CBKCFTBL) in response to a DISPLAYCFT command. The server has completed DISPLAYCFT command processing.

System action: The DISPLAYCFT command is complete.

User response: None.

Destination: Console

CBKxx1401I *cbkid* **CICS control file table construction starting.**

Explanation: A CICS BAC component is initializing and is beginning the process of building its internal control file table using the statements in the CBKCFTBL parmlib member. It uses the entries in this member to associate a CICS region with its VSAM control file. The CICS BAC component can be the communication server, the batch request utility, or the file maintenance utility. If the message is issued by the communication server, *cbkid* identifies the server, but in the case of the batch request utility or the file maintenance utility, it is the job name under which the utility is executing.

System action: None.

User response: None.

Destination: Console (if issued by the communication server) or the job log (if issued by one of the utilities).

CBKxx1404I *cbkid* **CBKPARMS member CBKCFTBL not found. Logical PARMLIB search will be attempted.**

Explanation: A CICS BAC component cannot find a control file table, CBKCFTBL, as a member of its CBKPARMS PDS. Either the member CBKCFTBL is not in the CBKPARMS PDS, or the CBKPARMS DD statement is missing from the component startup procedure.

The CICS BAC component can be the communication server, the batch request utility, or the file maintenance utility. If the message is issued by the communication server, *cbkid* identifies the server, but in the case of the batch request utility or the file maintenance utility, it is the job name under which the utility is executing.

System action: The CICS BAC component attempts to locate a CBKCFTBL member in the MVS logical parmlib concatenation.

User response: None.

Destination: Console (if issued by the communication server) or the job log (if issued by one of the utilities).

CBKxx1405E *cbkid* **PARMLIB member CBKCFTBL not found.**

Explanation: The CICS BAC component cannot find a control file table member, CBKCFTBL, in the MVS logical parmlib concatenation. CICS BAC has already failed to find CBKCFTBL in its own CBKPARMS PDS, and because it must have a CBKCFTBL to function correctly, the CICS BAC component cannot continue its

initialization. If the message is issued by the communication server, *cbkid* identifies the server, but in the case of the batch request utility or the file maintenance utility, it is the job name under which the utility is executing.

System action: The CICS BAC component terminates.

User response: Add a CBKCFTBL member to CICS BAC CBKPARMS PDS or to the MVS logical parmlib concatenation then restart the CICS BAC component.

Destination: Console (if issued by the communication server) or the job log (if issued by one of the utilities).

CBKxx1406E *cbkid ddname* **member CBKCFTBL is empty.**

Explanation: A CICS BAC component has found a CBKCFTBL member located in either the CBKPARMS PDS or the MVS logical parmlib concatenation, but the member does not contain any control file table entries. *ddname* indicates where the empty CBKCFTBL member has been found–showing a value of CBKPARMS or PARMLIB, as appropriate.

Provide a CBKCFTBL that contains at least one valid CICS APPLID and data set name pair for the CICS BAC component to initialize successfully. See "The control file table" on page 23 for information about the CBKCFTBL member and its definitions. If the message is issued by the communication server, *cbkid* identifies the server, but in the case of the batch request utility or the file maintenance utility, it is the job name under which the utility is executing.

System action: The CICS BAC component terminates.

User response: Update the indicated CBKCFTBL member to contain one or more valid control file entries, then restart the CICS BAC component.

Destination: Console (if issued by the communication server) or the job log (if issued by one of the utilities).

CBKxx1409I *cbkid* **CICS control file table successfully built from** *ddname* **member. Number of entries:** *num***.**

Explanation: A CICS BAC component has located the CICS control file table in the *ddname* DDNAME. From that CBKCFTBL table it has successfully built its own internal control file table that contains *num* entries. If the message is issued by the communication server, *cbkid* identifies the server, but in the case of the batch request utility or the file maintenance utility, it is the job name under which the utility is executing.

System action: None.

User response: None.

Destination: Console (if issued by the communication server) or the job log (if issued by one of the utilities).

CBKxx1450E *cbkid* **DYNALLOC error detected: code1(***code1***) code2(***code2***) DSN(***dsname***) DDN(***ddname***).**

Explanation: The communication server *cbkid* has attempted to allocate dynamically the data set *dsname* with DD name *ddname* and an error has occurred. The server has received return code *code1* and reason code *code2*.

System action: The data set is not allocated.

User response: Check for additional messages relating to this error, which should provide more information regarding the source of the problem. Take the corrective action indicated by these other messages. For information about the return and reason codes, see the *z/OS MVS Programming: Authorized Assembler Services Guide*.

Destination: Console

TCP/IP listener task messages (CBK*xx*1500–2499)

The following messages are issued by the TCP/IP listener task:

CBKxx1500I *cbkid* **CICS BAC communication server TCP/IP listener initialization starting.**

Explanation: The communication server *cbkid* TCP/IP listener subtask initialization is starting. The communication server uses the listener subtask to respond to CICS BAC workstation client requests.

System action: None.

User response: None.

Destination: Console

CBKxx1501I *cbkid* **CICS BAC no TCP/IP listener port supplied. Using default port.**

Explanation: The communication server *cbkid* did not find a TCP/IP listener port number specified in its startup parameters. It is using the default port number 55667.

System action: The TCP/IP listener initialization continues.

User response: None.

Destination: Console

CBKxx1502I *cbkid* **CICS BAC communication server TCP/IP listener initialization port:** *port*

Explanation: The communication server *cbkid* has read a TCP/IP port number from its runtime parameters

and when the TCP/IP listener initialization is finished, it waits for workstation client requests on TCP/IP port *port*.

System action: The TCP/IP listener initialization continues.

User response: None.

Destination: Console

CBKxx1503I *cbkid* **CICS BAC communication server TCP/IP listener initialization complete.**

Explanation: The communication server *cbkid* TCP/IP listener subtask initialization process is complete and the listener is ready to accept workstation client requests.

System action: The TCP/IP listener waits for client requests.

User response: None.

Destination: Console

CBKxx1504I *cbkid* **TCP/IP listener client request started:** *reqtype objtype user pcuser***.**

Explanation: The communication server *cbkid* TCP/IP listener has received a workstation client request and is attempting to process it. The request came from client *pcuser* and the server is authenticating this user with host user ID *user*. The requested action *reqtype* can be one of the following:

- DOWNLOAD
- UPLOAD
- BROWSE
- INQUIRE
- DELETE

The object type *objtype* being requested can be one of the following:

- FILE
- TDQUEUE
- TRANSID
- PROGRAM
- APPGROUP
- APPLIST
- CICS PROPERTIES.

System action: The communication server attempts to process the request.

User response: None.

Destination: Message log.

CBKxx1505I *cbkid* **TCP/IP listener client request completed:** *reqtype objtype user pcuser***.**

Explanation: The communication server *cbkid* TCP/IP listener has finished processing a workstation client request. The values of *reqtype* and *objtype* are the same as in message CBK*xx*1504I.

System action: None.

User response: None.

Destination: Message log.

CBKxx1506E *cbkid* **TCP/IP listener invalid function code received:** *func***.**

Explanation: The communication server *cbkid* TCP/IP listener has received a workstation administration client request containing an invalid function code, *func*.

System action: The TCP/IP listener discards the notification and continues to listen for CICS BAC workstation client requests.

User response: Contact Technical Support.

Destination: Console

CBKxx1507E *cbkid* **TCP/IP listener invalid client request received:** *reqtype objtype user pcuser***.**

Explanation: The communication server *cbkid* TCP/IP listener has received a workstation client request from user *pcuser* authorized as host user *user*. Either the request type *reqtype* or the object type *objtype* is invalid. For a description of *reqtype*, *objtype*, *user*, and *pcuser*, see message CBK*xx*1504I.

System action: The request is not processed.

User response: Contact Technical Support.

Destination: Console

CBKxx1508E *cbkid* **Timeout occurred on client request connection:** *reqtype objtype user pcuser***.**

Explanation: The communication server *cbkid* TCP/IP listener has timed out a workstation client request because the connection has exceeded the maximum wait time specified in the startup parameters. The request timed out is *reqtype objtype user pcuser*, the values of which are the same as in message CBK*xx*1504I.

System action: The client request terminates. Depending on the type of request, and at what point the request timed out, the request might, or might not, have been completed.

User response: Determine whether or not the request was completed. If not, reissue the request. You might want to increase the client request timeout value to lessen the likelihood of this occurring. If the problem persists, contact Technical Support.

Destination: Console

CBKxx1509E *cbkid* **Error detected on TCP/IP** *type* **request. R0(***r0***) R15(***r15***).**

Explanation: The communication server *cbkid* TCP/IP listener received an error condition from the *type* TCP/IP API function request. Register 0 *r0* and register 15 *r15* indicate the exact error returned by the API.

System action: The communication server issues write-to-operator-reply (WTOR) message CBK*xx*1510R to allow you to direct the listener on how it should proceed.

User response: Examine the return and reason codes for the API error. Then reply to the WTOR for message CBK*xx*1510R.

Destination: Console

CBKxx1510R *cbkid* **Reply RESET, SHUTDOWN to shut down the listener, or ABEND to abend the communication server.**

Explanation: The communication server *cbkid* has issued message CBK*xx*1509R and now requests your response, to indicate how the TCP/IP listener task is to proceed following the API error reported in the CBK*xx*1509E message.

System action: The TCP/IP listener task waits for your reply to the WTOR message.

User response: You can reply RESET, SHUTDOWN, or ABEND, with the following results:

RESET The communication server TCP/IP listener terminates its current connection and then attempts to re-establish a new one.

SHUTDOWN
The communication server terminates normally.

ABEND
The communication server terminates abnormally. This reply can be useful in obtaining diagnostic information for Technical Support.

Destination: Console

CBKxx1511I *cbkid* **RESET reply received. TCP/IP listener will be reset.**

Explanation: The communication server *cbkid* has accepted a RESET reply in response to message CBK*xx*1510R.

System action: The TCP/IP listener attempts to reset its TCP/IP connection.

User response: None.

Destination: Console

CBKxx1512W *cbkid* **SHUTDOWN reply received. The TCP/IP listener will be shut down.**

Explanation: The communication server *cbkid* has accepted a SHUTDOWN reply in response to message CBK*xx*1510R.

System action: The TCP/IP listener attempts to shut down.

User response: None.

Destination: Console

CBKxx1513E *cbkid* **ABEND requested. The TCP/IP listener will be terminated with abend code U1011.**

Explanation: The communication server *cbkid* has accepted an ABEND reply in response to the message CBK*xx*1510R.

System action: The TCP/IP listener terminates with a U1011 abend code.

User response: None.

Destination: Console

CBKxx1516E *cbkid* **TCP/IP listener invalid client object requested:** *objtype user pcuser***.**

Explanation: The communication server *cbkid* TCP/IP listener has received an invalid object type on a request from a workstation client. The request is from client name *pcuser* using host user ID *user*. The invalid object type *objtype* is rejected because it is *not* one of the following:
- FILE
- TDQUEUE
- TRANSID
- PROGRAM
- APPGROUP
- APPLIST
- CICS PROPERTIES.

System action: The communication server discards the request.

User response: Contact Technical Support.

Destination: Console

CBKxx1517I *cbkid* **CICS BAC communication server TCP/IP listener SHUTDOWN complete.**

Explanation: The communication server *cbkid* TCP/IP listener has shut down in response to a SHUTDOWN request and the server cannot accept any subsequent workstation client requests.

System action: The communication server no longer accepts client requests because it has no TCP/IP listener.

User response: None.

Destination: Console

CBKxx1518W *cbkid* **TCP/IP listener will be reset due to unrecoverable error.**

Explanation: The communication server *cbkid* TCP/IP listener has encountered an unrecoverable error and is forcing an internal reset to attempt to clear the error.

System action: The listener attempts to reset itself to clear the error condition that caused the reset.

User response: Examine the console for error messages associated with this message. Take the necessary steps to resolve the problem.

Destination: Console

CBKxx1519E *cbkid* **TCP/IP listener exceeded forced reset limit. Communication server terminated with abend code U1012.**

Explanation: The communication server *cbkid* listener, since it was last started, has forced an internal reset more times than is allowed. Because this could be a recursive error, the communication server address space terminates with abend code U1012.

System action: The communication server address space terminates and produces a dump.

User response: Determine the nature of the problem and take the necessary steps to resolve it. If you cannot determine the cause of the problem , contact Technical Support.

Destination: Console

CBKxx1520W *cbkid* **TCP/IP listener task is not APF authorized. Unable to validate client userids.**

Explanation: The communication server *cbkid* has detected, during TCP/IP listener initialization, that the TCP/IP subtask is not APF-authorized. The TCP/IP subtask needs to be APF-authorized to enable it to perform MVS host user ID validation.

System action: The TCP/IP listener continues it initialization. However, it is not able to validate the host user ID and password received for each client workstation request it receives.

User response: Your action depends on whether you want user ID validation on the host for workstation client requests as follows:

- If you don't require host user ID validation, take no action and ignore this message.
- If you do require host user ID validation, shut down the communication server, ensure all data sets in the STEPLIB library in the communication server startup procedure are APF-authorized, and then restart the server.

Destination: Console

CBKxx1521W *cbkid* **Validation of client userid skipped due to lack of APF authorization.**

Explanation: The communication server *cbkid* has received a workstation client request, but the TCP/IP listener is unable to validate the host user ID and password associated with the request. This is because the TCP/IP listener subtask is not APF-authorized (see message CBK*xx*1520W issued during TCP/IP initialization).

System action: The communication server processes the workstation client request without validating the host user ID or password.

User response: Your action depends on whether you want user ID validation on the host for workstation client requests as follows:

- If you don't require host user ID validation, take no action and ignore this message.
- If you do require host user ID validation, shut down the communication server, ensure all data sets in the STEPLIB library in the communication server startup procedure are APF-authorized, and then restart the server.

Destination: Message log.

CBKxx1522W *cbkid* **Client request rejected. Target CICS region** *applid* **did not go through CICS BAC shutdown processing.**

Explanation: The communication server has received a workstation administration client request. The target CICS region is not available, and that CICS BAC did not go through normal termination the last time it was active in the target CICS region. The CICS BAC region properties defined for the target CICS region specify that client requests are to be rejected in this situation.

System action: The workstation administration client request is rejected.

User response: Depending on the CICS BAC region properties for the target region, you might have to restart CICS BAC in the region to reset the CICS BAC shutdown flag for the region.

Destination: Console

Communication server operator command messages (CBK*xx*2500–2799)

The following messages are issued by the communication server operator commands component:

CBKxx2500I cbkid Operator command subtask starting initialization process.

Explanation: The CICS BAC communication server *cbkid* operator command subtask initialization is starting. The communication server uses the operator command subtask to obtain commands from the operator console.

System action: None.

User response: None.

Destination: Console

CBKxx2501I *cbkid* **Operator command subtask initialization complete. Accepting operator commands.**

Explanation: The CICS BAC communication server *cbkid* operator command subtask has completed the initialization process and is ready to accept operator commands.

System action: The command subtask waits for operator commands.

User response: None.

Destination: Console

CBKxx2502I *cbkid* **Operator command subtask now accepting commands.**

Explanation: The CICS BAC communication server *cbkid* operator command subtask is now accepting operator commands.

System action: The command subtask waits for operator commands.

User response: None.

Destination: Console

CBKxx2505I *cbkid* **Unknown or non-unique command entered. Command not processed.**

Explanation: The CICS BAC communication server *cbkid* has received a command that is either unknown or it is abbreviated such that the command subtask cannot differentiate it from two or more commands that have the same abbreviation.

System action: The command is not processed.

User response: Change the entered command to that of a known command or enter enough characters to make the command unique.

Destination: Console

CBKxx2501I *cbkid* **SHUTDOWN command received. CICS BAC server will be shut down.**

Explanation: The communication server has received a shutdown request. The server begins shutdown processing.

System action: The communication server is shut down.

User response: None.

Destination: Console

CBKxx2517I *cbkid* **Operator command subtask termination starting. Operator commands no longer accepted.**

Explanation: The CICS BAC communication server *cbkid* operator command subtask has received a request to shut down.

System action: The operator command subtask no longer accepts commands.

User response: None.

Destination: Console

CBKxx2518I *cbkid* **Operator command subtask termination complete.**

Explanation: The CICS BAC communication server *cbkid* operator command subtask has completed its shut down in response to an operator SHUTDOWN request.

System action: None.

User response: None.

Destination: Console

CBKxx2519E *cbkid* **Operator command subtask terminating abnormally with abend code U1042.**

Explanation: An internal error has occurred in the CICS BAC communication server *cbkid* operator command subtask, and therefore it is terminating the communication server address space with a U1042 abend.

System action: The communication server address space produces a dump and then terminate.

User response: Check the message log for any additional messages that might indicate the cause of

the problem. If you cannot determine the cause of the problem, contact Technical Support.

Destination: Console

Parser error messages (CBK*xx*2800–3999)

The following messages are issued by the command parser for input parameters and commands. The command parser is a common component that issues messages on behalf of the following:

- The batch request utility when it is processing CICS BAC commands read from the CBKIN data set.
- The communication server when it is processing runtime parameters read from the PARM string of the EXEC statement or from the CBKSRVR member of CBKPARMS.

CBKxx2800E Unknown or non-unique command. Request terminated.

Explanation: The command being processed is either unknown or the abbreviation is not unique. The command is the first word within the input string.

System action: The command is not processed.

User response: Correct the command and retry.

Destination: Console (communication server), message log (batch request utility), or 3270 terminal issuing the CICS BAC transaction.

CBKxx2801E Invalid end of parameter string. Parameter string must end with a ')'.

Explanation: The last non-blank character in a parameter value string is not a right parenthesis.

System action: The command is not processed.

User response: Each parameter value string must be enclosed in parentheses. Correct the input string and retry.

Destination: Console (communication server), message log (batch request utility), or 3270 terminal issuing the CICS BAC transaction.

CBKxx2802E Parameter at offset *offset* too long. Maximum parameter length is sixteen characters.

Explanation: The input string being processed contains a command or parameter that is longer than the maximum of 16 characters. The *offset* value indicates the position of the field that is in error within the input string.

System action: The command is not processed.

User response: Correct the input string and retry.

Destination: Console (communication server), message log (batch request utility), or 3270 terminal issuing the CICS BAC transaction.

CBKxx2803E Unable to locate end of parameter beginning at offset *offset*.

Explanation: The command processor cannot find the end of a parameter in an input string. The relative *offset* value indicates the position of the parameter that is in error.

System action: The command is not processed.

User response: Correct the input string and retry.

Destination: Console (communication server), message log (batch request utility), or 3270 terminal issuing the CICS BAC transaction.

CBKxx2804E Unknown parameter beginning at offset *offset*.

Explanation: The command parser cannot identify the parameter. The *offset* value indicates the location of the unidentified parameter.

System action: The command is not processed.

User response: Correct the input string and retry.

Destination: Console (communication server), message log (batch request utility), or 3270 terminal issuing the CICS BAC transaction.

CBKxx2805E Length of parameter value beginning at offset *offset* is incorrect for associated parameter.

Explanation: The command parser has detected that an operand is the wrong length for the parameter with which it is associated. This means it could be either too long or, in some cases, too short. The *offset* value indicates the position of the invalid value.

System action: The command is not processed.

User response: Correct the operand and retry.

Destination: Console (communication server), message log (batch request utility), or 3270 terminal issuing the CICS BAC transaction.

CBKxx2806E Expecting separator character "," at offset *offset*. Not found.

Explanation: While parsing an input string, the parser was expecting the separator character (comma), but it found a different character. The *offset* value indicates the position of the unexpected character.

System action: The command is not processed.

User response: Correct the input string and retry.

Destination: Console (communication server), message log (batch request utility), or 3270 terminal issuing the CICS BAC transaction.

CBKxx2807E Invalid character in parameter value at offset *offset*.

Explanation: The parser has found an invalid character in an operand field in an input string. The *offset* value indicates the position of the invalid character.

System action: The command is not processed.

User response: Correct the operand and retry.

Destination: Console (communication server), message log (batch request utility), or 3270 terminal issuing the CICS BAC transaction.

CBKxx2808E Invalid value for CICSSTARTMODE value at offset *offset*. SET or UPDATE required.

Explanation: The operand value for the CICSSTARTMODE parameter is invalid. The valid values are SET and UPDATE. The *offset* value indicates the position of the invalid operand.

System action: The command is not processed.

User response: Correct the operand and retry.

Destination: Console or 3270 terminal issuing the CICS BAC transaction.

CBKxx2809E Invalid value for *keyword* parameter at offset *offset*. Possibly too high or low.

Explanation: The parser has found an invalid numeric operand for the parameter *keyword*. For example, the operand value might be too high, too low, or contain non-numeric characters. The *offset* value indicates the position of the invalid operand.

System action: The command is not processed.

User response: Correct the numeric operand and retry.

Destination: Console (communication server), message log (batch request utility), or 3270 terminal issuing the CICS BAC transaction.

CBKxx2810E Command at offset *offset* too long. Maximum command length is sixteen characters.

Explanation: The length of the command in the input string is too long. The maximum length of a valid command is 16 characters. The *offset* value indicates the position of the command in error.

System action: The command is not processed.

User response: Correct the input string and retry.

Destination: Console (communication server), message log (batch request utility), or 3270 terminal issuing the CICS BAC transaction.

CBKxx2811E Non-unique parameter beginning at offset *offset*.

Explanation: There is a non-unique parameter in an input command, which means that it could be an abbreviation for more than one paraemter. The *offset* value indicates the position of the non-unique parameter.

System action: The command is not processed.

User response: Re-enter the command, ensuring that the parameter in error is a unique abbreviation.

Destination: Console (communication server), message log (batch request utility), or 3270 terminal issuing the CICS BAC transaction.

CBKxx2814E *keyword1* keyword is mutually exclusive with the *keyword2* keyword.

Explanation: The command being processed contains both the *keyword1* and *keyword2* parameters. These two parameters are mutually exclusive for this command.

System action: The command is not processed.

User response: Remove the redundant parameter from the command and retry.

Destination: Message log

CBKxx2815E OPENSTATUS value at offset *offset* is invalid. OPEN or CLOSED required.

Explanation: The operand value for the OPENSTATUS parameter is invalid. The valid values are OPEN and CLOSED. The *offset* value indicates the position of the invalid operand.

System action: The batch request utility rejects the command.

User response: Re-enter the command with a valid operand on the OPENSTATUS parameter.

Destination: Message log

CBKxx2816E ENABLESTATUS value at offset *offset* is invalid. ENABLE or DISABLE required.

Explanation: The operand value for the ENABLESTATUS parameter is invalid. The valid values are ENABLE and DISABLE. The *offset* value indicates the position of the invalid operand.

System action: The command is not processed.

User response: Re-enter the command with a valid operand on the ENABLESTATUS parameter.

Destination: Message log

CBKxx2817E *keyword* value at offset *offset* is invalid. YES or NO required.

Explanation: The operand value for the *keyword* parameter is invalid. The valid values are YES and NO. The *offset* value indicates the position of the invalid operand.

System action: The command is not processed.

User response: Re-enter the command with a valid operand on the parameter *keyword*.

Destination: Message log

CBKxx2818E DSN value at offset *offset* is invalid. PRIMARY or ALTERNATE required.

Explanation: The operand value specified on the DSN parameter is invalid. The valid values are PRIMARY and ALTERNATE. The *offset* value indicates the position of the invalid operand.

System action: The command is not processed.

User response: Re-enter the command with a valid operand on the DSN parameter.

Destination: Message log

CBKxx2820E OPENSTATUS and ENABLESTATUS parameters missing. One or the other is required for a SET *object* command.

Explanation: You have specified a SET *object* command that does not contain either the OPENSTATUS or the ENABLESTATUS parameter, and at least one of these is required on a SET *object* command.

System action: The command is not processed.

User response: Re-enter the command with at least one of the OPENSTATUS or ENABLESTATUS parameters to the command.

Destination: Message log

CBKxx2821E *keyword* parameter missing. It is required for a SET *object* command.

Explanation: You have specified a SET *object* command without the required *keyword*.

System action: The command is not processed.

User response: Add the required parameter to the command and retry.

Destination: Message log

CBKxx2822E DISPOSITION value at offset *offset* is invalid. SHR or OLD required.

Explanation: The operand value for the DISPOSITION parameter is invalid. The valid values are SHR and OLD. The *offset* value indicates the position of the invalid operand

System action: The command is not processed.

User response: Re-enter the command with either SHR or OLD on the DISPOSITION parameter.

Destination: Message log

CBKxx2823E Required object parameter is missing. FILE, TDQUEUE, TRANSID, PROGRAM, GROUP or LIST is required.

Explanation: The SET command read by the batch request utility does not contain a valid object. The valid object parameters are FILE, TDQUEUE, TRANSID, PROGRAM, GROUP, and LIST.

System action: The command is not processed.

User response: Add the required object parameter to the command and retry.

Destination: Message log

CBKxx2824E Unable to determine CICS to send this request to.

Explanation: The batch request utility is processing a command that does not specify a CICS applid, and the utility does not have a default applid either. This could be because the most recent DEFAULT command did not specify the CICS parameter, un-setting one that was defined earlier. The CICS parameter is needed, either on the command itself, or as a previously specified default value, to determine which CICS region to send the request to.

System action: The command is not processed.

User response: Add the CICS parameter to either the current command, the most recent DEFAULT command or to a new DEFAULT command and retry.

Destination: Message log

CBKxx2825E Request is not in command format. First field should contain the command name.

Explanation: The first field in the input string is not a command, that is, it is in parameter format instead of command format. Parameter format means that the field is followed immediately, in the next character position, by an operand enclosed in parentheses. In command format the first field is followed by a blank.

System action: The command is not processed.

User response: Correct the command syntax to specify a command name in the first field and retry.

Destination: Message log

CBKxx2826E *field* is not the name of a valid command.

Explanation: The first field in the input string, denoted by *field*, is not the name of a valid command. The valid command names are DEFAULT, LINK, RUNCEMT, SET and START.

System action: The command is not processed.

User response: Specify a valid command in the first field of the statement and retry.

Destination: Message log

CBKxx2827E Parameter *keyword* at offset *offset* has already been specified.

Explanation: The parameter *keyword* is specified more than once in the current statement. The *offset* value indicates the position of the second occurrence.

System action: The command is not processed.

User response: Remove the duplicate parameter and retry.

Destination: Message log

CBKxx2828E NOTACTIVE value at offset *offset* is invalid. CONTINUE or TERMINATE required.

Explanation: The NOTACTIVE operand value is invalid. The valid values for NOTACTIVE are CONTINUE and TERMINATE. The *offset* value indicates the position of the invalid operand.

System action: The command is not processed.

User response: Specify CONTINUE or TERMINATE on the NOTACTIVE parameter and retry.

Destination: Message log

CBKxx2829E *keyword1* is not specified. It is required with the *keyword2* parameter.

Explanation: The command contains *keyword2* but not *keyword1*. You must specify *keyword1* parameter if you specify *keyword2* .

System action: The command is not processed.

User response: Either remove the *keyword2* parameter or add the *keyword1* parameter and retry.

Destination: Message log

CBKxx2830E *keyword* is not specified. It is required with the *command* command.

Explanation: The command does not contain *keyword* , which is a required parameter for the *command* command.

System action: The command is not processed.

User response: Add the required parameter to the command and retry.

Destination: Message log

CBKxx2831E Unable to locate end of operand for *keyword* parameter. Expected ')' not found.

Explanation: The closing parenthesis is not present at the end of the operand for *keyword*. Instead, the parser has reached the end of the command string

System action: The command is not processed.

User response: This is probably because your command statement does not have the correct number of opening and closing parentheses within the operand value. You can nest open and close parentheses within an operand value, but each open parenthesis must be matched with a closing parenthesis. Check that the number of close parentheses matches the number of open parentheses for the operand.

Destination: Message log

CICS BAC startup messages in CICS region (CBK*xx*4000 —4999)

The following messages are issued by the CICS BAC component running in the CICS region.

CBKxx4001I *applid* **CICS Batch Application Control for z/OS Startup in progress via PLTPI entry.**

Explanation: The CICS component of CICS BAC is starting in CICS region *applid* during PLTPI processing of CBKCMNDS during CICS initialization.

System action: CICS BAC is started in the CICS region.

User response: None.

Destination: Console

CBKxx4002I *applid* **CICS Batch Application Control for z/OS (5697-I94)** *version* **starting.**

Explanation: The CICS component of CICS BAC is beginning startup processing. The CICS BAC version number is shown as *version* in region *applid*.

System action: None.

User response: None.

Destination: Console

CBKxx4003I *applid* **CICS Batch Application Control for z/OS started: TSQname(***queuename***) term/user(***termid/userid***).**

Explanation: The CICS component of CICS BAC has been successfully started in CICS region *applid*. The temporary storage control queue name is given by *queuename*. The *termid/userid* message variable is set as follows:

- If you started CICS BAC by using the KBKM command processor transaction, *termid/userid* identifies the terminal ID and user ID associated with the KBKM start request.
- If you started CICS BAC during CICS PLTPI processing, with a CBKCMNDS entry in the CICS region DFHPLTPI table, the *termid/userid* value is PLTPI.

System action: None.

User response: None.

Destination: Console

CBKxx4004E *applid* **Unable to process command. CICS Batch Application Control for z/OS is not active in this CICS region.**

Explanation: You have issued the CICS BAC command processor (KBKM) transaction with a command other than START, when CICS BAC is not active in CICS region *applid*. The only command that you can specify on KBKM when CICS BAC is not active is the START command.

System action: The command is not processed.

User response: Start CICS BAC and reenter the command as necessary.

Destination: Console

CBKxx4005I *applid* **Dump request complete. Dumpcode:** *dumpcode***.**

Explanation: You have issued the KBKM DUMP command requesting a CICS BAC dump. The dump has completed successfully in region *applid*, with the dump code identified by *dumpcode*.

System action: None.

User response: None.

Destination: Console

CBKxx4006W *applid* **CICS Batch Application Control for z/OS is already started in this region.**

Explanation: A request to start the CICS BAC CICS component has been received, but CICS BAC is already active in CICS region *applid*.

System action: The START request terminates.

User response: None.

Destination: Console

CBKxx4007W *applid* **CICS Batch Application Control for z/OS (5697-I94) is already shut down in this region.**

Explanation: You have issued a request to shut down CICS BAC in CICS region *applid*, but CICS BAC is not active in the region *applid*.

System action: The SHUTDOWN request is rejected.

User response: None.

Destination: Console

CBKxx4008E *applid* **Input data is too long. Maximum 71 characters allowed.**

Explanation: You have issued in CICS region *applid* the CICS BAC KBKM command processor with an input string that is longer than seventy-one characters.

System action: The request is rejected.

User response: Correct the input string and retry the KBKM transaction.

Destination: Console

CBKxx4010I *applid* **CICS Batch Application Control for z/OS SHUTDOWN via PLTSD started.**

Explanation: CICS BAC is shutting down in CICS region *applid* during CICS PLTSD processing. The CICS BAC command processor program, which performs shutdown, is driven by an entry in the PLTSD table.

System action: Shutdown processing begins.

User response: None.

Destination: Console

CBKxx4011I *applid* **CICS Batch Application Control for z/OS SHUTDOWN requested by user** *userid* **from terminal** *termid***.**

Explanation: You have issued the KBKM command processor transaction to shut down CICS BAC in CICS region *applid*. The command processor transaction KBKM was issued by user *userid* at terminal *termid*.

System action: Shutdown processing continues.

User response: None.

Destination: Console

CBKxx4014E *applid* **CICS Batch Application Control for z/OS will not run on this version of CICS. START terminated.**

Explanation: You have tried to start CICS BAC in CICS region *applid* that is running a version of CICS not supported by CICS BAC.

System action: The START request terminates.

User response: Ensure that you are attempting to start CICS BAC in a CICS region that is running a supported version of CICS. If you are, contact technical support.

Destination: Console

CBKxx4015E *applid* **Program ID error: Program(***progname***) code1(***code1***) code2(***code2***).**

Explanation: The CICS component of CICS BAC in region *applid* has received a CICS PGMIDERR error during its startup processing. The name of the program that resulted in the PGMIDERR is *program* and *code1* and *code2* are the EIBRESP and EIBRESP2 values respectively, returned with the PGMIDERR.

System action: The START request is terminated.

User response: This can be caused by a missing or incorrect program definition. It can also be caused by the module being missing. Determine the cause of the problem, correct it, and retry the CICS BAC START command.

Destination: Console

CBKxx4016E *applid* **CICS Batch Application Control for z/OS START request terminated due to subtask initialization failure. See job log for more information.**

Explanation: An unrecoverable error has occurred in the CICS BAC request server subtask during CICS BAC region startup processing in CICS region *applid*.

System action: The START request terminates.

User response: Check the CICS region job log for additional messages describing the error. Correct the problem and retry the START request. If you cannot correct the problem, contact technical support.

Destination: Console

CBKxx4017E *applid* **Error detected during CICS BAC startup. See job log for error information. CICS BAC start request terminated.**

Explanation: An unrecoverable error has occurred during CICS BAC startup processing in CICS region *applid*. CICS BAC writes additional error messages to the CICS job log.

System action: The START request is terminated.

User response: Correct the problem and retry the START request. If you cannot correct the problem, contact technical support.

Destination: Console

CBKxx4019I *applid* **CICS Batch Application Control for z/OS SHUTDOWN complete.**

Explanation: CICS BAC has successfully shut down in CICS region *applid* .

System action: None.

User response: None.

Destination: Console

CBKxx4030I *applid* **The following CICS BAC parameters will be used for this CICS region.**

Explanation: The CICS component of CICS BAC is starting in region *applid* and during its startup processing, CICS BAC displays the parameters that it will use in CICS region *applid*. CBK*xx*4030I is followed by several instances of message CBK*xx*4031I to display the parameters.

System action: CICS BAC startup processing continues.

User response: None.

Destination: Console

CBKxx4031I *applid location parmameter=value*

Explanation: CICS BAC displays one CBK*xx*4031I message for each CICS BAC region startup parameter in region *applid* . *location* can be either CICS or Default. If *location* is CICS, you specified the parameter as part of the command input string. If *location* is Default, you did not specify the parameter explicitly as part of the command input string, and CICS BAC is using the default value. *parameter* is the parameter name and *value* is the value that CICS BAC is to use.

System action: CICS BAC startup continues in the CICS region.

User response: None.

Destination: Console

CBKxx4032I *applid* **End of CICS BAC CICS region startup parameter list.**

Explanation: CICS BAC has finished listing its startup parameters for CICS region *applid*.

System action: CICS BAC startup continues in the CICS region.

User response: None.

Destination: Console

CBKxx4060E *applid* **Error rewriting region properties record during startup: DDN=***ddname* **RESP=***eibresp* **RESP2=***eibresp2***.**

Explanation: An error has occurred during CICS BAC startup processing in CICS region *applid* when CICS BAC was rewriting the CICS region properties record to the CICS BAC control file . The CICS BAC control file name is *ddname*. The error codes *eibresp* and *eibresp2* are the EIBRESP and EIBRESP2 values returned by CICS in response to the rewrite request.

System action: If you are using the CICS BAC command processor transaction, KBKM, to start CICS BAC, the START request terminates. If you are using PLTPI processing to start CICS BAC, CBK*xx*4060E is followed by message CBK*xx*4080R. See the explanation of message CBK*xx*4080R for more information.

User response: If you are using PLTPI processing to start CICS BAC, reply to message CBK*xx*4080R. If you are using the CICS BAC command processor transaction, KBKM, to start CICS BAC, correct the problem and retry the START request. If you cannot resolve the problem, contact technical support.

Destination: Console

CBKxx4070E *applid* **Error closing control file during startup. CICS BAC start request terminated. DDN=***ddname* **RESP=***eibresp* **RESP2=***eibresp2***.**

Explanation: An error has occurred during CICS BAC startup processing in CICS region *applid* when CICS BAC was closing its control file. The CICS BAC control file name is *ddname*. The error codes *eibresp* and *eibresp2* are the EIBRESP and EIBRESP2 values returned by CICS in response to the close request.

System action: If you are using the CICS BAC command processor transaction, KBKM, to start CICS BAC, the START request terminates. If you are using PLTPI processing to start CICS BAC, CBK*xx*4070E is followed by message CBK*xx*4080R. See the explanation of message CBK*xx*4080R for more information.

User response: If you are using PLTPI processing to start CICS BAC, reply to message CBK*xx*4080R. If you are using the CICS BAC command processor transaction, KBKM, to start CICS BAC, correct the problem and retry the START request. If you cannot resolve the problem, contact technical support.

Destination: Console

CBKxx4071E *applid* **Error opening control file during startup. CICS BAC start request terminated. DDN=***ddname* **RESP=***eibresp* **RESP2=***eibresp2***.**

Explanation: An error has occurred during CICS BAC startup processing in CICS region *applid* when CICS BAC was opening its control file for the region. The CICS BAC control file name is *ddname*. The error codes *eibresp* and *eibresp2* are the EIBRESP and EIBRESP2 values returned by CICS in response to the open request.

System action: If you are using the CICS BAC command processor transaction, KBKM, to start CICS BAC, the START request terminates. If you are using PLTPI processing to start CICS BAC, CBK*xx*4071E is followed by message CBK*xx*4080R. See the explanation of message CBK*xx*4080R for more information.

User response: If you are using PLTPI processing to start CICS BAC, reply to message CBK*xx*4080R. If you are using the CICS BAC command processor transaction, KBKM, to start CICS BAC, correct the problem and retry the START request. If you cannot resolve the problem, contact technical support.

Destination: Console

CBKxx4072E *applid* **Error checking control file state during startup. CICS BAC start request terminated. DDN=***ddname* **RESP=***eibresp* **RESP2=***eibresp2***.**

Explanation: An error has occurred during CICS BAC startup processing in CICS region *applid* when CICS

BAC was checking the state of its control file. The CICS BAC control file name is *ddname*. The error codes *eibresp* and *eibresp2* are the EIBRESP and EIBRESP2 values returned by CICS in response to the inquire request.

System action: If you are using the CICS BAC command processor transaction, KBKM, to start CICS BAC, the START request terminates. If you are using PLTPI processing to start CICS BAC, CBK*xx*4072E is followed by message CBK*xx*4080R. See the explanation of message CBK*xx*4080R for more information.

User response: If you are using PLTPI processing to start CICS BAC, reply to message CBK*xx*4080R. If you are using the CICS BAC command processor transaction, KBKM, to start CICS BAC, correct the problem and retry the START request. If you cannot resolve the problem, contact technical support.

Destination: Console

CBKxx4073E *applid* **Invalid control file state after open. CICS BAC start request terminated. DDN=***ddname* **state name=***state-name* **state value=***state-value***.**

Explanation: An error has occurred during CICS BAC startup processing in CICS region *applid* when CICS BAC was checking its control file attributes. The CICS BAC control file name is *ddname*. *state-name* is the name of the attribute that is not correct; *state-value* is the current value of the state for the control file. See *Appendix A* of the *CICS System Programming Reference* for possible values of *state-value*.

System action: If you are using the CICS BAC command processor transaction, KBKM, to start CICS BAC, the START request terminates. If you are using PLTPI processing to start CICS BAC, CBK*xx*4073E is followed by message CBK*xx*4080R. See the explanation of message CBK*xx*4080R for more information.

User response: If you are using PLTPI processing to start CICS BAC, reply to message CBK*xx*4080R. If you are using the CICS BAC command processor transaction, KBKM, to start CICS BAC, correct the problem and retry the START request. If you cannot resolve the problem, contact technical support.

Destination: Console

CBKxx4074E *applid object-type* **object record not found during startup processing. CICS BAC start request terminated.**

Explanation: CICS BAC cannot find one of the required records in its control file during CICS BAC startup processing in CICS region *applid*. The message variable *object-type* can be one of the following values:
- Control file version
- Region properties
- File default
- TDQueue default
- Transid default
- Program default

System action: If you are using the CICS BAC command processor transaction, KBKM, to start CICS BAC, the START request terminates. If you are using PLTPI processing to start CICS BAC, CBK*xx*4074E is followed by message CBK*xx*4080R. See the explanation of message CBK*xx*4080R for more information.

User response: If you are using PLTPI processing to start CICS BAC, reply to message CBK*xx*4080R. If you are using the CICS BAC command processor transaction, KBKM, to start CICS BAC, correct the problem and retry the START request. Note this problem can occur because you have not initialized the CICS BAC control file. If this is the case, run the CICS BAC control file initialization program to initialize the file and then retry the START request. If you cannot resolve the problem, contact technical support.

Destination: Console

CBKxx4075E *applid* **Error reading** *object-type* **object record during startup processing. CICS BAC start request terminated. DDN=***ddname* **RESP=***eibresp* **RESP2=***eibresp2***.**

Explanation: CICS BAC has encountered an error while was reading one of the required records from its control file during CICS BAC startup processing in CICS region *applid*. The message variable *object-type* can be one of the following values:
- Control file version
- Region properties
- File default
- TDQueue default
- Transid default
- Program default

The error codes *eibresp* and *eibresp2* are the EIBRESP and EIBRESP2 values returned by CICS in response to the READ request. The CICS BAC control file name is *ddname*. *eibresp* and *eibresp2* give the EIBRESP and EIBRESP2 values from the read request.

System action: If you are using the CICS BAC command processor transaction, KBKM, to start CICS BAC, the START request terminates. If you are using PLTPI processing to start CICS BAC, CBK*xx*4075E is followed by message CBK*xx*4080R. See the explanation of message CBK*xx*4080R for more information.

User response: If you are using PLTPI processing to start CICS BAC, reply to message CBK*xx*4080R. This error can be caused by a corrupt CICS BAC control file. If you cannot find the cause of the problem, contact technical support.

Destination: Console

CBKxx4076W *applid***Unable to change DSN for JCL allocated file** *ddname***. Current DSN=***current-dsn* **Requested DSN=***requested-dsn*

Explanation: During startup processing in CICS region *applid*, CICS BAC has determined that it should change the data set name associated with a CICS file. However, the data set is permanently allocated to the region because it is defined in the CICS region startup JCL. The data set associated with a file cannot be changed if the file is defined in the JCL. The name of the file being processed is *ddname*. *current-dsn* and *requested-dsn* are the current and requested data set name for the file.

System action: The data set name for the file is not changed. Startup processing continues.

User response: Determine whether or not the data set specified by *current-dsn* should be allocated to the CICS file. If not, make the change by some means other than through a CICS BAC command.

Destination: Console

CBKxx4078E *applid* **Incorrect record length for** *object-type* **object record. CICS BAC start request terminated. Record length=***reclen***.**

Explanation: During CICS BACstartup processing in CICS region *applid*, CICS BAC has read from its control file a required record that has an incorrect record length. The message variable *object-type* can be one of the following values:

- Control file version
- Region properties
- File default
- TDQueue default
- Transid default
- Program default

The CICS BAC control file name is *ddname*, and *reclen* is the length of the object record that CICS BAC read.

System action: If you are using the CICS BAC command processor transaction, KBKM, to start CICS BAC, the START request terminates. If you are using PLTPI processing to start CICS BAC, CBK*xx*4078E is followed by message CBK*xx*4080R. See the explanation of message CBK*xx*4080R for more information.

User response: If you are using PLTPI processing to start CICS BAC, reply to message CBK*xx*4080R. This error can indicate a corrupt CICS BAC control file. Contact technical support.

Destination: Console

CBKxx4079E *applid* **Incorrect control file version record detected: region=***cics-version* **file=***file-version* **DSN=***dsn*

Explanation: During CICS BACstartup processing in CICS region *applid*, CICS BAC has detected an incompatibility between the version of the product and the CICS BAC control file. *cics-version* is the version of the CICS BAC product running in the CICS region. *file-version* specifies the version of the CICS BAC control file. *dsn* is the data set name of the CICS BAC control file.

System action: If you are using the CICS BAC command processor transaction, KBKM, to start CICS BAC, the START request terminates. If you are using PLTPI processing to start CICS BAC, CBK*xx*4079E is followed by message CBK*xx*4080R. See the explanation of message CBK*xx*4080R for more information.

User response: If you are using PLTPI processing to start CICS BAC, reply to message CBK*xx*4080R. If you cannot determine the cause of the incompatibility, contact technical support.

Destination: Console

CBKxx4080R *applid* **Reply GO to continue without CICS BAC or CANCEL to cancel the CICS region.**

Explanation: CICS BAC is waiting in region *applid* for a reply of GO or CANCEL, because of an unrecoverable error detected during CICS BAC startup during CICS PLTPI processing. The error is reported by an earlier error message. If you reply GO, CICS region startup processing continues, but CICS BAC is *not* started in the region. If you reply CANCEL, the CICS region abends.

System action: The CICS BAC startup processing waits for an operator reply to the WTOR.

User response: This message is preceded by one or more CICS BAC messages that describe the error condition. Reply to message CBK*xx*4080R with a GO or CANCEL. If you cannot determine and resolve the cause of the problem, contact technical support.

Destination: Console

CBKxx4081W *applid* **Region startup will continue without CICS BAC support.**

Explanation: The CICS BAC startup processor has received a GO reply to message CBK*xx*4080R or message CBK*xx*4086R.

System action: CICS startup processing continues, but CICS BAC is not active in the region.

User response: None.

Destination: Console

CBKxx4082E *applid* **Region will be abended with abend code** *abend-code***.**

Explanation: The CICS BAC startup processor has received a CANCEL reply to message CBK*xx*4080R or message CBK*xx*4086R.

System action: CICS BAC abnormally terminates CICS region *applid* with abend code *abend-code*.

User response: If you cannot determine and correct the cause of the problem, contact technical support.

Destination: Console

CBKxx4083E *applid* **Error detected during CICS BAC startup processing. Startup terminated. Check system console for additional messages.**

Explanation: CICS BAC has been started in CICS region *applid* by the CICS BAC command processor transaction, KBKM. An unrecoverable error has occurred in the startup processor, causing startup processing to be terminated.

System action: CICS BAC startup processing in the CICS region terminates.

User response: Check the system console or CICS job log for additional messages describing the error condition. If you cannot determine and correct the cause of the problem, contact technical support.

Destination: 3270 terminal that issued the START command

CBKxx4084W *applid* **Unable to obtain CICS BAC region control file enqueue. Startup terminated.**

Explanation: CICS BAC has been started in CICS region *applid* by the CICS BAC command processor transaction, KBKM. However, CICS BAC cannot obtain a systems-level enqueue on its region control file. CICS BAC requires this to ensure that it has exclusive control of the CICS BAC region control file. CICS BAC has tried to obtain the enqueue thirty times, at one second intervals, and has now abandoned the attempt.

System action: CICS BAC terminates the START request.

User response: The enqueue can be held by either a batch request utility job step, or a communication server address space. If you cannot determine and correct the cause of the condition, contact technical support.

Destination: 3270 terminal that issued the START command

CBKxx4085W *applid* **Unable to obtain CICS BAC region control file enqueue.**

Explanation: CICS BAC has been started in CICS region *applid* during PLTPI processing. However, CICS BAC cannot obtain a systems-level enqueue on its region control file. CICS BAC requires this to ensure that it has exclusive control of the CICS BAC region control file. CICS BAC has tried to obtain the enqueue thirty times, at one second intervals, and follows CBK*xx*4085W with message CBK*xx*4086R.

System action: CICS BAC prompts for an operator response to the error with message *xx*4086R.

User response: See message CBK*xx*4086R

Destination: Console

CBKxx4086R *applid* **Reply RETRY, GO to continue without CICS BAC, or CANCEL to cancel the CICS region.**

Explanation: CICS BAC has issued message CBK*xx*4085W and now wants a reply to this RETRY, GO, or CANCEL prompt.

System action: The CICS region waits for an operator reply.

User response: Reply RETRY if you want CICS BAC to continue trying to obtain the enqueue. Reply GO if you want the CICS region to continue startup processing without CICS BAC. Reply CANCEL to terminate the CICS region.

Destination: Console

CBKxx4087I *applid* **Response was RETRY. CICS BAC will try to obtain enqueue again.**

Explanation: CICS BAC has received an operator reply of RETRY to message CBK*xx*4086R.

System action: CICS BAC makes thirty more attempts to obtain its control file enqueue, at one second intervals.

User response: None

Destination: Console

CBKxx4088W Incorrect final state for *resourcetype resourcename***:** *resourcestate***. CICS BAC startup processing continuing.**

Explanation: CICS BAC has attempted to set a resource state during startup processing in the CICS region. The resource type and state are given by *resourcetype resourcename*. Immediately after setting the resource state, CICS BAC performs an inquiry of the resource state to determine whether or not the resource is at the requested state. As a result, CICS BAC has found that the resource is not at the state requested by CICS BAC for at least one state attribute. The first state

discrepancy detected by CICS BAC is given by *resourcestate*. CICS BAC does not check to see if there is more than one discrepancy for a resource.

System action: Startup processing continues. CICS BAC does not attempt to correct the incorrect resource state that it found.

User response: Check the CICS BAC audit log for detailed information on what state changes CICS BAC attempted for the named resource and what the results of the request were. If the final state is not acceptable, change the resource state using some other method, for example, the CICS CEMT transaction.

Destination: Console

CBKxx4092W *applid* **Error detected while processing** *resourcetype*: *resourcename errorcode1 errorcode2 errordata*. **See CICS BAC audit log for more information.**

Explanation: CICS BAC has detected an error during its startup processing in the CICS region. Various types of diagnostic data are provided in the message. Additional diagnostic data might also have been written to the CICS BAC audit log.

System action: Processing is terminated for the resource, and continues with the next resource on the CICS BAC control file.

User response: Contact technical support if you cannot determine the reason for the error and correct it.

Destination: Audit log

CBKxx4093W *applid errorcount* **errors detected during CICS BAC startup processing for** *resourcetype*. **See audit log for more information.**

Explanation: In CICS region *applid*, CICS BAC has detected one or more errors (*errorcount*) during CICS BAC region startup processing for resource type *resourcetype*.

System action: Processing continues.

User response: Contact technical support if you cannot determine the reason for the errors and correct them.

Destination: Audit log

CBKxx4700I *applid* **CICS request handler subtask initialization starting.**

Explanation: The CICS BAC request handler subtask is being started in CICS region *applid*.

System action: CICS BAC startup processing continues in the CICS region.

User response: None.

Destination: Console

CBKxx4701I *applid* **CICS request handler subtask initialization complete. Now accepting requests.**

Explanation: The CICS BAC request handler subtask has completed its initialization processing in CICS region *applid* and is ready to begin accepting requests from a CICS BAC batch request utility and communication server.

System action: CICS BAC startup processing continues in the CICS region.

User response: None.

Destination: Console

CBKxx4702I *applid* **CICS request handler shutting down. Batch utility requests no longer accepted.**

Explanation: The CICS BAC request handler subtask running in CICS region *applid* has received a shutdown request. It no longer accepts requests from a CICS BAC batch request utility or communication server.

System action: CICS BAC shutdown processing continues.

User response: None.

Destination: Console

CBKxx4707E *applid* **CICS request handler unable to dynamically allocate control file. Request handler terminating.**

Explanation: During startup processing in CICS region *applid*, the CICS BAC request handler subtask has detected an unrecoverable error when attempting dynamically to allocate the CICS BAC control file for the region. Additional messages describing the error can be displayed before this message.

System action: CICS BAC startup processing in the CICS region terminates.

User response: If you cannot determine and correct cause of the problem, contact technical support

Destination: Console

Batch request utility messages (CBK*xx*5000–5999)

The following messages are issued by the batch request utility:

CBKxx5001E Required DDNAME CBKPRINT is missing. Batch request utility terminating.

Explanation: The CBKPRINT DDNAME is missing from the batch request utility JCL. The batch request utility cannot run without the CBKPRINT data set.

System action: The batch request utility terminates.

User response: Add the CBKPRINT DD statement to the batch request utility job step and resubmit the job.

Destination: Message log

CBKxx5002E Open failed for the CBKPRINT DDNAME. Batch request utility terminating.

Explanation: The batch request utility has failed to open the CBKPRINT DDNAME data set, which is required for the batch request utility to run.

System action: The batch request utility terminates.

User response: Check the job log for additional MVS messages relating to this error, take the corrective action recommended by those messages, and resubmit the job.

Destination: Message log

CBKxx5003E Required DDNAME CBKIN is missing. Batch request utility terminating.

Explanation: The CBKIN DDNAME is missing from the batch request utility job step. The batch request utility cannot run without the CBKIN data set.

System action: The batch request utility terminates.

User response: Add the CBKIN DD statement to the batch request utility job step and then resubmit the job.

Destination: Message log.

CBKxx5004E Open failed for the CBKIN DDNAME. Batch request utility terminating.

Explanation: The batch request utility has failed to open the CBKIN data set, which is required in order for the batch request utility to run.

System action: The batch request utility terminates.

User response: Check the job log for additional MVS messages relating to the error, take the corrective action recommended by those messages, and resubmit the job.

Destination: Message log

CBKxx5005E Insufficient storage exists to process request. Batch request utility terminating.

Explanation: The batch request utility is unable to obtain the virtual storage needed to process its input requests.

System action: The batch request utility terminates.

User response: Check the REGION parameter for the batch request utility job step. The REGION parameter specifies how much virtual storage is allocated for the batch request utility. If possible, increase the value to make additional virtual storage available to the batch request utility. If it is not possible to increase this value, examine the input statements to the batch request utility to determine the best way to break the current job step into multiple job steps.

Destination: Message log

CBKxx5006E Request(s) not processed due to error(s) with one or more statements.

Explanation: At least one input statement has an error that prevents the statement from being executed. When this condition occurs, the batch request utility does not execute any of the statements.

System action: The batch request utility terminates without executing any of the input statements.

User response: Examine the Message log to determine which statements have errors. Correct those errors and resubmit the JCL.

Destination: Message log.

CBKxx5007E Error processing request: EIBRESP=*resp1*, EIBRESP2=*resp2*, VSAM CODE=*vsamcode*, Type=*objecttype*, Object=*object*, Text=*errortext*.

Explanation: An error has occurred while processing the request. This error prevents the request from completing successfully. The *object* and *objecttype* fields indicate the object and object type on which the error occurred. The *errortext* field, if any, provides additional information regarding the nature of the error. If the error occurs while accessing the CICS region control file, the *vsamcode* field is non-zero. If the error occurs while processing a CICS API command, *resp1* and *resp2* contain the CICS EIB response codes for the command.

System action: The current request is not successfully completed. All subsequent requests are skipped and the batch request utility terminates.

User response: Contact technical support.

Destination: Message log

CBKxx5008E Statement was not processed due to error(s) with one or more statements.

Explanation: This command is not executed due to an error in one or more previous statements.

System action: The current command is not executed.

User response: Examine the message log for any errors prior to the current command. Correct those errors and resubmit the job step.

Destination: Message log

CBKxx5009I Statement was successfully processed.

Explanation: The batch request utility has executed the current statement without any errors. If the request was passed to a target CICS region for execution, it means also that the CICS BAC request server has executed the command successfully.

System action: The batch request utility proceeds on to the next input statement.

User response: None.

Destination: Message log

CBKxx5010I Default values updated.

Explanation: The DEFAULT statement has been successfully processed. The DEFAULT values for subsequent statements have been updated.

System action: The batch request utility proceeds on to the next input statement using the applicable parameter values from this DEFAULT statement.

User response: None.

Destination: Message log

CBKxx5011I Returned data area contents:

Explanation: A LINK PROGRAM or RUNCEMT command has been successfully completed and the data returned from that request is formatted in a standard dump format. Note that the returned data is dependent upon the program or CEMT command being executed. If more than 256 bytes of data are returned, only the first 256 bytes are dumped to the message log.

System action: The returned data from that request is formatted in a standard dump format and written to CBKPRINT.

User response: None.

Destination: Message log

CBKxx5012I Parameter values in effect for the *command* command:

Explanation: The batch request utility is about to display the parameter definitions for the current command. The utility displays each parameter in a separate CBK*xx*5013I message, showing the values in effect for the command. If the command contains no parameters, the utility does not display any CBK*xx*5013 messages.

System action: None.

User response: None.

Destination: Message log

CBKxx5013I *keyword*(*value*).

Explanation: The batch utility displays a parameter and value pair, *keyword* (*value*) of the current command it is processing.

System action: None.

User response: Ensure that the parameter and its value are correct for the current statement. Note that the source of the parameter might be the current statement itself or the most recent DEFAULT statement.

Destination: Message log

CBKxx5014W Request completed with warnings: EIBRESP=*resp1*, EIBRESP2=*resp2*, VSAM CODE=*vsamcode*, Type=*objecttype*, Object=*object*, Text=*errortext*.

Explanation: A problem has occurred while processing the request that might have prevented the execution of the command from having the required results. The *object* and *objecttype* fields indicate the object and object type on which the problem occurred. The *errortext* field, if any, provides additional information regarding the nature of the problem. If the problem has occurred while accessing the CICS region control file, then the *vsamcode* field is non-zero. If the problem has occurred while processing a CICS API command, *resp1* and *resp2* contains the CICS EIB response codes for the command.

System action: The current request might not have completed successfully. The batch request utility continues on to the next request.

User response: Contact technical support.

Destination: Message log

CBKxx5015E *jobname* Empty or no control file table located. Batch request utility abending with abend code U1503.

Explanation: The batch request utility has to directly update a CICS region control file because CICS BAC is not active in the CICS region. However, the batch

request utility doesn't know the name of the control file data set because it cannot locate a control file table, or the control file table it located is empty.

System action: The batch request utility job step abends with abend code U1503.

User response: The batch request utility requires a control file table containing an entry for the target CICS region to find the data set name for the target region control file. For more information about the control file table, see "The control file table" on page 23. Ensure that a control file table that contains an entry for the target CICS region is available to the batch request utility job step, and retry the job step.

Destination: Job log

CBKxx5016E Dynamic allocation for CBKLOG failed. Unable to process request(s).

Explanation: The batch request utility is unable to allocate dynamically the CICS BAC audit log. This message can be preceded by one or more messages that provide additional information relating to the dynamic allocation failure.

System action: The batch request utility job step terminates with condition code 12.

User response: If you cannot determine and correct the cause of the dynamic allocation failure, contact technical support.

Destination: Job log

CBKxx5017E Open failed for CBKLOG file. Unable to process request(s).

Explanation: The batch request utility is unable to open the CICS BAC audit log. This message can be preceded by one or more messages that provide additional information relating to the open failure.

System action: The batch request utility job step terminates with condition code 12.

User response: If you cannot determine and correct the cause of the open failure, contact technical support.

Destination: Job log

CBKxx5018E Unable to audit due to lack of required CBKLOG file.

Explanation: The batch request utility needs to log requests, but there is no CBKLOG data set DD statement in the job step JCL, and the options that enable the batch request utility to allocate dynamically the audit log are not specified.

System action: The batch request utility job step terminates with condition code 12

User response: Define an audit log data set in the batch request utility job step JCL or change the appropriate CICS region properties record to enable the audit log to be allocated dynamically.

Destination: Job log

CBKxx5019E Invalid control file detected. Job step abending with U1502 abend.

Explanation: The batch request utility needs to update the target CICS region control file directly because CICS BAC is not active in the region. However, the batch request utility detected an invalid control file during its control file validation processing. Messages providing additional information about the problem with the control file can precede this message.

System action: The batch request utility job step abends with abend code U1502.

User response: If you cannot determine and correct the cause of the problem, contact technical support.

Destination: Job log

CBKxx5020E Unable to load CICS EXCI module. CICS SDFHEXCI load library may not be available: code1(*retcode*) code2(*rsncode*)

Explanation: During batch request utility initialization, CICS BAC has attempted to load the CICS EXCI module DFHXCPRX to check that the CICS SDFHEXCI load library is available. The load request has failed with the errors indicated by *retcode* and *rsncode*, which contain the return code and reason code from the failed load attempt.

System action: The batch request utility job step terminates with condition code 12.

User response: Ensure that the CICS SDFHEXCI load library is available to the batch request utility by adding it to the MVS linklist, or to the JOBLIB or STEPLIB concatenation in the batch request utility JCL.

Destination: Job log

CBKxx5021I The following commands were serviced by the CICS BAC request server in CICS region *applid*:

Explanation: The batch request utility has received confirmation that the commands it passed to the CICS BAC request server for CICS region *applid* have been processed. The commands are listed in the message log immediately following the CBK*xx*5021I message.

System action: The batch request utility lists the commands that have been processed by the request server.

User response: None.

Destination: Message log

CBKxx5022I The following commands were serviced directly by the CICS BAC batch request utility because CICS region *applid* is not active or the CICS BAC request server is not active in that region:

Explanation: The batch request utility is unable to send commands to CICS region *applid*, because the region is not active, or because CICS BAC is not active in the region. The batch request utility has serviced the listed commands directly itself. The commands are listed in the message log immediately following the CBK*xx*5022I message.

System action: The batch request utility lists the commands that it has serviced.

User response: None.

Destination: Message log

CBKxx5023W Batch request utility detected CICS region *applid* did not go through normal CICS BAC shutdown processing.

Explanation: The batch request utility has detected that CICS BAC is not active in CICS region *applid* (or the CICS region is not available), and that CICS BAC did not go through its normal termination the last time it was active in the region. The region properties for CICS region *applid* specify that the batch request utility should wait for an operator reply to determine how to proceed.

System action: The batch request utility issues message CBK*xx*5024R to prompt for a reply to this message.

User response: See message CBK*xx*5024R.

Destination: Console

CBKxx5024R Reply GO to continue processing or ABEND to terminate job step.

Explanation: The batch request utility has issued message CBK*xx*5023W and now prompts for a answer on how to proceed.

System action: The batch request utility waits for a GO or ABEND reply.

User response: See message CBK*xx*5023W for a an explanation and reply accordingly

Destination: Console

CBKxx5025W Reply was GO. Batch request utility will continue processing.

Explanation: The batch request utility has received a GO reply to message CBK*xx*5024R.

System action: The batch request utility job step continues.

User response: None.

Destination: Console

CBKxx5026E Reply was ABEND. Batch request utility will be abended with abend code *abend-code*.

Explanation: The batch request utility has received an ABEND reply to message CBK*xx*5024R.

System action: The batch request utility job step abends with user abend code *abend-code*. No additional commands are processed.

User response: Depending on the situation, you might need to restart CICS BAC in the CICS region and then shut it down to reset the shutdown flag.

Destination: Console

CBKxx5027E Batch request utility abending. Target CICS region *applid* did not go through CICS BAC shutdown processing.

Explanation: CICS BAC did not go through normal shutdown processing the last time it was active in CICS region *applid*. The region properties for the target region specify that a batch request utility job step should be abended when this condition occurs.

System action: The batch request utility job step abends with the abend code specified in the target CICS region properties record.

User response: Depending on the situation, you might need to restart CICS BAC in the CICS region and then shut it down to reset the shutdown flag.

Destination: Message log

CBKxx5028E Batch request utility terminating. Target CICS region *applid* did not go through CICS BAC shutdown processing.

Explanation: CICS BAC did not go through normal shutdown processing the last time it was active in CICS region *applid*. The region properties for the target region specify that a batch request utility job step should be terminated when this condition occurs.

System action: The batch request utility job step terminates with the condition code specified in the target CICS region properties record.

User response: Depending on the situation, you might need to restart CICS BAC in the CICS region and then shut it down to reset the shutdown flag.

Destination: Message log

CBKxx5032E Authorization check failed, RACROUTE RC=*racroute*, RACF RC=*rc*, RACF REASON=*reason*,

Explanation: The batch request utility is unable to issue a command because the current user ID is not authorized. See the *z/OS Security Server RACROUTE Macro Reference* manual, SA22-7692, for an explanation of the return and reason codes.

System action: The current request has not successfully completed. All subsequent requests are skipped and the batch request utility terminates.

User response: Contact your RACF administrator if you believe the batch request utility user ID should be authorized to issue the command.

Destination: CBKPRINT file.

Region control file initialization messages (CBK*xx*6000–6099)

The following messages are issued by the region control file initialization utility:

CBKxx6000I CICS region control file initialization starting.

Explanation: The CICS BAC region control file initialization utility is starting.

System action: None.

User response: None.

Destination: Job log

CBKxx6001I CICS region applid: *applid*.

Explanation: The CICS region applid for the control file is *applid*.

System action: None.

User response: None.

Destination: Job log

CBKxx6002I Version record successfully added: *version*.

Explanation: The region control file initialization utility has successfully added the CICS BAC version record to the control file it is initializing. The version number is *version*.

System action: None.

User response: None.

Destination: Job log

CBKxx6003I CICS region properties record successfully added.

Explanation: The region control file initialization utility has successfully added the CICS BAC region properties record to the control file it is initializing. The record contains default values.

System action: None.

User response: None.

Destination: Job log

CBKxx6004I *object-type* default record successfully added.

Explanation: The region control file initialization utility has successfully added a CICS BAC default object record for *object-type* to the control file it is initializing. The *object-type* record contains default values. *object-type* can be on of the following:

- File
- Program
- TDQueue
- Transid

System action: None.

User response: None.

Destination: Job log

CBKxx6051E Error *action* control file. Initialization terminating. code1=*retcode* code2=*rsncode*

Explanation: An error has occurred in the region control file initialization utility while it was opening or closing a region control file. *retcode* and*rsncode* are the return and reason codes for the open or close request.

System action: The control file initialization utility abends with abend code U1866 if the error occurred on an open request, or U1867 if the error occurred on a close request.

User response: Check the file definition attributes to ensure that you have defined the control file correctly, delete and redefine the control file, then and rerun the initialization utility. If you cannot determine the cause of the problem, contact technical support.

Destination: Job log

CBKxx6052E Invalid or no CICS applid provided via PARM on EXEC JCL statement. Initialization terminating.

Explanation: The region control file initialization utility is attempting to initialize a region control file, but you have not specified a CICS region applid in the PARM string of the EXEC JCL statement.

System action: The control file initialization utility

abends with abend code U1868.

User response: Specify the missing CICS applid in the PARM string of the EXEC JCL statement then rerun the control file initialization utility.

Destination: Job log

CBKxx6060E Error adding version record. Initialization terminating. code1=*code1* **code2=***fdbkword*

Explanation: An error has occurred in the region control file initialization utility when attempting to add the control file version record. *code1* is the internal use error code and*fdbkword* is the VSAM feedback word for the request.

System action: The control file initialization utility abends with abend code U1860.

User response: Contact technical support if you cannot diagnose and correct the problem.

Destination: Job log

CBKxx6061E Error adding CICS region properties record. Initialization terminating. code1=*code1* **code2=***fdbkword*

Explanation: An error has occurred in the region control file initialization utility when attempting to add a CICS region properties record. *code1* is the internal use error code and*fdbkword* is the VSAM feedback word for the request.

System action: The control file initialization utility abends with abend code U1861.

User response: Contact technical support if you cannot diagnose and correct the problem.

Destination: Job log

CBKxx6062E Error adding ***object-type*** **default record. Initialization terminating. code1=***code1* **code2=***fdbkword*

Explanation: An error has occurred in the region control file initialization utility when attempting to add a default record for *object-type*. *code1* is the internal use error code and*fdbkword* is the VSAM feedback word for the request.

System action: The control file initialization utility abends with abend code U1862.

User response: Contact technical support if you cannot diagnose and correct the problem.

Destination: Job log

CBKxx6099I CICS region control file initialization completed successfully.

Explanation: The CICS BAC region control file initialization utility has successfully initialized a CICS region control file.

System action: None.

User response: None.

Destination: Job log

File maintenance utility messages (CBK*xx*7000–7999)

CBKxx7001E Required DDNAME CBKPRINT is missing. File maintenance utility terminating.

Explanation: The CBKPRINT DDNAME is missing from the file maintenance utility JCL. The file maintenance utility cannot run without the CBKPRINT data set.

System action: The file maintenance utility job step terminates with return code 12.

User response: Add the CBKPRINT DD statement to the file maintenance utility job step and resubmit the job.

Destination: Job log.

CBKxx7002E Open failed for the CBKPRINT DDNAME. File maintenance utility terminating.

Explanation: The file maintenance utility has failed to open the CBKPRINT DDNAME data set, which is a required data set for the file maintenance utility.

System action: The file maintenance utility job step terminates with return code 12.

User response: Check the job log for additional MVS messages relating to this error, take the corrective action recommended by those messages, and resubmit the job.

Destination: Job log.

CBKxx7003E Required DDNAME CBKIN is missing. File maintenance utility terminating.

Explanation: The CBKIN DDNAME is missing from the file maintenance utility JCL. The file maintenance utility cannot run without the CBKIN data set.

System action: The file maintenance utility job step terminates with return code 12.

User response: Add the CBKIN DD statement to the file maintenance utility job step and resubmit the job.

Destination: Message log.

CBKxx7004E Open failed for the CBKIN DDNAME. File maintenance utility terminating.

Explanation: The file maintenance utility has failed to open the CBKIN DDNAME data set, which is a required data set for the file maintenance utility.

System action: The file maintenance utility job step terminates with return code 12.

User response: Check the job log for additional MVS messages relating to this error, take the corrective action recommended by those messages, and resubmit the job.

Destination: Message log.

CBKxx7005E Insufficient storage exists to process request. File maintenance utility terminating.

Explanation: The file maintenance utility is unable to obtain the virtual storage it needs to process its input requests.

System action: The file maintenance utility job step terminates with return code 12.

User response: Check the REGION parameter for the file maintenance utility job step. If possible, increase the region size to make additional virtual storage available to the file maintenance utility. If it is not possible to increase this value, examine the input statements to the file maintenance utility to determine the best way to break up the current job step into multiple job steps.

Destination: Message log.

CBKxx7006E No input cards provided in the CBKIN data set. File maintenance utility terminating.

Explanation: The file maintenance utility has not found any input data in the CBKIN data set.

System action: The file maintenance utility job step terminates with return code 8.

User response: Provide some valid input to the file maintenance utility and resubmit the job.

Destination: Message log.

CBKxx7007E No commands provided the CBKIN data set. File maintenance utility terminating.

Explanation: The file maintenance utility has not found any commands in the CBKIN data set. Possibly all the input statements are comments or blank lines.

System action: The file maintenance utility job step terminates with return code 8.

User response: Provide some valid commands to the file maintenance utility and resubmit the job.

Destination: Message log.

CBKxx7008E CBKCNTL DD card is missing and APPLID was not specified on the parm. File maintenance utility terminating.

Explanation: The file maintenance utility cannot identify the CICS region for which the input commands are intended. The file maintenance utility uses a

combination of the control file table and one of either the CBKCNTL DD statement or the named applid from the PARM string on the JCL EXEC statement. These are both missing.

System action: The file maintenance utility job step terminates with return code 12.

User response: Provide either a CBKCNTL DD statement pointing to a CICS BAC VSAM control file, or specify the CICS APPLID on the PARM string of the EXEC statement in the JCL. For information about identifying a CICS region, see Chapter 7, “CICS BAC file maintenance utility,” on page 67

Destination: Message log.

CBKxx7009I Requests will be serviced in *mode* mode. CICS applid is *applid*.

Explanation: The file maintenance utility services commands either in CICS mode or NATIVE mode. In CICS mode, the requests are shipped to the CICS region identified by *applid* and the VSAM control file is updated in that CICS region.

In NATIVE mode the requests are processed directly in the file maintenance utility batch job's region.

System action: Processing continues.

User response: None.

Destination: Message log.

CBKxx7010E Both the parameter APPLID(...) and the CBKCNTL DD card are present. File maintenance utility terminating.

Explanation: The file maintenance utility needs to know the identity of the target CICS region and the data set name of the CICS region control file. The file maintenance utility uses a combination of the control file table *and* one of the following to identify the target CICS region and the control file data set name:

- The CICS region applid from the PARM string of the EXEC statement
- The CICS BAC control file data set name as specified on the CBKCNTL DD statement.

With one of these pieces of information, the utility can look up either the CICS region applid or the CICS region control file. The utility has found that both items are present, which is an error. For more information about the use of the APPLID runtime option and the CBKCNTL DD statement, see “File maintenance utility runtime parameters” on page 68

System action: The file maintenance utility job step terminates with return code 12.

User response: Provide only one of the following:

- A CBKCNTL DD statement pointing to a CICS BAC VSAM control file
- The CICS applid on the PARM string of the EXEC statement.

For information about identifying a CICS region, seeChapter 7, “CICS BAC file maintenance utility,” on page 67

Destination: Message log.

CBKxx7011E Unable to load CICS EXCI module. CICS SDFHEXCI load library may not be available: code1(*retcode*) code2(*rsncode*)

Explanation: During its initialization, the CICS BAC file maintenance utility has attempted to load the CICS EXCI module DFHXCPRX to check that the CICS SDFHEXCI load library is available. The load request has failed with the errors indicated by *retcode* and *rsncode*, which give the return code and reason code from the failed load attempt.

System action: The file maintenance utility job step terminates with return code 12.

User response: Ensure that the CICS SDFHEXCI load library is available to the batch request utility by adding it to the MVS linklist, the JOBLIB, or the STEPLIB concatenation in the file maintenance utility JCL.

Destination: Message log.

CBKxx7012E Error detected in input parameter from *parmloc*. Processing terminating.

Explanation: The file maintenance utility has detected an error in an input parameter. In the message text, *parmloc* specifies the location from which the parameter was read, and which can be one of the following values:

- MVS PARMLIB
- CBKPARMS member CBKMAINT
- EXEC PARM

This message is preceded by a message giving the details of the parameter error.

System action: The file maintenance utility job step terminates with return code 12.

User response: Correct the parameter error and resubmit the job.

Destination: Message log.

CBKxx7100E *jobname* Empty or no control file table located. Processing terminated.

Explanation: The file maintenance utility can't find the name of the control file data set because it cannot locate a control file table, or the control file table it located is empty. The file maintenance utility needs to know which CICS region is to service requests and to do this it needs to reference the control file table.

System action: The file maintenance utility job step terminates with return code 12.

User response: The file maintenance utility requires a control file table containing an entry for the target CICS region to find the data set name for the target region control file. For more information about the control file table, see "The control file table" on page 23. Ensure that a control file table that contains an entry for the target CICS region is available to the file maintenance utility job step, and retry the job step.

Destination: Message log

CBKxx7101I *jobname*. **File maintenance utility initialization complete: MAB=***mab*.

Explanation: File maintenance utility initialization is complete and the control card input is now being processed. The name of the batch job running the file maintenance utility is *jobname*, and *mab* is the address of an internal control block used in diagnosing problems.

System action: Processing continues.

User response: None.

Destination: Message log.

CBKxx7102E Function *function* **failed. RC=***retcode* **RS=***rsncode*.

Explanation: An unexpected internal error has occurred while processing. *function* is the name of the function in which the failure occurred. The return code and reason code from the function invoked are *retcode* and *rsncode* respectively.

System action: The file maintenance utility job step terminates with return code 16. By default, a return code of 16 or higher produces an abend U3502.

User response: Contact technical support if you cannot diagnose and correct the problem.

Destination: Message log.

CBKxx7200E *objtype objname* **already imbedded in control record.**

Explanation: While processing a command that is adding an object to a list of objects, the file maintenance utility has found that the object already exists. *objtype* and *objname*

System action: The current command processing is terminated with return code 8.

User response: Correct the input to obtain the desired result.

Destination: Message log.

CBKxx7201E *objtype objname* **request failed due to bad internal type** *objtypeshort*.

Explanation: The file maintenance utility is processing a command, but an internal error has occurred.

System action: The current command processing is terminated with return code 16.

User response: Contact technical support.

Destination: Message log.

CBKxx7202E *objtype objname* **has too many imbedded objects. Object count=***objmax*.

Explanation: The file maintenance utility is attempting to add an object named *objname* of object type *objtype* but the object being added contains more than the maximum allowed number *objmax* of imbedded objects. This error can apply to the following resource types:

- APPGROUP object records, where the imbedded objects can be file, TD queue, transaction ID, and program resource names.
- APPLIST object records, where the imbedded objects can be application group names.
- FILE and TDQUEUE object records, where the imbedded objects can be transaction ID, and program resource names.

System action: The current command processing is terminated with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7203E More objects in remove list than exist in *objtype objname* **on Control file.** *objcnt* **imbedded objects exist.**

Explanation: The file maintenance utility is attempting to update the object named *objname* to remove one or more objects from its object list. However, the utility but has found that there are more objects listed for removal in the update command than actually exist in the object record on the CICS region control file. *objcnt* is the count of objects that actually exist in the object list.

System action: The current command processing is terminated with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7204E No *objtype* **objects removed from** *objname*. **Update not done.**

Explanation: The file maintenance utility is attempting to update the object named *objname* to remove one or more objects from its object list, but has found that the

requested objects of *objtype* are not present in the object list.

System action: The current command processing is terminated with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7205E *objtype objname* **already exists on control file. Object not added.**

Explanation: The object type *objtype* named *objname* already exists on the control file, therefore this object cannot be added.

System action: The current command processing is terminated with command return code 8.

User response: Use the REPLACE(YES) option on the command if you want the object replaced instead of added.

Destination: Message log.

CBKxx7206E *objtype objtype* **does not exist on control file.**

Explanation: The file maintenance utility cannot find the object type *objtype* named *objname* on the control file.

System action: The current command processing is terminated with command return code 8.

User response: Correct the input and resubmit the command.

Destination: Message log.

CBKxx7207I *objtype objname* **has been added to the control file.**

Explanation: The file maintenance utility has added object type *objtype* named *objname* to the control file.

System action: Processing continues with command return code 0.

User response: None.

Destination: Message log.

CBKxx7208I *objtype objname* **has been replaced on the control file.**

Explanation: The file maintenance utility has replaced object type *objtype* named *objname* on the control file.

System action: Processing continues with command return code 0.

User response: None.

Destination: Message log.

CBKxx7209I *objtype objname* **has been updated on the control file.**

Explanation: The file maintenance utility has updated object type *objtype* named *objname* on the control file.

System action: Processing continues with command return code 0.

User response: None.

Destination: Message log.

CBKxx7210E Deletion of the *objname* **$Default record is not permitted.**

Explanation: The file maintenance utility does not permit $Default record of object name *objname* to be deleted because it is required by CICS BAC. You cannot delete any $Default records.

System action: The current command processing terminates with command return code 8.

User response: Remove the DELETE statement and resubmit the job.

Destination: Message log.

CBKxx7211I *objtype objname* **has been deleted from the control file.**

Explanation: The file maintenance utility has deleted object *objname* of object type *objtype* from the control file.

System action: Processing continues with command return code 0.

User response: None.

Destination: Message log.

CBKxx7212I *objcnt objtype* **objects deleted.**

Explanation: The file maintenance utility has deleted multiple objects of type *objtype*. The number of objects deleted is *objcnt*

System action: Processing continues with command return code 0.

User response: None.

Destination: Message log.

CBKxx7213E *setvar* **has been set to** *setvalue***.**

Explanation: The file maintenance utility has processed a SET command for the condition code *setvar* and has set the requested value to *setvalue*.

System action: Processing continues with command return code 0.

User response: None.

Destination: Message log.

CBKxx7214W The following control statements not processed due to *errtype* errors above.

Explanation: The file maintenance utility has found errors of type *errtype* and has flushed subsequent commands. CBK*xx*7214W precedes a list of all command input that has not been processed because of the error.

System action: Command processing is terminated.

User response: Review the messages associated with the command in error, take corrective action, and resubmit the job.

Destination: Message log.

CBKxx7215E The *parm1* parameter requires the *parm2* parameter to be set to *value1*.

Explanation: The file maintenance utility has found that parameter *parm1* is present in the command, but the parameter *parm2* is not present, or is present with an incorrect value. Parameter *parm2* is required with *value1* on this command.

System action: Command processing is terminated with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7216E The *parm1*(*value1*) parameter requires the *parm2*. parameter to be specified.

Explanation: The file maintenance utility has found that parameter *parm1*, with a value of *value1* is present in the command. However, parameter *parm2* is not present and is required.

System action: Command processing is terminated with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7217E The *parm1*(*value1*) is mutually exclusive with the *parm2* parameter.

Explanation: The file maintenance utility has found that parameter *parm1*, with a value of *value1* is present in the command, together with parameter *parm2*. However, these two parameters are mutually exclusive and you cannot specify them both on the same command.

System action: Command processing is terminated with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7218E *objcnt objtype* objects not listed due to security.

Explanation: The file maintenance utility is processing a LIST command but has found that access to objects of type *objtype* is denied by RACF. This is because the file maintenance utility user ID does not have sufficient authority for this command and object type. The number of objects not listed is *objcnt*.

System action: The current command processing terminates with command return code 4.

User response: Consult your security administrator to obtain the required access authority to the objects for which permission has been refused.

Destination: Message log.

CBKxx7219E *objcnt objtype* objects deleted, *objdel objtype* objects not deleted due to security.

Explanation: The file maintenance utility is processing a DELETE command but has found that access to some objects of type *objtype* is denied by RACF. This is because the file maintenance utility user ID does not have sufficient authority for all the objects of this object type.

The number of objects successfully deleted is *objcnt*; *objdel* is the number of objects of *objtype* not deleted.

System action: The current command processing terminates with command return code 4.

User response: Consult your security administrator to obtain the required access authority to the objects for which permission has been refused.

Destination: Message log.

CBKxx7300E Access to the control file is denied. File maintenance utility terminating.

Explanation: Before shipping a command to the CICS region for processing, the file maintenance utility has called the MVS external security manager (ESM) to check that the user ID running the file maintenance utility batch job has the required access to the CICS region control file. The user ID on the batch job for this run of the file maintenance utility does not have the required update authority for the region control file.

System action: Command processing is terminated with command return code 12.

User response: Review access restrictions to the VSAM region control file for the user ID under which the batch job is executing, and if appropriate grant update authority to the user ID and resubmit the job.

Destination: Message log.

CBKxx7302E Requested *objtype objname* not found on control file.

Explanation: The file maintenance utility, during command processing, cannot find the requested *objtype* named *objname*, because it does not exist the CICS region VSAM KSDS control file.

System action: The file maintenance utility terminates with return code 8, unless you have used a SET *xxxx*CC command for the failing command to vary the default condition code to a value that permits processing to continue. For more information, see the "SET" on page 92 command on page "SET" on page 92.

User response: Correct the input and, if the job has terminated, resubmit the job.

Destination: Message log.

CBKxx7305E Internal error on *funcname* call. Control block ID is not CRH.

Explanation: During command processing in the file maintenance utility, an internal error has occurred when invoking the *funcname* function.

System action: Command processing is terminated with command return code 16.

User response: Contact technical support if you cannot diagnose and correct the problem.

Destination: Message log.

CBKxx7306E Internal error on *funcname* call. Product name is invalid.

Explanation: During command processing in the file maintenance utility, an internal error has occurred when invoking the *funcname* function.

System action: Command processing is terminated with command return code 16.

User response: Contact technical support if you cannot diagnose and correct the problem.

Destination: Message log.

CBKxx7308E Control file does not exist.

Explanation: The file maintenance utility is attempting to process a command but cannot find the CICS BAC VSAM CICS region control file.

System action: Command processing is terminated with command return code 12.

User response: Contact technical support if you cannot diagnose and correct the problem.

Destination: Message log.

CBKxx7309E Request version is not valid.

Explanation: The file maintenance utility is attempting to process a command but has found that the internal version ID is incorrect.

System action: Command processing is terminated with command return code 12.

User response: Contact technical support if you cannot diagnose and correct the problem.

Destination: Message log.

CBKxx7310E Download buffer is too small.

Explanation: The file maintenance utility is processing a command but has found that the amount of data to be transferred is too large for the specified download buffer size.

System action: Command processing is terminated with command return code 4.

User response: Either refine your selection criteria or increase the download buffer size using the MAXDOWNLOADPAGES runtime parameter. If your corrective actions still do not solve the problem, contact technical support.

Destination: Message log.

CBKxx7311E Too many browse inquiry entries requested.

Explanation: The file maintenance utility is processing a command but has found that the number of requested records is too large.

System action: Command processing is terminated with command return code 12.

User response: Contact technical support if you cannot diagnose and correct the problem.

Destination: Message log.

CBKxx7312E CICS is not active for this request.

Explanation: The file maintenance utility is processing a command but has found that the CICS region has became inactive or unavailable for some reason.

System action: Command processing is terminated with command return code 12.

User response: Review the logs from the target CICS region for any additional information. Contact technical support if you cannot diagnose and correct the problem.

Destination: Message log.

CBKxx7313E Control file is invalid.

Explanation: The file maintenance utility is attempting to process a command but has found that the control file is invalid.

System action: Command processing is terminated with command return code 12.

User response: Contact technical support if you cannot diagnose and correct the problem.

Destination: Message log.

CBKxx7314E CICS region is not defined.

Explanation: The target CICS region is undefined.

System action: Command processing is terminated with command return code 12.

User response: Contact technical support if you cannot diagnose and correct the problem.

Destination: Message log.

CBKxx7315E *objtype objname* **is invalid. Object skipped.**

Explanation: The file maintenance utility is processing a command but has found that the object *objname* of object type *objtype* is internally invalid.

System action: Command processing is terminated with command return code 12.

User response: Contact technical support if you cannot diagnose and correct the problem.

Destination: Message log.

CBKxx7316I CICS region *applid* **did not go through normal CICS BAC shutdown processing.**

Explanation: The CICS BAC component did not go through normal shutdown processing the last time it was active in CICS region *applid*.

System action: Command processing continues.

User response: None.

Destination: Message log.

CBKxx7317E Unknown EXCI error. Review *applid* **CICS log for more information.**

Explanation: CICS is processing a file maintenance utility command, but an error has occurred in the EXCI protocol.

System action: Command processing is terminated with command return code 12.

User response: Review any logs from the target CICS region for additional information. Contact technical support if you cannot diagnose and correct the problem.

Destination: Message log.

CBKxx7318E *objtype objname* **is protected. Requested operation was not performed.**

Explanation: The file maintenance utility is processing a command that references a resource to which access is denied by RACF because the file maintenance utility user ID does not have sufficient access authority for the resource identified by *objtype* and *objname*.

System action: The current command processing terminates with command return code 8.

User response: Consult your security administrator to obtain the required access authority to the objects for which permission has been refused.

Destination: Message log.

CBKxx7399E Internal error on *funcname* **call. Unknown return code: rc=***retcode* **rs=***rsncode*

Explanation: The file maintenance utility is processing a command but the function *funcname* has returned an unexpected return code *retcode* and reason code *rsncode*.

System action: Command processing terminates with command return code 12.

User response: Contact technical support if you cannot diagnose and correct the problem.

Destination: Message log.

CBKxx7801E Unknown command *command* **on line** *cmdlineno* **column** *cmdcolumn***.**

Explanation: The file maintenance utility is processing an input statement beginning on line *cmdlineno*, but has found that the word starting in column *cmdcolumn* is not a valid file maintenance utility command.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7802E Unknown parameter *parameter* **on line** *prmlineno* **column** *prmcolumn***.**

Explanation: The file maintenance utility is processing an input statement beginning on line*prmlineno*, but the word starting in column *prmcolumn* is not a valid parameter for the command.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7803E *parameter* **does not take any values: Line** *prmlineno* **column** *prmcolumn***.**

Explanation: The file maintenance utility is processing an input statement beginning on line*prmlineno*, but the parameter *parameter*, starting in column *prmcolumn*, has a value specified where no value is allowed.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7804E *parameter* **requires values: Line** *prmlineno* **column** *prmcolumn***.**

Explanation: The file maintenance utility is processing an input statement beginning on line*prmlineno*, but the parameter *parameter*, starting in column *prmcolumn*, has no value specified where a value is required.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7805E *parameter* **has too many values: Line** *prmlineno* **column** *prmcolumn***.**

Explanation: The file maintenance utility is processing an input statement beginning on line*prmlineno*, but the parameter *parameter*, starting in column *prmcolumn*, has more values specified than are allowable.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7806E Value can only be YES or NO: *parameter***(***value***) line** *vallineno* **column** *valcolumn***.**

Explanation: The file maintenance utility is processing an input statement beginning on line*vallineno*, the parameter *parameter* has a value of *value* specified, but you can specify only YES or NO on this parameter.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7807E Value is too long: *parameter***(***value***) line** *vallineno* **column** *valcolumn***.**

Explanation: The file maintenance utility is processing an input statement beginning on line*vallineno*, but the parameter *parameter* has a value of *value* specified and this value is too long for the parameter.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7808E Value has an invalid character: *parameter***(***value***) line** *vallineno* **column** *valcolumn***.**

Explanation: The file maintenance utility is processing an input statement beginning on line*vallineno*, but the parameter *parameter* has a value of *value* specified, where at least one of the characters is invalid for this parameter.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7809E First character is invalid: *parameter***(***value***) line** *vallineno* **column** *valcolumn***.**

Explanation: The file maintenance utility is processing an input statement beginning on line*vallineno*, but the parameter *parameter* has a value of *value* specified and the first character is invalid for this parameter.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7810E Null data set level: *parameter***(***value***) line** *vallineno* **column** *valcolumn***.**

Explanation: The file maintenance utility is processing an input statement beginning on line*vallineno*, but the parameter *parameter* has a value of *value* specified. Parameter *parameter* refers to a DSN but the data set name specified has a null data set qualifier.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7811E Long data set level: *parameter*(*value*) **line** *vallineno* **column** *valcolumn*.

Explanation: The file maintenance utility is processing an input statement beginning on line*vallineno*, but the parameter *parameter* has a value of *value* specified. Parameter *parameter* refers to a DSN but the data set name specified has a data set qualifier longer than 8 characters.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7812E No data set levels: *parameter*(*value*) **line** *vallineno* **column** *valcolumn*.

Explanation: The file maintenance utility is processing an input statement beginning on line*vallineno*, the parameter *parameter* has a value of *value* specified. Parameter *parameter* refers to a DSN but the data set name specified only has one data set qualifier.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7813E Trailing period is invalid: *parameter*(*value*) **line** *vallineno* **column** *valcolumn*.

Explanation: The file maintenance utility is processing an input statement beginning on line*vallineno*, but the parameter *parameter* has a value of *value* specified. Parameter *parameter* refers to a DSN but the data set name specified ends with a period.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7814E Invalid value: *parameter*(*value*) **line** *vallineno* **column** *valcolumn*.

Explanation: The file maintenance utility is processing an input statement beginning on line*vallineno*, but the parameter *parameter* has a value of *value* specified. The value specified is not valid for this parameter.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7815E Too deep of a nesting level: *parameter*(*value*) **line** *vallineno* **column** *valcolumn*.

Explanation: The file maintenance utility is processing an input statement beginning on line*vallineno*, but the parameter *parameter* has a value of *value* specified. However, the value specified is enclosed in too many parenthetical pairs.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7816E Extraneous parenthesis: line *lineno* **column** *column*.

Explanation: The file maintenance utility is processing an input statement beginning on line*lineno*, but has found extraneous parenthesis or parentheses within the command.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7817E Quote characters are not allowed in input: *lineno* **column** *column*.

Explanation: The file maintenance utility is processing an input statement beginning on line*lineno* and has found invalid quote characters in the command.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7819E Command *command* **has no parameters specified.**

Explanation: The file maintenance utility has received a command with no parameters but on this command parameters are required.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CBKxx7820E Parameter *parameter* **value** *value* **is out of range.**

Explanation: The file maintenance utility is processing command but the parameter *parameter* value *value* is outside of the permitted range.

System action: Command processing terminates with command return code 8.

User response: Correct the input and resubmit the job.

Destination: Message log.

CICS BAC miscellaneous messages (CBK*xx*8000–8499)

The following messages can issued by any CICS BAC component:

CBKxx8004I *identifier* **CICS region control file dynamically allocated: CICS(***applid***) DSN(***data-set-name***).**

Explanation: The CICS BAC region control file data set *data-set-name* for CICS region *applid* has been successfully allocated by either the communication server or the CICS region request server subtask. *identifier* is the communication server job name or the CICS region applid.

System action: None.

User response: None.

Destination: Console or job log

CBKxx8005I *identifier* **Unable to determine data set name to allocate CICS control file**

Explanation: During startup processing in the CICS region, CICS BAC was unable to determine the data set name for the required CICS BAC control file.

System action: CICS BAC startup processing is terminated.

User response: Check the CICS console messages for other messages that might help to diagnose the problem. Ensure that the required CICS BAC control file is correctly defined to the CICS region. If you cannot correct the problem, contact technical support.

Destination: Console

CBKxx8006E Unknown error detected while processing logical parmlib: Caller=*caller***, DDN=***ddname***, Member=***member-name***, Request=***request***, Code1=***retcode***, Code2=***rsncode***.**

Explanation: An unrecoverable error has occurred in either the CICS BAC communication server or in the batch request utility while processing the MVS logical parmlib concatenation. *caller* is a CICS BAC internal code that identifies the caller of the routine. *ddname* is the DD name passed to the MVS logical parmlib concatenation service.

member-name identifies the name of the member being processed as one of the following:
- CBKCFTBL
- CBKSRVR
- CBKBATCH

request is either ALLOCATE or READMEMBER, indicating the type of request being processed when the error occurred.

retcode and *rsncode* are the return and reason codes returned by the logical parmlib concatenation service.

System action: The communication server or batch request utility terminate.

User response: Contact technical support if the cause of the problem cannot be determined and corrected.

Destination: Console or job log

CBKxx8007E *jobname* **Unable to establish IRC communications to ship batch request to CICS region** *applid* **Batch request utility abending with abend code U3112.**

Explanation: The CICS BAC batch request utility is unable to establish IRC communications with CICS region *applid* while attempting to ship a batch request command to the CICS region.

System action: The batch request utility job step abends with abend code U3112.

User response: This condition is most likely occurs because no CICS region has started IRC communications on the MVS image, and the CICS EXCI DFHXCOPT options table does not specify an MRO SVC for EXCI communications use. See the *CICS External Interfaces Guide* for more information. Contact technical support if you cannot determine and correct the cause of the problem.

Destination: Job log

CBKxx8008E *identifier* **Unknown error received from region** *applid* **during** *request-type* **request. Error codes:** *code1 code2 code3 code4 code5*

Explanation: The communication server or batch request utility has received an unknown error condition in response to an EXCI request sent to CICS region *applid*.

identifier is the job name of the batch request utility if the error has occurred in batch request utility processing. *identifier* is the ID (*cbkid*) of the communication server if the error has occurred while processing a workstation administration client request.

request is 'EXCI' or 'DPL' indicating the type of request that resulted in the error:

- For an EXCI request, *code1* through *code5* provide the contents of the 5-word return area (return_area) passed back on all EXCI requests.
- For a DPL request, *code1* through *code3* provide the contents of the 3-word return area (dpl_retarea)passed back on an EXCI DPL_Request command. Note that for a DPL_request command, *code4* and *code5* are not meaningful and should be ignored.

See the *CICS External Interfaces Guide* for explanations of the EXCI response and reason codes areas (return_area and dpl_retarea) returned by the EXCI CALL Interface.

System action: If the error has occurred in the communication server address space as a result of a workstation administration client request, the communication server terminates the client request.

If the error has occurred in a batch request utility job step, the job step abends with a U3111 abend.

User response: If you can determine and correct the cause of the error, retry the request or rerun the batch request utility job step. Otherwise, contact technical support.

Destination: Console or job log

CBKxx8009E *identifier* **EXCI message for unknown error:** *msg-text***.**

Explanation: In some cases, an EXCI error condition described by message CBK*xx*8008E includes a message describing the error condition. When this happens, message CBK*xx*8009E will follow message CBK*xx*8008E in order to display up to eighty characters of the EXCI error message text.

If this occurs while processing a batch request utility request, *identifier* is the job name of the batch request utility. If this occurs while processing a workstation administration client request, *identifier* is the ID of the communication server that processed the request.

System action: See message CBK*xx*8008E.

User response: See message CBK*xx*8008E.

Destination: Console or job log

CBKxx8010E *jobname* **Error reading** *object-type* **object: DSN=***data-set-name* **errorcode=***vsam-fdbkword*

Explanation: The CICS BAC batch request utility *jobname* is attempting to read a record from the target CICS region control file and an error has occurred. *object-type* can be one of the following:
- CICS region properties
- File default properties
- TDQ default properties
- Transid default properties
- Program default properties

data-set-name is the data set name for the target region control file. *vsam-fdbkword* is the VSAM request RPL feedback word from the failing request.

System action: The batch request utility job step abends with abend code U1505.

User response: Contact technical support if you cannot determine and correct the cause of the problem.

Destination: Job log

CBKxx8011W *identifier* **EXCI error detected after reply received from region** *applid***. Processing continuing. Error codes:** *code1 code2 code3 code4 code5*

Explanation: The CICS BAC request server task running in CICS region *applid* has previously sent a normal response after processing a batch request utility command or a workstation administration client request. However, an error has now occurred after sending the normal response. This could occur, for example, if an error was returned on an EXCI Close_PIPE request.

code1 through *code5* provide the contents of the 5-word return area (return_area) passed back on all EXCI requests. See the *CICS External Interfaces Guide* for explanations of the EXCI response and reason codes areas (return_area and dpl_retarea) returned by the EXCI CALL Interface.

If this occurs while processing a batch request utility request, *identifier* is the job name of the batch request utility. If this occurs while processing a workstation administration client request, *identifier* is the ID of the communication server that processed the request.

System action: Processing continues.

User response: Verify that the results of the batch request utility or workstation request are what was expected. You should also determine the cause of the error condition to ensure that it does cause problems with future requests or commands.

Destination: Console or job log

CBKxx8012W *identifier* **EXCI message for unknown error:** *msg-text***.**

Explanation: In some cases, an EXCI error condition as described by message CBK*xx*8011W will include a message describing the error condition. When this happens, message CBK*xx*8012W will follow message CBK*xx*8011W in order to display up to eighty characters of the EXCI error message text.

If this occurs while processing a batch request utility request, *identifier* is the job name of the batch request utility. If this occurs while processing a workstation administration client request, *identifier* is the ID of the communication server that processed the request.

System action: See message CBK*xx*8011W.

User response: See message CBK*xx*8011W.

Destination: Console or job log

CBKxx8013E *identifier* **CICS region** *applid* **termination detected during EXCI processing. Results unpredictable. Error codes** *retcodersncode*

Explanation: The target CICS region *applid* has terminated while processing a request from a batch request utility job step or a workstation client request from a communication server. *retcode* and *rsncode* contain CICS BAC internal error codes.

If this occurs while processing a batch request utility request, *identifier* is the job name of the batch request utility. If this occurs while processing a workstation administration client request, *identifier* is the ID of the communication server that processed the request.

System action: If this occurred during batch request utility processing, the job step terminates with abend code U3113.

If this occurs during the processing of a workstation request, the communication server terminates the request.

User response: Results are unpredictable depending on when the region terminated with respect to the CICS BAC processing that was taking place. You should evaluate the results of the request and take action accordingly.

Destination: Console or job log

CBKxx8015E *identifier* **Invalid control file detected by** *caller***:** *object-type* **record not found. REGION=***applid* **DSN=***data-set-name request* **terminated.**

Explanation: The CICS BAC communication server or batch request utility is processing a CICS BAC region control file that does not contain one of the required object records. The missing object record type is *object-type*.

identifier is either the job name of the batch request utility or the communication server ID (*cbkid*).

caller is either 'communication server' or 'batch request utility'.

applid is the applid of the CICS region that owns the control file.

data-set-name is the CICS region control file data set name.

request is either 'client request' for the communication server, or 'batch request utility' for a batch request utility request.

System action: In the case of the batch request utility, the job step terminates with condition code 12.

In the case of a communication server that is processing a workstation administration client request, the client request terminates.

User response: This can be caused by a corrupt control file. Contact technical support if you cannot determine and correct the cause of the problem.

Destination: Job log

CBKxx8016E *identifier* **Invalid control file version detected:** *caller*=*caller-version* **File=***file-version* **REGION=***applid* **DSN=***data-set-name request* **terminated.**

Explanation: The CICS BAC communication server or batch request utility is processing a CICS BAC region control file that is at an incompatible version or release level. The communication server or batch request utility version is *caller-version* and the control file version is *file-version*.

identifier is the job name of either the batch request utility or communication server ID.

caller is either 'communication server' or 'batch request utility'.

applid is the applid of the CICS region that owns the control file.

data-set-name is the CICS region control file data set name.

request is either 'client request' for the communication server, or 'batch request utility' for a batch request utility request.

System action: In the case of the batch request utility, the job step terminates with condition code 12.

In the case of a communication server that is processing a workstation administration client request, the client request terminates.

User response: Ensure the control file and product code of the communication server and the batch request utility are at compatible levels. Contact technical support if you cannot determine and correct the cause of the problem.

Destination: Job log

CBKxx8017E *identifier* **Empty control file detected by** *caller***. REGION=***applid* **DSN=***data-set-name request* **terminated.**

Explanation: The CICS BAC communication server or batch request utility is processing a CICS BAC region control file that is empty.

identifier is the job name of either the batch request utility or communication server ID.

caller is either 'communication server' or 'batch request utility'.

applid is the applid of the CICS region that owns the control file.

data-set-name is the CICS region control file data set name.

request is either 'client request' for the communication server, or 'batch request utility' for a batch request utility request.

System action: In the case of the batch request utility, the job step terminates with condition code 12.

In the case of a communication server that is processing a workstation administration client request, the client request terminates.

User response: The most likely cause of this error is a request directed to a region control file that is not initialized. Initialize the control file and retry the request.

Destination: Job log

CBKxx8018E *cbkid* **Unable to establish IRC communications to ship client request to CICS region** *applid***. Client request terminated.**

Explanation: The CICS BAC communication server *cbkid* is unable to establish IRC communications while attempting to send a workstation request to CICS region *applid*.

System action: The communication server terminates workstation request.

User response: This condition is most likely occurs because no CICS region has started IRC communications on the MVS image, and the CICS EXCI DFHXCOPT options table does not specify an MRO SVC for EXCI communications use. See the *CICS External Interfaces Guide*for more information. Contact technical support if you cannot determine and correct the cause of the problem.

Destination: Console or job log

CBKxx8020I *jobname* **CICS BAC shutdown flag for region** *applid* **successfully reset by** *caller***.**

Explanation: CICS BAC is not active in CICS region *applid*, and CICS BAC did not go through normal shutdown processing the last time it was active. The region shutdown flag is reset to normal because the CICS region properties record in the control file specifies this action (see the Region termination options tab in the Maintain CICS Region Properties window of the workstation client). *caller* is either 'communication server' or 'batch request utility'.

System action: None.

User response: None.

Destination: Job log

Audit log messages (CBK*xx*8500–8999)

The messages described in this section are written to the CICS BAC audit log by the CICS request server, or by the batch request utility if it services a command directly because the target CICS region is not available. Expressions such as "written by CICS BAC", or similar, are used generically in the message explanations to mean either one of these two components. The DD name for the audit log is CBKLOG.

The CICS region and the batch request utility job step dynamically allocate the audit log, based on the audit log attributes in the CICS region properties. You can define, view, or alter the audit log attributes using the workstation administration client, opening the Region Properties window and selecting the Miscellaneous options tab.

CBKxx8501I *date time* **Start: CICS=***applid***By={STARTUP|CICSSVR|BATCH}** *jobname stepname procname jobnumber sysid* **By=CICSSPI** *jobname userid termid transid*

Explanation: CICS BAC is performing one of the following actions:

By=STARTUP
CICS BAC is starting in a CICS region. The CICS BAC startup procedure identifies the CICS region by *jobname*, *stepname*, *procname*, and *jobnumber*, and it also displays the MVS *sysid*.

By=CICSSVR
The CICS BAC request server is processing a batch utility request. The CICS BAC request server identifies the batch request utility by its *jobname*, *stepname*, *procname*, and *jobnumber*, and it also displays the MVS *sysid*.

BY=BATCH
The CICS BAC batch request utility is servicing a

command, or set of commands, itself because CICS BAC is not active in the target CICS region. The CICS BAC batch request utility identifies itself by its *jobname*, *stepname*, *procname*, *jobnumber*, and it also displays the MVS *sysid*.

BY=CICSSPI
The CICS BAC state monitor within the CICS region has detected a change to a CICS resource. The CICS BAC state monitor identifies the the task responsible for the change by its *userid*, *termid*, and *transid*.

In all the above cases, the CICS BAC component displays the *date* and *time* of the request.

System action: None

User response: None

Destination: Audit log

CBKxx8503I *seqnum* **LIST {START|END}** *listname* **O=openstatus E=***enablestatus* **DSN={PRIMARY|ALTERNATE}**

Explanation: CICS BAC has started processing, or has finished processing, a CICS BAC application list. *listname* shows the name of the application list being processed, and *openstatus*, *enablestatus*, and DSN show the values specified by the command.

System action: None

User response: None

Destination: Audit log

CBKxx8504I *seqnum* **GROUP {START|END}** *groupname* **O=***openstatus* **E=***enablestatus* **DSN={PRIMARY|ALTERNATE}**

Explanation: CICS BAC has started processing, or has finished processing, a CICS BAC application group. *groupname* shows the name of the application group being processed, and *openstatus*, *enablestatus*, and DSN show the values specified by the command.

System action: None

User response: None

Destination: Audit log

CBKxx8505I *seqnum* **FILE {BEFORE|REQUESTED|AFTER}** *filename* **O=***openstatus* **E=***enablestatus***A=***addstatus* **U=***updatestatus* **D=***deletestatus* **R=***readstatus* **B=***browsestatus* **T=***processtransids* **P=***processprograms disp dsn*

Explanation: CICS BAC is processing a file state change request. A CBK*xx*8505I message can be written up to three times for each request, depending on the specific processing that takes place:

BEFORE
CICS BAC writes a BEFORE record for each request.

REQUESTED|AFTER
If the request is serviced successfully, CICS BAC follows the *before* state record by a REQUESTED record and an AFTER record. If CICS BAC does not write a REQUESTED or AFTER record to the log, it writes other records to explain why the request was not serviced

A file record in a CICS BAC control file can specify transactions and programs that are to be processed when a file request is being processed by a batch request utility command. If the file record does specify transactions and programs to be processed, the audit log can contain transaction ID and program name audit log records between the BEFORE and AFTER file audit log records, depending on the state changes requested.

seqnum is the command sequence number relative to one if the request is performed by the CICS BAC batch request utility. For all other cases, such as a CICS SPI request, *seqnum* is zero.

The various status fields (*openstatus*, *enablestatus*, and so on) contain various values depending on the type of record written (BEFORE, REQUESTED, or AFTER), and where the request is serviced (CICS BAC request server or CICS BAC batch request utility).

For BEFORE and AFTER log records, a question mark (?) status value means that CICS BAC does not know the status of the file at the time of the request. For example, if a batch request utility command is used to process a file newly defined to CICS BAC while the CICS region that owns the file is not active, CICS BAC is not able to determine the current status of the file. BEFORE and AFTER log records do not contain the *processtransids* or *processprograms* status indicators.

For REQUESTED log records, a blank status value means that the resource state change request did not attempt to change that particular state value.

The variables in the message text can have the following values:

Variable
Possible values (and meanings)

openstatus
O (open) or C (closed)

enablestatus
E (enabled), D (disabled), or U (unenabled)

addstatus
Y (addable) or N (not addable)

updatestatus
Y (updatable) or N (not updatable)

deletestatus
Y (deletable) or N (not deletable)

readstatus
Y (readable) or N (not readable)

browsestatus
Y (browsable) or N (not browsable)

processtransids
Y (associated transaction IDs are processed) or N (associated transaction IDs are not processed)(REQUESTED records only)

processprograms
Y (associated programs are processed) or N (associated programs are not processed)(REQUESTED records only)

disp SHR, OLD, or blank (the disposition of the file)

dsn The name of the data set associated with the file, or blank

System action: None

User response: None

Destination: Audit log

CBKxx8506I *seqnum* **TDQUEUE {BEFORE|REQUESTED|AFTER}** *tdqname* **O=***openstatus* **E=***enablestatus* **T=***processtransids* **P=***processprograms*

Explanation: CICS BAC is processing a transient data queue state change request. A CBK*xx*8506I message can be written up to three times for each request, depending on the specific processing that takes place.

BEFORE
CICS BAC writes a BEFORE record for each request.

REQUESTED|AFTER
If the request is serviced successfully, CICS BAC follows the *before* state record by a REQUESTED record and an AFTER record. If CICS BAC does not write a REQUESTED or AFTER record to the log, it writes other records to explain why the request was not serviced

seqnum is the command sequence number relative to one if the request is performed by the CICS BAC batch request utility. For all other cases, such as a CICS SPI request, *seqnum* is zero.

The status fields *openstatus* and *enablestatus* contain various values depending on the type of record written (BEFORE, REQUESTED, or AFTER), and where the request is serviced (CICS BAC request server or CICS BAC batch request utility).

For BEFORE and AFTER log records, a question mark (?) status value means that CICS BAC does not know the status of the transient data queue at the time of the request. For example, if a batch request utility command is used to process a transient data queue newly defined to CICS BAC while the CICS region that owns the transient data queue is not active, CICS BAC is not able to determine the current status of the queue. BEFORE and AFTER log records do not contain the *processtransids* or *processprograms* status indicators.

For REQUESTED log records, a blank status value means that the resource state change request did not attempt to change that particular state value

The variables in the message text can have the following values:

Variable
Possible values (and meanings)

openstatus
O (open) or C (closed)

enablestatus
E (enabled), D (disabled), or U (unenabled)

processtransids
Y (associated transaction IDs are processed) or N (associated transaction IDs are not processed)(REQUESTED records only)

processprograms
Y (associated programs are processed) or N (associated programs are not processed)(REQUESTED records only)

System action: None

User response: None

Destination: Audit log

CBKxx8507I *seqnum* **TRANSID {BEFORE|REQUESTED|AFTER}** *transid* **E=***enablestatus*

Explanation: CICS BAC is processing a transaction ID state change request. A CBK*xx*8507I message can be written up to three times for each request, depending on the specific processing that takes place.

BEFORE
CICS BAC writes a BEFORE record for each request.

REQUESTED|AFTER
If the request is serviced successfully, CICS BAC follows the *before* state record by a REQUESTED record and an AFTER record. If CICS BAC does not write a REQUESTED or AFTER record to the log, it writes other records to explain why the request was not serviced

seqnum is the command sequence number relative to one if the request is performed by the CICS BAC batch request utility. For all other cases, such as a CICS SPI request, *seqnum* is zero.

The *enablestatus* field contains various values depending on the type of record written (BEFORE, REQUESTED, or AFTER), and where the request is serviced (CICS BAC request server or CICS BAC request utility).

For BEFORE log records, a question mark (?) status value means that CICS BAC does not know the status of the transaction ID at the time of the request. For example, if a batch request utility command is used to process a transaction ID newly defined to CICS BAC while the CICS region that owns the transaction ID is not active, CICS BAC is not able to determine the current status of the transaction ID.

Where the status is known, the value can be an E (enabled) or D (disabled).

System action: None

User response: None

Destination: Audit log

CBKxx8508I *seqnum* **PROGRAM {BEFORE|REQUESTED|AFTER}** *program* **E=***enablestatus*

Explanation: CICS BAC is processing a program state change request. A CBK*xx*8508I message can be written up to three times for each request, depending on the specific processing that takes place.

BEFORE
CICS BAC writes a BEFORE record for each request.

REQUESTED|AFTER
If the request is serviced successfully, CICS BAC follows the *before* state record by a REQUESTED record and an AFTER record. If CICS BAC does not write a REQUESTED or AFTER record to the log, it writes other records to explain why the request was not serviced

seqnum is the command sequence number relative to one if the request is performed by the CICS BAC batch request utility. For all other cases, such as a CICS SPI request, *seqnum* is zero.

For all types of record (BEFORE, REQUESTED, and AFTER) the *enablestatus* variable can be either E or D indicating that the program is either enabled or disabled. For BEFORE record types only, an enable status shown as a question mark (?) means that CICS BAC does not know the status of the program at the time of the request. For example, if a batch request utility command is used to process a program newly defined to CICS BAC while the CICS region that owns the program is not active, CICS BAC is not able to determine the current status of the transaction ID.

System action: None

User response: None

Destination: Audit log

CBKxx8511I *seqnum resourcetype resourcename* **State not changed due to CICSSTARTMODE=UPDATE requested**

Explanation: CICS BAC is being started in the CICS region with CICSSTARTMODE parameter specified with a value of UPDATE.

With a start mode of UPDATE, CICS BAC does not set the state of any CICS resources. It does, however, update the resource records defined in the CICS BAC control file with the current state of the resource specified by *resourcetype* and *resourcename*.

System action: None

User response: None

Destination: Audit log

CBKxx8512I *seqnum resourcetype resourcename* **State not changed because requested state is unknown.**

Explanation: During CICS BAC startup processing in a CICS region, CICS BAC has read a resource record where the last requested status of the resource is unknown. CICS BAC is unable to set the state of the resource.

This situation can occur if you add a resource object record to a CICS BAC region control file while the CICS region is not available. In this situation, CICS BAC is unable to determine the last state of the object.

System action: None

User response: None

Destination: Audit log

CBKxx8513I *seqnum resourcetype resourcename* **already at requested state. No state changes necessary**

Explanation: CICS BAC has received a request to change the state of a resource, but the resource is already at the requested state. CICS BAC leaves the state of the resource unchanged.

System action: None

User response: None

Destination: Audit log

CBKxx8514I *seqnum* **FILE** *filename* **Can't change DSN. File permanently allocated. Req DSN=***datasetname*

Explanation: The CICS BAC request server is processing a batch request utility command that specifies, either by the NEWDSN or DSN parameter, a change of data set name for file *filename*. However, *filename* is defined in the CICS region startup JCL and allocated to the CICS job my MVS, thus CICS BAC cannot change the data set name. You can change the name of a data set only when CICS files are dynamically allocated by CICS, and *not* when they are allocated by MVS during job step initiation.

This condition can also occur during CICS BAC startup processing if a request to change the data set name was processed directly in the CICS region control file, by the batch request utility, while the target CICS region was not available. The name change request is then detected during the next execution of CICS BAC startup processing. At this point, the CICS BAC startup processor can find that the name change is not allowed, and reject the request, as described above for the CICS BAC request server.

System action: If this occurs as a result of a batch request utility command, the batch request utility job step terminates. If it occurs during CICS BAC startup processing, the data set name is not changed, and startup processing continues.

User response: If this occurs as a result of a batch request utility command, correct the command and rerun the job step.

Destination: Audit log

CBKxx8515I *seqnum resourcetype resourcename* **not processed. Object not defined to CICS.**

Explanation: CICS BAC is processing a batch request utility command that specifies a resource (*resourcetype* and *resourcename*) that is not defined to CICS.

CICS BAC also writes this audit log message if, during CICS BAC startup processing, it finds that the control file contains a resource that is not defined in the CICS region.

System action: If CICS BAC writes this log record as a result of processing a batch request utility command, the batch request utility job step terminates with return code 12.

If CICS BAC writes this log record during CICS BAC startup processing, startup processing continues.

User response: If CICS BAC writes this audit log message as a result of a batch request utility command, correct the command and rerun the job step.

Destination: Audit log

CBKxx8516E *seqnum resourcetype resourcename* **Invalid attributes on control file: O=***openstatus* **E=***enablestatus* **A=***addstatus* **U=***updatestatus* **D=***deletestatus* **R=***readstatus* **B=***browsestatus* **DISP=***disposition***.**

Explanation: During CICS BAC startup processing in the CICS region, one or more invalid status values have been detected in the control file record for the resource named by *resourcename*. *openstatus*, *enablestatus*, *addstatus*, *updatestatus* , *deletestatus*, *readstatus*, *browsestatus*, and *disposition* give the hexadecimal value(s) of the status field(s) in error. A blank status field value means the value is not in error, or it is not relevant for the type of resource being processed. For example, the only relevant status field for programs and transaction ID's is *enablestatus*.

System action: CICS BAC does not change the status of the object. Startup processing continues.

User response: Contact technical support.

Destination: Audit log

CBKxx8517I *seqnum* **FILE** *filename* **Forcing {PRIMARY|ALTERNATE} DSN:** *eibresp eibresp2 enablestatus datasetname*

Explanation: CICS BAC has detected that there is no data set name for a file it is about to open. Because of the NODSN parameter value specified when CICS BAC was started in the CICS region, CICS BAC sets the data set name for the file to either the primary or alternate data set name specified in the CICS BAC control file record for the file. *eibresp* and *eibresp2* give the resulting EIBRESP and EIBESP2 values in hexadecimal format for the EXEC CICS SET FILE command that set the data set name. *enablestatus* is the hexadecimal representation of the value used for the ENABLE parameter when the EXEC CICS SET FILE command was issued. The meaning of the numeric values can be found in *Appendix A* of the *CICS System Programming Reference*. . *datasetname* is the data set name that CICS BAC set for the file.

System action: None

User response: None

Destination: Audit log

CBKxx8520I *seqnum resourcetype resourcename* **Record did not exist on control file but has been added.**

Explanation: CICS BAC has processed a batch request utility command that specifies a resource (*resourcetype* and *resourcename*) that is not defined in the CICS region control file. You have specified in the CICS region properties that unknown objects of this type are to be added to the control file.

System action: CICS BAC adds a record for the object

resourcetype and *resourcename* to the CICS region control file.

User response: None

Destination: Audit log

CBKxx8521I *seqnum resourcetype resourcename* **Record did not exist on control file. Record not added.**

Explanation: CICS BAC has processed a batch request utility command that specifies a resource (*resourcetype* and *resourcename*) that is not defined in the CICS region control file. You have specified in the CICS region properties that unknown objects of this type are to be processed but *not* added to the control file. CICS BAC has processed the command, but has not added a record for the resource to the control file.

System action: The command is processed.

User response: None

Destination: Audit log

CBKxx8522I *seqnum resourcetype resourcename* **Record did not exist on control file. Command not processed.**

Explanation: CICS BAC has processed a batch request utility command that specifies a resource (*resourcetype resourcename*) that is not defined in the CICS region control file. You have specified in the CICS region properties that unknown objects of this type are not to be processed.

System action: CICS BAC does not add a record for the resource to the control file. Further action taken depends on the CICS BAC region properties for the CICS region. The command can be ignored, in which case processing continues with the next command. Alternatively, the batch request utility job step can terminate.

User response: Correct the problem and resubmit the batch request utility job step.

Destination: Audit log

CBKxx8528I *seqnum* **FILE** *filename* **File must be temporarily set to CLOSED/DISABLED to set to final requested state.**

Explanation: During its startup processing in the CICS region, CICS BAC has determined that a file must be closed and disabled before it can set the file to its last requested state. This is because the file is currently open and enabled, or open and disabled, or closed and enabled. Even though the last requested state might be open and enabled, or closed and enabled, it must still be set to closed and disabled first for the change to be made. For example, if CICS BAC needs to change one or more of the file access attributes (such as READ, UPDATE, and ADD), CICS requires that a file must be closed and disabled before these attributes can be changed. Note that this message can occur only during CICS BAC startup processing.

System action: Processing continues. The file will be temporarily set to CLOSED DISABLED before setting to its last requested state.

User response: None

Destination: Audit log

CBKxx8529I *seqnum* **SET FILE(X'***filename***'), OPENSTATUS(***openstatus***), ENABLESTATUS(***enablestatus***)**

Explanation: CICS BAC displays the EXEC CICS SET FILE command issued by CICS BAC startup processing if a file must be temporarily closed or disabled to set it to its last requested state. *filename* provides the hexadecimal representation of the file name used for the SET FILE command. *openstatus* and *enablestatus* provide the OPENSTATUS and ENABLESTATUS values used for the SET FILE command. The meaning of the two numeric values can be found in *Appendix A* in the *CICS System Programming Reference*. For additional information on the circumstances that result in this message being displayed, see message CBK*xx*8528I.

System action: Processing continues.

User response: None

Destination: Audit log

CBKxx8530I *seqnum resourcetype resourcename* **Objects of this type not allowed for this region. Object not processed.**

Explanation: CICS BAC has processed a batch request utility command that specifies a resource type that is not allowed for this region by the CICS BAC region properties defined in the CICS region control file.

System action: CICS BAC action depends on the CICS BAC region properties for the CICS region. The command can be ignored, in which case processing continues with the next command. Alternatively, the batch request utility job step can terminate.

User response: Correct the problem and resubmit the batch request utility job step.

Destination: Audit log

CBKxx8532I *seqnum tdqname* **Open/close and enable/disable cannot be set for indirect transient data queue.**

Explanation: CICS BAC is attempting to process a batch request utility command that specifies one of the following actions for an indirect transient data queue.

- Change the open status to open or closed

- Change the enabled status to enabled or disabled.

You cannot use CICS BAC to open, close, enable, or disable indirect transient data queues.

System action: Processing continues and CICS BAC processes any transaction IDs or programs associated with the transient data queue.

User response: None

Destination: Audit log

CBKxx8533I *seqnum tdqname* **Open/close cannot be set for intrapartition transient data queue.**

Explanation: CICS BAC is attempting to process a batch request utility command to open or close an intrapartition transient data queue. You cannot use CICS BAC to open or close intrapartition transient data queues.

System action: Processing continues and CICS BAC processes any transaction IDs or programs associated with the transient data queue.

User response: None

Destination: Audit log

CBKxx8534I *seqnum tdqname* **Open/close cannot be set for remote transient data queue.**

Explanation: CICS BAC is attempting to process a batch request utility command to open or close a remote transient data queue. You cannot use CICS BAC to open or close a remote transient data queues.

System action: Processing continues and CICS BAC processes any transaction IDs or programs associated with the transient data queue.

User response: None

Destination: Audit log

CBKxx8535I *seqnum* **{LINK|START|RUNCEMT}** {*transid* | *program*} **Requests of this type not allowed for this region. Request not processed.**

Explanation: CICS BAC has received a command that is not allowed in this CICS region. The message includes either the *transid* or the *program* message variable, depending on the type of command, as follows:

- For LINK commands only, *program* is set to the name of the target program .
- For START commands, *transid* is set to the name of the transaction to be started
- For RUNCEMT commands, *transid* is set to CEMT.

System action: CICS BAC rejects the command. Further action taken depends on the CICS BAC region properties for the CICS region. The command can be ignored, in which case processing continues with the next command. Alternatively, the batch request utility job step can terminate.

User response: Correct the problem and resubmit the batch request utility job step.

Destination: Audit log

CBKxx8536I *seqnum filename* **State changes cannot be performed for a remote file. Processing continues.**

Explanation: CICS BAC is attempting to process a batch request utility command to change the state of a remote file. You cannot use CICS BAC to change the state of remote files.

System action: Processing continues and CICS BAC processes any transaction IDs or programs associated with the named file.

User response: None

Destination: Audit log

CBKxx8538E *seqnum filename* **RLS file state change(s) not allowed based on CICS BAC control file setting. Processing terminated.**

Explanation: CICS BAC has received a request from the CICS BAC batch request utility to change the state of a file defined as using record level sharing (RLS). The RLS option for this file is set to terminate or abend such requests.

System action: Processing is terminated for the batch request utility job step. The job step is either terminated immediately or abended depending on the settings for the target file.

User response: Determine whether or not this action is appropriate for your environment. Change the associated CICS BAC settings if a different CICS BAC action is required.

Destination: Audit log

CBKxx8539I *seqnum filename* **RLS file in quiescing state. Waiting two seconds for state to stabilize.**

Explanation: CICS BAC is attempting to change the quiesce status for the data set associated with *filename*. The data set is currently in the quiescing transient state. CICS BAC must wait for the state to change from quiescing to some other state before it can process the request. After two seconds, CICS BAC checks the data set state again. It continues to do this until the data set state is no longer quiescing, or it has waited thirty times (approximately one minute).

System action: If the data set state changes to something other than quiescing, CICS BAC continues to

process the batch request utility request. If after waiting for two seconds thirty times, the data set quiesce state is still quiescing, the batch request utility job step is terminated with a condition code of twelve.

User response: If the batch request utility job step is terminated because the data set quiesce state remained quiescing for too long, determine why the data set is in that state. If necessary, ensure that the state has changed from quiescing and rerun the batch request utility job step.

Destination: Audit log

CBKxx8540I *seqnum* **{LINK | START | RUNCEMT}** {*transid* | *program*} **CICS region not active. {LINK | START | RUNCEMT} request ignored. Processing continues.**

Explanation: The CICS BAC batch request utility is trying to process a LINK, START, or RUNCEMT command at a time when it is processing requests directly because CICS BAC is not active in the target CICS region, or because the CICS region is not running. Because it cannot pass the command to the target CICS region, it cannot process commands of the type LINK, START, and RUNCEMT.

System action: The command is ignored and because the NOTACTIVE parameter specifies CONTINUE, batch request utility processing continues with the next command.

User response: None

Destination: Audit log

CBKxx8541I *seqnum* **{LINK | START | RUNCEMT}** {*transid* | *program*} **CICS region not active. Job step will be terminated: RC=12**

Explanation: The CICS BAC batch request utility is trying to process a LINK, START, or RUNCEMT command at a time when it is processing requests directly because CICS BAC is not active in the target CICS region, or because the CICS region is not running. Because it cannot pass the command to the target CICS region, CICS BAC cannot process commands of the type LINK, START, and RUNCEMT.

System action: The command is ignored and because the NOTACTIVE parameter specifies TERMINATE, the batch request utility terminates.

User response: None

Destination: Audit log

CBKxx8542E *seqnum filename* **OPEN requested for migrated RLS data set or RLS not supported. Processing terminated.**

Explanation: CICS BAC is attempting to process a file that is defined to CICS as using record level sharing (RLS). Either the data set associated with the file has been migrated, or RLS processing in the CICS region is not allowed (by the CICS SIT RLS parameter).

System action: If the request is for CICS BAC startup processing, the file state is not changed and processing continues with the next object. If the request originated from the CICS BAC batch request utility, the batch request utility job step is immediately terminated with a condition code of twelve.

User response: Call technical support if the cause of this problem cannot be determined and corrected.

Destination: Audit log

CBKxx8543E *seqnum filename* **Data set exceeded quiescing state time limit. Processing terminated.**

Explanation: CICS BAC has exceeded its time limit for waiting for a data set to change from the quiescing state. See message CBK*xx*8539I for more information. This message is issued only when the file is defined in CICS as using record level sharing.

System action: If the request is for CICS BAC startup processing, the file's state is not changed and processing continues with the next object. If the request originated from the CICS BAC batch request utility, the batch request utility job step is immediately terminated with a condition code of twelve.

User response: If you cannot determine the cause of this problem and correct it, call technical support.

Destination: Audit log

CBKxx8550I *seqnum resourcetype resourcename* **EXCLUDED: Action={Terminate | Abend} {AC | CC}=***code*

Explanation: The target resource of a CICS BAC batch request utility command is defined as excluded in the CICS region control file. The resource type is *resourcetype* and resource name is *resourcename*.

System action: If the resource is defined as EXCLUDE and TERMINATE, or EXCLUDE and ABEND, CICS BAC either terminates with termination code (CC) or abends with the abend code (AC) as defined.

User response: None

Destination: Audit log

CBKxx8551I *seqnum resourcetype resourcename* **set as excluded in CICS BAC control file.**

Explanation: During startup processing in the CICS region, CICS BAC has detected a resource that is defined as excluded in the CICS region control file. The resource type and resource name are *resourcetype* and *resourcename*.

System action: The state of the resource is not changed, even if the CICS BAC control file contains a requested state different from the current state of the resource.

User response: None

Destination: Audit log

CBKxx8560I *seqnum resourcetype* **{BEFORE|REQUESTED|AFTER}** *filename* **RLS data set detected: Quiesce state={Q|U} DSN=***dataset*

Explanation: CICS BAC has detected that a batch request utility command is attempting to open or close a file that is defined to CICS as RLSACCESS(YES). In order to process the command, CICS BAC might have to change the quiesce state of the data set associated with the file. The data set name associated with the file is *dataset*. The various quiesce states, given as Q (for quiesce) or U (for unquiese), are indicated by the message as follows:

- If the message text states BEFORE, the quiesce state is before CICS BAC process the command.
- If the message text states REQUESTED, the quiesce state is the state required in order for CICS BAC to process the open or close command.
- If the message text states AFTER, the quiesce state is after CICS BAC has altered it, if it was not already at the requested state.

System action: None

User response: None

Destination: Audit log

CBKxx8561I *seqnum resourcetype filename* **Data set already at requested quiesce state:** *dataset*

Explanation: CICS BAC has determined that a data set *dataset* associated with a target file *filename* is defined as RLSACCESS(YES) and is currently at the quiesce state necessary for CICS BAC to perform the open or close request against the target file.

System action: None

User response: None

Destination: Audit log

CBKxx8562E *seqnum* **FILE** *filename* **RLS {INQUIRE|SET} error detected: EIBRESP=***eibresp* **EIBRESP2=***eibresp2*

Explanation: While processing an RLS data set, a failure has occurred in CICS BAC on an EXEC CICS INQUIRE DSNAME or an EXEC CICS SET DSNAME command. The CICS EIBRESP *eibresp* and EIBRESP2 *eibresp2* values indicate the cause of the error. For an explanation of these CICS SPI error codes, see the *System Programming Reference*.

System action: Processing for the command terminates and CICS BAC does not process any further commands from the batch request utility. The batch request utility job step is terminated with return code 12.

User response: Contact technical support if you cannot determine the reason for the failure and correct it.

Destination: Audit log

CBKxx8570I *seqnum* **{LINK|START|RUNCEMT}** {*transid* | *program*} *termid* **starting**

Explanation: The CICS BAC request server is about to process a LINK, START, or RUNCEMT command. The message includes the *transid*, *program*, or *termid* message variable, depending on the type of command, as follows:

- For LINK commands only, *program* is set to the name of the target program .
- For START commands, *transid* is set to the name of the transaction to be started, and *termid* is set to the terminal ID specified on the START command, but if no terminal ID is specified *termid* is set to spaces.
- For RUNCEMT commands, *transid* is set to CEMT.

System action: None

User response: None

Destination: Audit log

CBKxx8579I *seqnum* **{LINK|START|RUNCEMT}** {*transid* | *program*} *termid* **EIBRESP=***eibresp* **EIBRESP2=***eibresp2* **CC=12**

Explanation: The CICS BAC request server is processing a LINK, START, or RUNCEMT command and CICS has returned an error. The message includes the *transid*, *program*, or *termid* message variables, depending on the type of command, as follows:

- For LINK commands only, *program* is set to the name of the target program .
- For START commands, *transid* is set to the name of the transaction to be started, and *termid* is set to the terminal ID specified on the START command, but if no terminal ID is specified *termid* is set to spaces.
- For RUNCEMT commands, *transid* is set to CEMT.

For LINK and RUNCEMT commands, *eibresp* and *eibresp2* are the EIB response codes returned from the EXEC CICS LINK command. For START commands, *eibresp* and *eibresp2* are the EIB response codes returned from the EXEC CICS START command.

System action: The batch request utility job step terminates with return code 12

User response: Correct the problem and rerun the batch request utility job step.

Destination: Audit log

CBKxx8580I *seqnum* **SET FILE(X'***filename***'), OPENSTATUS(***openstatus***), ENABLESTATUS(***enablestatus***), DISPOSITION(***disposition***),**

Explanation: When CICS BAC issues an EXEC CICS SET FILE command during startup processing or while processing a batch request utility job step command, it writes this message to the CICS BAC audit log to document the command and the parameters used. *filename* is the hexadecimal representation of the name of the file for which the EXEC CICS SET FILE command is being issued. *openstatus*, *enablestatus*, and *disposition* are the values in decimal form for the parameters used with the command. The meaning of these values are given in *Appendix A* in the *CICS System Programming Reference*. This message is followed by message CBK*xx*8581I and, optionally, message CBK*xx*8582I to provide the remainder of the EXEC CICS SET FILE command parameter values.

System action: Processing continues.

User response: None

Destination: Audit log

CBKxx8581I *seqnum* **ADD(***addstatus***), UPDATE(***updatestatus***), DELETE(***deletestatus***), READ(***readstatus***), BROWSE(***browsestatus***),**

Explanation: When CICS BAC issues an EXEC CICS SET FILE command during startup processing or while processing a batch request utility job step command, it writes this message following message CBK*xx*8580I to the CICS BAC audit log to document the command and the parameters used. *addstatus*, *updatestatus*, *deletestatus*, *readstatus*, and *browsestatus* are the values in decimal form for the parameters used with the command. The meaning of these values are given in *Appendix A* in the *CICS System Programming Reference*. This message can be followed by message CBK*xx*8582I.

System action: Processing continues.

User response: None

Destination: Audit log

CBKxx8582I *seqnum* **DSNAME(X'***datasetname***')**

Explanation: When CICS BAC issues an EXEC CICS SET FILE command during startup processing or while processing a batch request utility job step command, and the command contains the DSNAME parameter, it writes this message following message CBK*xx*8581I to the CICS BAC audit log to document the data set name used with the command.*datasetname* contains the hexadecimal representation of the DSNAME parameter value.

System action: Processing continues.

User response: None

Destination: Audit log

CBKxx8583I *seqnum* **SET DSNAME(***X'datasetname***')**

Explanation: CICS BAC writes this message CBK*xx*8583I to the CICS BAC audit log when CICS BAC issues an EXEC CICS SET DSNAME command to change the quiesce state of a data set associated with a file being processed by CICS BAC. This can occur during CICS BAC startup processing in the CICS region, or when CICS BAC is processing a CICS BAC batch request utility command in the CICS region.*datasetname* contains the hexadecimal representation of the DSNAME parameter value. This message is followed by message CBK*xx*8584I.

System action: Processing continues.

User response: None

Destination: Audit log

CBKxx8584I *seqnum* **QUIESCESTATE(***quiescestate***)**

Explanation: This message CBK*xx*8584I follows message CBK*xx*8583I on the CICS BAC audit log when CICS BAC issues an EXEC CICS SET DSNAME command to change the quiesce state of a data set associated with a file being processed by CICS BAC. This can occur during CICS BAC startup processing in the CICS region, or when CICS BAC is processing a CICS BAC batch request utility command in the CICS region. *quiescestate* is the value in decimal form for the QUIESCESTATE parameter used with the command. The meaning of this value is given in *Appendix A* in the *CICS System Programming Reference*.

System action: Processing continues.

User response: None

Destination: Audit log

CBKxx8585I *seqnum* **SET TDQUEUE(X'***tdqname***'), OPENSTATUS(***openstatus***), ENABLESTATUS(***enablestatus***)**

Explanation: CICS BAC writes message CBK*xx*8585I to the CICS BAC audit log when CICS BAC issues an EXEC CICS SET TDQUEUE command to change the state of a transient data queue. This can occur during CICS BAC startup processing in the CICS region, or when CICS BAC is processing a CICS BAC batch request utility command in the CICS region. *tdqname* contains the hexadecimal representation of the transient data queue name being processed. *openstatus* and *enablestatus* are the values in decimal form for the OPENSTATUS and ENABLESTATUS parameters,

respectively, used with the command. The meaning of these values are given in *Appendix A* in the *CICS System Programming Reference*.

System action: Processing continues.

User response: None

Destination: Audit log

CBKxx8586I *seqnum* **SET PROGRAM(X'***programname***'), STATUS(***enablestatus***)**

Explanation: CICS BAC writes message CBK*xx*8586I to the CICS BAC audit log when CICS BAC issues an EXEC CICS SET PROGRAM command to change the state of a program. This can occur during CICS BAC startup processing in the CICS region, or when CICS BAC is processing a CICS BAC batch request utility command in the CICS region. *programname* contains the hexadecimal representation of the program name being processed. *enablestatus* is the value in decimal form for the STATUS parameter used with the command. The meaning of this value is given in *Appendix A* in the *CICS System Programming Reference*.

System action: Processing continues.

User response: None

Destination: Audit log

CBKxx8587I *seqnum* **SET TRANSACTION(X'***transid***'), STATUS(***enablestatus***)**

Explanation: CICS BAC writes message CBK*xx*8587I to the CICS BAC audit log when CICS BAC issues an EXEC CICS SET transaction command to change the state of a transaction ID. This can occur during CICS BAC startup processing in the CICS region, or when CICS BAC is processing a CICS BAC batch request utility command in the CICS region. *transid* contains the hexadecimal representation of the transaction ID being processed. *enablestatus* is the value in decimal form for the STATUS parameter used with the command. The meaning of this value is given in *Appendix A* in the *CICS System Programming Reference*.

System action: Processing continues.

User response: None

Destination: Audit log

CBKxx8599I *date time* **Start: CICS=***applid* **By={STARTUP|CICSSVR|BATCH}** *jobname stepname procname jobnumber sysid* **By={CICSSPI|LINKAPI}** *jobname userid termid transid*{**CC=***completioncode* **| AC=***abendcode***}**

Explanation: CICS BAC is performing one of the following actions:

By=STARTUP
CICS BAC is starting in a CICS region. The CICS BAC startup procedure identifies the CICS region by *jobname*, *stepname*, *procname*, and *jobnumber*, and it also displays the MVS *sysid*.

By=CICSSVR
The CICS BAC request server is processing a batch utility request. The CICS BAC request server identifies the batch request utility by its *jobname*, *stepname*, *procname*, and *jobnumber*, and it also displays the MVS *sysid*.

BY=BATCH
The CICS BAC batch request utility is servicing a command, or set of commands, itself because CICS BAC is not active in the target CICS region. The CICS BAC batch request utility identifies itself by its *jobname*, *stepname*, *procname*, *jobnumber*, and it also displays the MVS *sysid*. The CICS BAC batch request utility also includes the final *completioncode* (or the *abendcode* in the case of an abend).

BY=CICSSPI
The CICS BAC state monitor within the CICS region has detected a change to a CICS resource. The CICS BAC state monitor identifies the the task responsible for the change by its *userid*, *termid*, and *transid*. The CICS BAC state monitor also includes the final *completioncode*.

BY=LINKXPI
The CICS BAC request server is being driven by a CICS application program through the CICS BAC callable API. The CICS BAC request server identifies the task responsible for the request by its *userid*, *termid*, and *transid*. The CICS BAC request server also includes the final *completioncode* .

In all the above cases, the CICS BAC component displays the *date* and *time* of the request.

System action: None

User response: None

Destination: Audit log

CBKxx8600I *date time* **CICS BAC region startup in progress: mode={Set|Update} tran=***transid* **term=***termid* **user=***userid*

Explanation: CICS BAC startup processing is beginning in the CICS region. The type of startup is designated by the mode value as follows:

Set CICS BAC sets the state of all CICS resources on defined in the CICS BAC region control file to their last known states.

Update CICS BAC does not set resource states, but it updates the resource records in the CICS BAC as necessary to reflect the state of the resources.

System action: None

User response: None

Destination: Audit log

CBKxx8699I *date time* **CICS BAC region shutdown in progress: tran=***transid* **term=***termid* **user=***userid*

Explanation: CICS BAC is shutting down in the CICS region. The log record includes the transaction ID, terminal ID, and user ID (*transid*, *termid*, and *userid*) associated with the shutdown request. If the shutdown is taking place as a result of CICS going through shutdown PLT processing, the transaction ID is CPLT, the terminal id is blank, and the user ID is the default user ID for the CICS region.

System action: None

User response: None

Destination: Audit log

CBKxx8701I Calling pre-command user exit: *exitpgmname***.**

Explanation: CICS BAC is linking to the pre-command user exit program *exitpgmname*.

System action: None

User response: None

Destination: Audit log

CBKxx8702E Error linking to {pre-command | post-command} user exit: exit=*exitpgmname* **RESP=***eibresp* **RESP2=***eibresp2*

Explanation: CICS BAC has detected an error when linking to the user exit program *exitpgmname* . The errors reported by CICS from the EXEC CICS LINK request are EIBRESP *eibresp* and EIBRESP2 *eibresp2*.

System action: The batch request utility job step terminates immediately with return code 12.

User response: Contact technical support if you cannot determine the reason for the failure and correct it.

Destination: Audit log

CBKxx8707E {Pre-command | Post-command} GETMAIN for XPI area failed: RESP=*eibresp* **RESP2=***eibresp2* **length=***stglength*

Explanation: CICS BAC has detected an error when attempting to acquire storage for the commarea to be passed to the user exit program. The errors from the EXEC CICS GETMAIN request are EIBRESP *eibresp* and EIBRESP2 *eibresp2* and the storage length requested in teh exit program is *stglength*.

System action: The batch request utility job step terminates immediately with return code 12.

User response: Contact technical support if you cannot determine the reason for the failure and correct it.

Destination: Audit log

CBKxx8709I Exit *exitpgmname* **done: RC=***returncode* **AC=***abendcode* **CC=***completioncode* **ET=***errortext*

Explanation: CICS BAC has received a control back from the pre-command user exit program *exitpgmname*. *returncode*, *abendcode*, *completioncode*, and *errortext* contain the values returned by the user exit program.

System action: The action taken depends on the value returned in *returncode*. Depending on the return code, processing could continue, the current command could be ignored, the job step could be terminated, or the job step could be abended.

User response: None

Destination: Audit log

CBKxx8711I Calling post-command user exit: *exitpgmname***.**

Explanation: CICS BAC is linking to the post-command user exit program *exitpgmname*.

System action: None

User response: None

Destination: Audit log

CBKxx8719I Exit *exitpgmname* **done.**

Explanation: The post-command user exit program *exitpgmname* has returned control to CICS BAC.

System action: None

User response: None

Destination: Audit log

CBKxx8990E *pgmname* **detected severe error. Diagnostic data follows. Check console for more information.**

Explanation: CICS BAC has detected a severe error during startup processing while processing a resource, such as a file or a transient data queue.

System action: Processing terminates for the resource, and continues with the next resource on the CICS BAC control file. CBK*xx*8990E can be followed by one or more CBK*xx*8997E messages to provide additional diagnostic data.

User response: Contact technical support if you cannot determine the reason for the error and correct it.

Destination: Audit log

CBKxx8997E *pgmname resourcename errorcode*

Explanation: CBK*xx*8997E can follow message CBK*xx*8990E to provide additional diagnostic data when an error has been detected during CICS BAC startup processing in the CICS region.

System action: Processing is terminated for the resource, and continues with the next resource on the CICS BAC control file. CBK*xx*8997E can be followed by one or more messages providing additional diagnostic data.

User response: Contact technical support if the reason for the error cannot be determined and corrected.

Destination: Audit log

CBKxx8998E *pgmname filename errorcode datasetname*

Explanation: CBK*xx*8998E can follow message CBK*xx*8990E to provide additional diagnostic data when an error has been detected during CICS BAC startup processing in the CICS region.

System action: Processing is terminated for the resource, and continues with the next resource on the CICS BAC control file. CBK*xx*8998E can be followed by one or more messages providing additional diagnostic data.

User response: Contact technical support if the reason for the error cannot be determined and corrected.

Destination: Audit log

CBKxx8999E *pgmid resourcetype resourcename errorcode eibresp eibresp2 field1 field2 field3 field4 field5 field6 field7 field8*

Explanation: This message can follow message CBK*xx*8990E to provide additional diagnostic data when an error has been detected during CICS BAC startup processing in the CICS region.

System action: Processing is terminated for the resource, and continues with the next resource on the CICS BAC control file.

User response: Contact technical support if the reason for the error cannot be determined and corrected.

Destination: Audit log

CICS BAC general purpose messages (CBK*xx*9000–9999)

CBKxx9001E Internal error: code1(*code1*) code2(*code2*) data(*data*).

Explanation: CICS BAC has detected an unrecoverable error. *code1 code2* and *data* are internal CICS BAC codes that provide diagnostic information.

System action: The action taken depends on the error detected.

User response: Contact technical support.

Destination: Console or job log

CBKxx9101E CICS BAC unrecoverable error detected. Batch request utility abending with abend code *abend-code***.**

Explanation: The CICS BAC batch request utility has abnormally terminated as a result of an unrecoverable error.

System action: The batch request utility job step abnormally terminates with abend code *abend-code*.

User response: Contact technical support.

Destination: Console or job log

CBKxx9201W Unable to load the *language* **message table, Abend code=***abendcode***, Reason=***rsncode*

Explanation: The communication server or batch request utility is unable to load the message table module for the requested language *language*. *abendcode* is the abend code resulting from the LOAD request.*rsncode* is the reason code returned by the LOAD request.

System action: Processing continues using the US English language module.

User response: Ensure that the requested message table load module is available in the STEPLIB concatenation or link list concatenation. All message table module names are made up of the prefix CBKMT followed by the three character language code, for example CBKMTENU.

Destination: Job log

CBKxx9202I CICS Batch Application Control for z/OS message table loaded. Language=*language*

Explanation: The message table specified by *language* was successfully loaded and will be used for all messages for the component that loaded the message table. The message table chosen is the language specified by the LANGUAGE runtime parameter, or US English if the LANGUAGE parameter was not specified or the message table for the requested language could not be loaded.

System action: Processing continues.

User response: None.

Destination: Job log

CBKxx9501E Internal err: code1(*code1*) code2(*code2*). Request server subtask terminating with *abend-code* abend.

Explanation: An unrecoverable error has occurred in the CICS request server subtask causing it to terminate abnormally. *code1* and *code2* are CICS BAC internal codes that provide diagnostic information. *abend-code* is the user abend code.

System action: The CICS request server subtask terminates abnormally with abend code *abend-code*. CICS BAC is not started in the CICS region.

User response: Contact technical support.

Destination: Console or job log

Chapter 11. CICS BAC abend codes

This chapter explains the abend codes issued by CICS BAC.

1011

Explanation: The CICS BAC communication server TCP/IP listener encountered an unrecoverable TCP/IP API error and issued message CBK*xx*1509E, followed by CBK*xx*1510R requesting an operator response, to which the listener received an ABEND reply.

System action: The listener subtask issues CBK*xx*1513E acknowledging the ABEND reply, attempts to produce a dump, and terminates with a U1011 abend.

User response: See the associated console messages for additional information about the TCP/IP API error and how to correct it.

1012

Explanation: The CICS BAC communication server TCP/IP listener has exceeded its internal error count. This could imply that a recursive TCP/IP error is occurring.

System action: The CICS BAC TCP/IP listener subtask abnormally terminates and attempts to produce a dump.

User response: See the console messages CBK*xx*9001E and CBK*xx*1519E issued before the abend for more information on the cause of the error and how to correct it.

1042

Explanation: The CICS BAC communication server operator command subtask has encountered an unrecoverable internal error.

System action: The communication server address space attempts to produce a dump and then terminates.

User response: This abend is preceded by messages CBK*xx*9001E and CBK*xx*2519E. Examine those messages to obtain additional information regarding the error and what corrective action should be taken.

1099

Explanation: The CICS BAC communication server TCP/IP listener has received an invalid request from a client workstation. Either the client request is invalid or the requested object type is invalid.

System action: The TCP/IP listener subtask attempts to produce a dump then abnormally terminates.

User response: See message CBK*xx*1507E (invalid request) *or* CBK*xx*1516E (invalid object type) in the job log to determine which message was issued and what action you need to take to correct the problem.

1420

Explanation: The CICS BAC communication server is unable to build the control file table and cannot continue without this table.

System action: The CICS BAC communication server attempts to produce a dump and abnormally terminates.

User response: See the console message CBK*xx*1015E that precedes the abend for more information about the cause of the error.

1501

Explanation: The target CICS region for a batch request utility job step is not available, and CICS BAC did not perform its normal shutdown processing the last time it was active in the CICS region. The CICS region properties record for the region specify that, in this case, the CICS BAC batch request utility job step is to prompt the operator, in messages CBK*xx*5023W and CBK*xx*5024R, to reply with the action to take. CICS BAC has received an ABEND reply, issues message CBK*xx*5026E, and the job step abnormally terminates.

System action: The batch request utility job step abnormally terminates.

User response: You might need to restart the CICS region and CICS BAC in the region before running any batch request utility job steps. Alternatively, consider changing the region properties to allow the batch request utility job step to run and, optionally, reset the region CICS BAC shutdown flag.

1502

Explanation: The batch request utility has detected that the target CICS region control file is invalid. The batch request utility precedes the abend with message CBK*xx*5019E, together with other messages giving details of the control file problem.

System action: The batch request utility job step terminates.

User response: Check the job log for additional messages that explain the problem detected with the control file. If necessary, contact technical support.

1503

Explanation: The batch request utility is unable to locate a control file table member in the CBKPARMS data set or the MVS logical parmlib concatenation, or it located a member that is empty.

System action: The batch request utility job step issues relevant messages (for example, CBK*xx*1405E or CBK*xx*1406E) explaining the cause of the problem, and then abends.

User response: Check the job log for additional messages. Ensure that a non-empty control file table is available to the batch request utility job step and rerun the job step.

1504

Explanation: The batch request utility has detected an unrecoverable error and has issued messages CBK*xx*9001E and CBK*xx*9101E.

System action: The batch request utility job step abnormally terminates.

User response: Contact technical support.

1505

Explanation: The batch request utility has detected an unrecoverable error while attempting to read the target CICS region control file, and issued messages CBK*xx*8010E and CBK*xx*9101E with additional information about the error.

System action: The batch request utility job step abnormally terminates.

User response: Contact technical support.

1600

Explanation: The batch request utility has detected an unrecoverable error, and issued messages CBK*xx*9001E and CBK*xx*9101E.

System action: The batch request utility job step abnormally terminates.

User response: Contact technical support.

1610

Explanation: The batch request utility has detected an unrecoverable error, and issued messages CBK*xx*9001E and CBK*xx*9101E.

System action: The batch request utility job step abnormally terminates.

User response: Contact technical support.

1620

Explanation: The batch request utility has detected an unrecoverable error, and issued messages CBK*xx*9001E and CBK*xx*9101E.

System action: The batch request utility job step abnormally terminates.

User response: Contact technical support.

1630

Explanation: The batch request utility has detected an unrecoverable error, and issued messages CBK*xx*9001E and CBK*xx*9101E.

System action: The batch request utility job step abnormally terminates.

User response: Contact technical support.

1701

Explanation: The CICS BAC request server subtask has detected an unrecoverable error during CICS BAC startup processing, and issued message CBK*xx*9501E.

System action: The CICS BAC request server subtask abnormally terminates. The CICS region continues without CICS BAC support.

User response: Contact technical support.

1702

Explanation: The CICS BAC request server subtask has detected an unrecoverable error during CICS BAC startup processing, and issued message CBK*xx*9501E.

System action: The CICS BAC request server subtask abnormally terminates. The CICS region runs without CICS BAC support.

User response: Contact technical support.

1703

Explanation: The CICS BAC request server subtask has detected an unrecoverable error during CICS BAC startup processing, and issued message CBK*xx*9501E.

System action: The CICS BAC request server subtask abnormally terminates. The CICS region continues without CICS BAC support.

User response: Contact technical support.

1860

Explanation: The CICS BAC control file initialization utility has detected an unrecoverable error while trying to add the version record to the control file, and issued message CBK*xx*6060E.

System action: The CICS BAC control file

initialization utility terminates abnormally.

User response: Contact technical support if you cannot diagnose and correct the cause of the error.

1861

Explanation: The CICS BAC control file initialization utility has detected an unrecoverable error while trying to add the CICS region properties record, and issued message CBK*xx*6061E.

System action: The CICS BAC control file initialization utility terminates abnormally.

User response: Contact technical support if you cannot diagnose and correct the cause of the error.

1862

Explanation: The CICS BAC control file initialization utility has detected an unrecoverable error while trying to add the default file object record.

System action: The CICS BAC control file initialization utility issues message CBK*xx*6062E, and then terminates abnormally.

User response: Contact technical support if you cannot diagnose and correct the cause of the error.

1863

Explanation: The CICS BAC control file initialization utility has detected an unrecoverable error while trying to add the default program object record, and issued message CBK*xx*6062E.

System action: The CICS BAC control file initialization utility terminates abnormally.

User response: Contact technical support if you cannot diagnose and correct the cause of the error.

1864

Explanation: The CICS BAC control file initialization utility has detected an unrecoverable error while trying to add the default transient data queue object record, and issued message CBK*xx*6062E.

System action: The CICS BAC control file initialization utility terminates abnormally.

User response: Contact technical support if you cannot diagnose and correct the cause of the error.

1865

Explanation: The CICS BAC control file initialization utility has detected an unrecoverable error while trying to add the default transaction ID object record, and issued message CBK*xx*6062E.

System action: The CICS BAC control file initialization utility terminates abnormally.

User response: Contact technical support if you cannot diagnose and correct the cause of the error.

1866

Explanation: The CICS BAC control file initialization utility has detected an unrecoverable error while trying to open the control file data set, and issued message CBK*xx*6051E.

System action: The CICS BAC control file initialization utility terminates abnormally.

User response: Contact technical support if you cannot diagnose and correct the cause of the error.

1867

Explanation: The CICS BAC control file initialization utility has detected an unrecoverable error while trying to close the control file data set, and issued message CBK*xx*6051E.

System action: The CICS BAC control file initialization utility terminates abnormally.

User response: Contact technical support if you cannot diagnose and correct the cause of the error.

1868

Explanation: The CICS BAC control file initialization utility has found that the PARM string on the EXEC JCL statement does not contain valid CICS region applid, and issued message CBK*xx*6052E.

System action: The CICS BAC control file initialization utility terminates abnormally.

User response: Contact technical support if you cannot diagnose and correct the cause of the error.

1869

Explanation: The CICS BAC control file initialization utility has detected an error while trying to load the CICS BAC message table, and issued message CBK*xx*9001E.

System action: The CICS control file initialization utility abnormally terminates.

User response: Contact technical support.

1870

Explanation: The CICS BAC control file initialization utility has detected an unrecoverable error, and issued message CBK*xx*9001E

System action: The CICS control file initialization utility abnormally terminates.

User response: Contact technical support.

2001

Explanation: The CICS BAC region startup processor has detected an error. The startup processor is running during CICS initialization during PLTPI processing. Because, CICS BAC cannot start in the CICS region, CICS BAC issues a message that describes the problem, and follows this with message CBK*xx*4080R for the action to take. CICS BAC has received a CANCEL reply.

System action: The CICS region abnormally terminates

User response: Check the console and job log for additional messages (for example, CBK*xx*4060E, or one of the CBK*xx*407*n* messages) to help you determine the cause of the error. Contact technical support if you cannot diagnose and correct the cause of the error.

2002

Explanation: The CICS BAC region startup processor has detected an error during CICS initialization during PLTPI processing. This error prevents CICS BAC starting in the CICS region, and therefore it issues messages CBK*xx*4085W and CBK*xx*4086R. CICS BAC has received a CANCEL reply to CBKxx4080R.

System action: The CICS region abnormally terminates.

User response: Check the console and job log for additional messages to help determine the cause of the error. Contact technical support if you cannot diagnose and correct the cause of the error.

3001

Explanation: The CICS BAC batch request utility has encountered an unrecoverable error. The error is indicated in message CBM*xx*9001E.

System action: The batch request utility attempts to produce a dump and then abnormally terminates.

User response: Check the job log for any previous associated CBK*xx*9001E message for an explanation of the problem and any corrective active to be taken. If the problem persists, contact Technical Support.

3099

Explanation: The batch request utility is unable to obtain the storage it needs to establish its execution environment. Establishing this environment is the first step that the utility attempts to perform.

System action: The batch request utility attempts to produce a dump and then abnormally terminates.

User response: Increase the amount of virtual storage available to the batch request utility job step by increasing the value specified on the job step REGION parameter. If this fails to correct the problem, contact Technical Support.

3110

Explanation: The batch request utility has detected an unrecoverable error, and issued messages CBK*xx*9001E and CBK*xx*9101E.

System action: The batch request utility job step abnormally terminates.

User response: Contact technical support.

3111

Explanation: The batch request utility has detected an unrecoverable EXCI error, and issued message CBK*xx*8008E and, optionally, message CBK*xx*8009E.

System action: The batch request utility job step abnormally terminates.

User response: Contact technical support if the cause of the error cannot be determined and corrected.

3112

Explanation: The CICS BAC batch request utility is unable to establish CICS interregion communication (IRC) because the support is not available on the MVS image. See message CBK*xx*8007E for more information.

System action: The batch request utility job step abnormally terminates.

User response: Contact technical support if the cause of this cannot be determined and corrected.

3113

Explanation: The batch request utility has been notified that the target CICS region with which it was communicating has terminated before the request could be completed. See message CBK*xx*8013E for more information.

System action: The batch request utility job step abnormally terminates.

User response: Contact technical support if necessary.

Appendix A. The CICS BAC sample migration utility

This appendix describes the sample migration utility that CICS BAC provides to enable you to migrate from an existing application. The sample migration utility is called CBKMIGRT, and is supplied in the SCBKSAMP library.

If you have an existing product that performs a function the same as, or similar to, that provided by CICS BAC, you will be faced with the task of converting the commands for the existing product into the command syntax required by CICS BAC. To make this task as easy as possible for you, CICS BAC provides a sample utility that you can modify to migrate from any existing product. This appendix provides an overview of the sample migration utility, with some guidance on how you might approach the migration process.

Overview of the sample migration utility

The migration utility is a front-end program that, after converting command statements, can call the CICS BAC batch request utility. The CICS BAC sample migration utility is written in assembler, but you follow its design to write a similar utility in COBOL, REXX, or another suitable language. The following is a summary of the main steps the sample performs:

Obtain storage
This is working storage needed, for example, for control block fields such as file DCBs.

Process parameters
The sample migration utility accepts a parameter that can have one of two values:

CALLCICSBAC
This is the default, and indicates that you want the migration utility to call the CICS BAC batch request utility when it has completed migrating the input statements.

NOCALLCICSBAC
Indicates that you do not want the migration utility to call the CICS BAC batch request utility when it has finished migrating the input statements.

Open the print file
The migration utility uses the print file (DD name PRINT) to log its progress. It prints each statement in its original form and in its CICS BAC equivalent syntax. It also writes any error messages to this print file.

Open the input file
The migration utility opens the input file from which it is to read the existing statements that it is to convert and migrate to CBKIN. The sample procedure to run the utility shown in Figure 14 on page 187 shows the DD name of the input file as INPUT, but you might want to change this to the DD name of the existing product from which you are migrating.

Allocate and open CBKIN
The migration utility allocates and opens for output the CBKIN data set.

If the CBKIN data set is allocated to the job by a DD statement, it uses that existing data set. Use a predefined data set when you want to keep a permanent copy of the converted statements.

If the CBKIN data set is not allocated by a DD statement, the migration utility allocates a temporary data set for output. You can use a temporary data set in this way when the migration utility is to call the CICS BAC batch request utility and pass to it the statements migrated to CBKIN (the CALLCICSBAC option) and you do not want to keep a copy of the statements.

Allocate CBKPRINT
The migration utility dynamically allocates the print file, if it is not already allocated, to a SYSOUT data set, or uses an existing allocation for CBKPRINT. The print file is required later by the CICS BAC batch request utility.

Read input statements
The migration utility reads each input statement. At this point in its process, the sample program reads input statements with a fictitious syntax and writes an equivalent CICS BAC statement. You should replace this part of the program so that it can process the particular syntax of your existing product. The dummy syntax that the sample migration utility uses is explained within the source code of the program in SCBKSAMP. CBKMIGRT is fully described by comments within the source code, and it explains how you can change it to handle input statements of different syntaxes.

Close CBKIN
The migration utility closes the CBKIN file. This makes the migrated statements available for us by the CICS BAC batch request utility, if it is being called in this run of the migration utility.

Check for errors
The migration utility checks to see if any errors occurred during the migration process and if not, assumes that the migration succeeded and, if requested (by CALLCICSBAC) invokes the CICS BAC batch request utility. When control is returned from the CICS BAC batch request utility, all the migrated statements should have been processed.

Close
The migration utility performs its termination routines. It closes the input and print files and releases work area storage.

The sample migration utility contains all the instructions that you need to help you customize the program for your own needs. The actual routines that convert existing statements to CICS BAC syntax are in the form of subroutines that you call with parameters for the keywords and operand values, simplifying the whole process. There are also some switches at the front of the module that you can customize. These are designed to enable you to control things such as what SYSOUT class to allocate CBKPRINT to, what unit to allocate CBKIN to, and so on.

```
//*******************************************************************
//*                                                                 *
//* Copyright (c) 2004 HLA Software, LLC. All Rights Reserved.      *
//*                                                                 *
//*******************************************************************
//*                                                                 *
//* You can use the following procedure, suitably modified, to      *
//* execute the sample migration utility.                           *
//* To use this procedure, make the required changes and move it to *
//* an appropriate procedure library.                               *
//*                                                                 *
//* You can also convert this procedure into an inline set of JCL   *
//* statements to be run independently of a PROC.                   *
//*                                                                 *
//*   1. Make sure all the required data sets are defined and       *
//*      available.                                                 *
//*                                                                 *
//*   2. Change ALL the data sets names in lower case below to the  *
//*      names of the data sets at your installation.               *
//*                                                                 *
//* For more information, see the CICS BAC User's Guide.            *
//*                                                                 *
//*******************************************************************
//*                                                                 *
//CBKMGJCL PROC P=''
//*
//CBKMIGRT EXEC PGM=CBKMIGRT,PARM='&P'
//*
//STEPLIB    DD DISP=SHR,DSN=your.cicsbac.loadlib
//SYSUDUMP   DD SYSOUT=*
//PRINT      DD SYSOUT=*
//INPUT      DD DISP=SHR,DSN=your.statements.to.be.migrated
//         PEND
```

Figure 14. A procedure to run the sample migration utility

The CICS BAC sample migration utility is designed to read command statements of an existing product and generate the equivalent set of CICS BAC statements. This allows you to convert your existing input statements into the functionally equivalent CICS BAC statements, and allows the CICS BAC batch request utility to execute the migrated statements.

There are possibly two main approaches that you might want to consider when planning the migration to CICS BAC. The one you choose will probably depend upon the number of jobs you have to migrate.

Migrating a small number of jobs

If you have only a small number of jobs to migrate, consider migrating them all at once, perhaps over a weekend. With this approach, you can first run your customized migration utility against each set of input statements that needs to be migrated. Your migration utility would read the existing statements defined as input, and write the equivalent CICS BAC command statements to a new file, in a form ready to be used by the CICS BAC batch request utility. When you have created the equivalent CICS BAC commands, you can then update the JCL for each job, converting it read the CICS BAC commands file as CBKIN instead of the existing input file.

However, this approach is probably only suitable for a small number of jobs that you can handle in this way over a short period of time.

Migrating a large number of jobs

If the number of jobs to migrate to CICS BAC is too large to migrate them all at once, you probably need to adopt the alternative migration plan described here. In this plan, you invoke the migration utility from the JCL that drives the existing product from which you are migrating. Using this technique, the migration utility not only migrates your existing input statements into the CICS BAC equivalent, but it also sets up the environment for the CICS BAC batch request utility. For this purpose, the sample migration utility is written to enable it to operate as follows:

- Allocate dynamically the required data sets for DD names CBKIN and CBKPRINT
- Write the migrated statements to the CBKIN file
- Invoke the CICS BAC batch request utility to execute the migrated statements in CBKIN.

The final step above is done by issuing an MVS LINK macro to pass control to the CBKBMAIN program. This approach allows you to drive the CICS BAC batch request utility using your existing JCL when the migration utility has migrated the statements. With this technique, you can fully migrate subsets of your JCL to CICS BAC over a period of time, slowly eliminating the need for the migration utility.

Appendix B. CICS BAC national language support (NLS)

This appendix describes the CICS BAC national language support (NLS).

Overview

CICS BAC provides support for NLS in the communication server, the batch request utility, in the CICS component, and in the workstation administration client.

You can choose the language in which messages are issued to some destinations by specifying the appropriate national language code as a runtime parameter for the communication server and batch request utility (see "Host component NLS").

For the CICS region component, the language is determined by the language in operation at the time you start CICS BAC in the CICS region (see "Host component NLS").

You choose the national language for the workstation administration client by selecting the appropriate client module to download and install (see "CICS BAC workstation administration client NLS" on page 190).

Host component NLS

For CICS BAC host mainframe components, you specify the language you want use as follows:

- For the communication server, specify the LANGUAGE parameter as described under "Communication server runtime parameters" on page 27
- For the batch request utility, specify the LANGUAGE parameter as described under "CICS BAC batch request utility runtime parameters" on page 33.
- For CICS system console and job log messages, the language used is the language in effect for the CICS BAC region startup processor transaction at the time the CICS component starts.
  - If you start the CICS component during CICS initialization through a CICS PLTPI entry, CICS BAC uses the default language as specified by the CICS NATLANG system initialization parameter.
  - If you start the CICS component from a terminal using the KBKM transaction, the language used is the language in effect for the user signed on to the terminal at the time you issue the START command. If the terminal does not have a signed-on user, CICS BAC uses the default language for the region.
- For CICS BAC KBKM transaction maps, CICS BAC uses the language in effect for the user signed on to the terminal when the transaction is issued. If there is no user signed on at the terminal, CICS BAC uses the default language for the CICS region.
- For audit log messages, CICS BAC writes all audit log messages in US English, regardless of the language in effect for the various components at the time the messages are written.

National languages supported by CICS BAC host components

Table 1 shows the languages supported by CICS BAC. In some cases, even though a language other than ENU (US English) is specified, messages might appear in English. This is because some message destinations, such as the system console, do not support the character sets of some languages. The following table provides the language used for various message destinations

Table 1. Summary of NLS support in the host components

Language parameter	System console messages	Job log messages	CBKPRINT messages	CICS maps
ENU	US English	US English	US English	US English
JPN	US English	US English	Japanese (DBCS)	Japanese (DBCS)

Note: The CICS BAC audit log messages are always in US English, regardless of the language in effect for the various components at the time the messages are written.

CICS BAC workstation administration client NLS

The CICS BAC workstation administration client also supports multiple national languages. CICS BAC provides this support by supplying multiple versions of the workstation administration client, each one using a different language. You determine the language you want to use on a particular workstation when you choose which client module to download from the host mainframe to the workstation. Table 2 describes the workstation administration client modules distributed with CICS BAC in the SCBKDWLD data set.

Table 2. National languages in the workstation administration client

SCBKDWLD member	Language
CBKWCENU	US English
CBKWCJPN	Japanese

See the *CICS BAC Workstation User's Guide* for more information about downloading and installing the client.

Bibliography

Information to come

Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully.

CICS BAC provides accessibility features though its 3270 interface, which provides screen-reader accessible 3270 textual displays and a keyboard-only interface. Note that CICS BAC provides help and other related information using the standard screen overlay technique used by ISPF and other 3270 products. You might need to configure scripts within your screen reader to simplify the reading of these overlays.

Notices

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

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

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

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

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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

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

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

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

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Pentium is a trademarks of Intel Corporation in the United States, other countries, or both.

Sending your comments to IBM

About this task

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To ask questions, make comments about the functions of IBM products or systems, or to request additional publications, contact your IBM representative or your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail, to this address:

  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  SO21 2JN
  United Kingdom
- By fax:
  - From outside the U.K., after your international access code use 44–1962–816151
  - From within the U.K., use 01962–816151
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink™: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

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

Index

Numerics

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

W

X

SC34-6321-06