CICS Transaction Server for z/OS 5.4

Resource Reference

Note

Before using this information and the product it supports, read the information in “Notices” on page 453.

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

© Copyright International Business Machines Corporation 1974, 2023.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this PDF ix

Chapter 1. Resource definition attributes 1

ATOMSERVICE resource definitions 1
ATOMSERVICE attributes 1
Installing ATOMSERVICE resource definitions 4
ATOMSERVICE resource definitions 4
BUNDLE resources 4
BUNDLE attributes 5
BUNDLE resource definitions 6
CONNECTION resources 6
CONNECTION attributes 9
Installing connection definitions 19
ISC/MRO connection resource definitions 20
DB2CONN resources 20
DB2CONN attributes 20
Installing DB2 connection definitions 30
DB2 connection resource definitions 31
DB2ENTRY resources 31
DB2ENTRY attributes 32
Installing DB2 entry definitions 36
DB2 entry resource definitions 37
DB2TRAN resources 37
DB2TRAN attributes 37
Installing DB2 transaction definitions 39
DB2 transaction resource definitions 39
DOCTEMPLATE resources 39
DOCTEMPLATE attributes 40
Document template resource definitions 43
ENQMODEL resources 43
ENQMODEL attributes 44
Installing enqueue model definitions 45
Enqueue model definitions 46
FEPI node list resource definitions 46
FEPI pool resource definitions 46
FEPI property set resource definitions 46
FEPI target list resource definitions 46
FILE resources 46
Remote files 47
Coupling facility data tables 47
Shared data tables 48
FILE attributes 49
Installing file definitions 64
FILE resource definitions 65
IPCONN resources 65
IPCONN attributes 66
Installing IPCONN definitions 74
IPCONN resource definitions 75
JOURNALMODEL resources 75
JOURNALMODEL attributes 75

The default JOURNALMODEL....79
Journal model resource definitions....80
Examples....80
JVMSERVER resources....81
JVMSERVER attributes....81
JVMSERVER resource definitions....83
LIBRARY resources....83
LIBRARY attributes....84
Installing a LIBRARY resource definition by using CEDA....87
LIBRARY resource definitions....88
LSRPOOL resources....88
LSRPOOL attributes....88
LSR pool resource definitions....93
MAPSET resources....93
MAPSET attributes....94
Map set resource definitions....95
MQCONN resources....95
MQCONN attributes....96
MQCONN resource definitions....99
MQMONITOR resources....99
MQMONITOR attributes....100
MQMONITOR resource definitions....102
PARTITIONSET resources....102
PARTITIONSET attributes....103
Partition set resource definitions....105
PARTNER resources....105
PARTNER attributes....105
Installing partner definitions....107
Partner resource definitions....108
PIPELINE resources....108
PIPELINE attributes....109
Pipeline resource definitions....111
PROCESSTYPE resources....111
PROCESSTYPE attributes....112
PROCESSTYPE definitions....114
PROFILE resources....114
PROFILE attributes....114
Profile resource definitions....120
PROGRAM resources....120
PROGRAM attributes....121
PROGRAM resource definitions....131
SESSIONS resources....131
SESSIONS attributes....132
Installing sessions definitions....141
Session resource definitions....141
TCPIPSERVICE resources....141
TCPIPSERVICE attributes....142
TCPIPSERVICE: interrelated attributes....153
TCPIPSERVICE resource definitions....154
TDQUEUE resources....154
Dual-purpose resource definition for transient data....154
TDQUEUE attributes....155
Required TDQUEUE definitions....167
Installing transient data queue definitions....170
Replacing existing transient data queue definitions....171
Disabling transient data queues....171
Using dummy transient data queues....171
Transient data queue resource definitions....172

TERMINAL resources....172
Terminals for printing....172
Pipeline terminal definitions....174
Devices with LDC lists....175
APPC (LUTYPE6.2) single session terminal....176
Terminals for transaction routing....176
TERMINAL attributes....181
Installing terminal definitions....191
Checking terminal definitions....191
Terminal resource definitions....192
TRANCLASS resources....192
TRANCLASS attributes....192
Transaction class definitions....194
TRANSACTION resources....194
TRANSACTION attributes....196
Transaction resource definitions....211
TSMODEL resources....211
TSMODEL attributes....212
Temporary storage model definitions....216
TYPETERM resources....216
Default values for TYPETERM attributes....217
Devices supported....225
TYPETERM attributes....230
Typeterm resource definitions....254
URIMAP resources....254
URIMAP: related resources....256
URIMAP attributes....257
Installing URIMAP resource definitions....270
URIMAP resource definitions....271
WEBSERVICE resources....271
WEBSERVICE attributes....272
Installing WEBSERVICE resource definitions....274
Web service resource definitions....274

Chapter 2. Resource management transaction CEDA commands....275
The CEDA ADD command....275
The CEDA ALTER command....277
The CEDA APPEND command....280
The CEDA CHECK command....281
The CEDA COPY command....283
The CEDA DEFINE command....287
The CEDA DELETE command....290
The CEDA DISPLAY command....293
The CEDA DISPLAY GROUP command....295
The CEDA DISPLAY LIST command....295
The CEDA EXPAND command....297
The CEDA EXPAND GROUP command....299
The CEDA EXPAND LIST command....300
The CEDA INSTALL command....301
The CEDA LOCK command....304
The CEDA MOVE command....306
The CEDA REMOVE command....309
The CEDA RENAME command....311
The CEDA UNLOCK command....313
The CEDA USERDEFINE command....315
The CEDA VIEW command....319

Chapter 3. Resource management utility DFHCSDUP commands.....323
The DFHCSDUP ADD command..... 323
The DFHCSDUP ALTER command..... 325
The DFHCSDUP APPEND command.....327
The DFHCSDUP COPY command..... 329
The DFHCSDUP DEFINE command.....332
The DFHCSDUP DELETE command.....334
The DFHCSDUP EXTRACT command..... 337
The DFHCSDUP INITIALIZE command.....338
The DFHCSDUP LIST command..... 338
The DFHCSDUP PROCESS command.....340
The DFHCSDUP REMOVE command..... 341
The DFHCSDUP SCAN command..... 342
The DFHCSDUP SERVICE command.....345
The DFHCSDUP UPGRADE command.....345
The DFHCSDUP USERDEFINE command.....347
The DFHCSDUP VERIFY command..... 349

Chapter 4. DFHCSDUP: syntax and command rules.....351

Chapter 5. Resource definition macros..... 353
CLT: command list table..... 353
Control section: DFHCLT TYPE=INITIAL..... 353
Specifying alternate systems: DFHCLT TYPE=LISTSTART.....354
Specifying takeover commands: DFHCLT TYPE=COMMAND.....355
Messages to the operator: DFHCLT TYPE=WTO.....355
Closing the command list: DFHCLT TYPE=LISTEND..... 356
PDIR: DL/I directory..... 356
Control section: DFHDLPSB TYPE=INITIAL..... 356
Program specification blocks: DFHDLPSB TYPE=ENTRY.....356
FCT: file control table.....357
Control section: DFHFCT TYPE=INITIAL.....357
Local filesâ€”DFHFCT TYPE=FILE.....358
DFHFCT example.....365
MCT: monitoring control table.....365
Control section: DFHMCT TYPE=INITIAL.....366
User event monitoring points: DFHMCT TYPE=EMP..... 369
Control data recording: DFHMCT TYPE=RECORD.....372
DFHMCT examples.....386
PLT: program list table.....387
Control section: DFHPLT TYPE=INITIAL..... 388
Entries in program list table: DFHPLT TYPE=ENTRY.....388
DFHPLT example..... 389
RST: recoverable service table..... 390
Control section: DFHRST TYPE=INITIAL.....391
Recoverable service elements: DFHRST TYPE=RSE.....391
DBCTL subsystems: DFHRST TYPE=SUBSYS.....391
DFHRST example..... 392
SRT: system recovery table..... 392
Control section: DFHSRT TYPE=INITIAL.....392
Abend codes: DFHSRT TYPE=SYSTEM|USER.....393
DFHSRT example..... 394
TCT: terminal control table.....394
DFHTCT macro types..... 395
DFHTCT logical device codes: z/OS Communications Server non-3270..... 398
Sequential devices..... 405

Remote terminals for transaction routing..... 410
DFHTCT: CICS terminals list..... 412
TLT: terminal list table..... 414
Control section: DFHTLT TYPE=INITIAL..... 414
Entries in terminal list table: DFHTLT TYPE=ENTRY..... 415
DFHTLT example..... 416
TST: temporary storage table..... 416
Control section: DFHTST TYPE=INITIAL..... 418
Recoverable temporary storage: DFHTST TYPE=RECOVERY..... 418
Local temporary storage: DFHTST TYPE=LOCAL..... 420
Remote temporary storage: DFHTST TYPE=REMOTE..... 420
Temporary storage security checking: DFHTST TYPE=SECURITY..... 421
Temporary storage data sharing: DFHTST TYPE=SHARED..... 422
DFHTST example..... 423
XLT: transaction list table..... 423
Control section: DFHXLT TYPE=INITIAL..... 424
Entries in transaction list table: DFHXLT TYPE=ENTRY..... 424
DFHXLT example..... 425

Chapter 6. Obsolete attributes retained for compatibility..... 427

Chapter 7. CICS-supplied resource definitions, groups, and lists..... 429
DFHLIST definitions..... 429
CICS-supplied groups not in DFHLIST..... 436
CICS-supplied compatibility groups..... 437
Sample application program groups..... 437
TYPETERM definitions in group DFHTYPE..... 441
Model TERMINAL definitions in group DFHTERM..... 445
PROFILE definitions in group DFHEP..... 448
PROFILE definitions in group DFHISC..... 448
PROFILE definitions in group DFHSTAND..... 449
Model definitions in group DFHPGAIP..... 451

Notices..... 453

Index..... 459

About this PDF

This PDF is a reference of the attributes of CICS resources. To find out how to define these resources, see the PDF called *Configuring CICS TS for z/OS*. Before CICS TS V5.4, the information in this PDF was in the *Resource Definition Guide*.

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

Date of this PDF

This PDF was created on 2024-01-04 (Year-Month-Date).

© Copyright IBM Corp. 1974, 2023

Chapter 1. Resource definition attributes

Most resources can be defined using CICS RDO or CICSPlex SM Business Application Services (BAS).

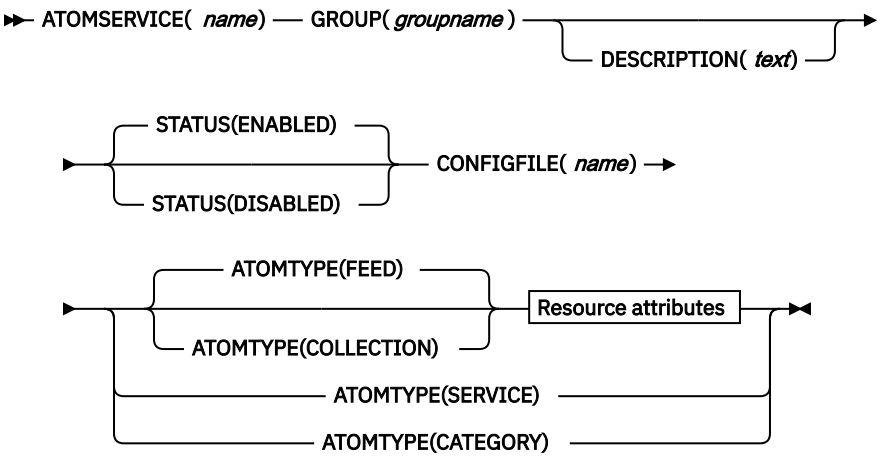
ATOMSERVICE resource definitions

An ATOMSERVICE resource defines an Atom service, a feed, a collection, or a category document, and identifies the resources that are used to supply the data for the feed.

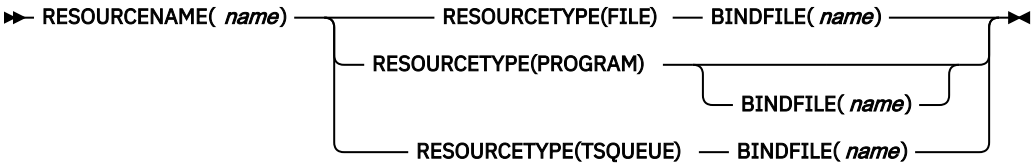
ATOMSERVICE attributes

Describes the syntax and attributes of the ATOMSERVICE resource.

Syntax



Resource attributes



Attributes

ATOMSERVICE(*name*)
Specifies the 8-character name of this resource definition.

Acceptable characters:
`A-Z 0-9 $ @ #`
Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

ATOMTYPE({FEED|SERVICE|COLLECTION|CATEGORY})
Specifies the type of Atom document that is returned for this ATOMSERVICE definition.

© Copyright IBM Corp. 1974, 2023

CATEGORY
An Atom category document, which lists the categories for entries in a collection. You can set up a category document if you want to use the same categories to define multiple collections.

COLLECTION
An Atom collection document, which contains a group of Atom entries that can be edited by Web clients using HTTP POST, PUT, and DELETE requests, as well as being retrieved using HTTP GET requests. The Atom configuration file for a collection must begin with the root element <cics:atomservice type="collection">.

FEED
An Atom feed document, which describes the metadata for a feed, and contains Atom entries that provide data for the feed. An Atom feed can be retrieved using HTTP GET requests, but it cannot be edited by a Web client. The Atom configuration file for an Atom feed must begin with the root element <cics:atomservice type="feed">.

SERVICE
An Atom service document, which provides information about the editable collections that are available on a server.

BINDFILE(*name*)
Specifies the fully qualified (absolute) or relative name of an XML binding stored in z/OS UNIX System Services. This attribute is not used for an Atom service or category document. You create an XML binding using the CICS XML assistant program DFHLS2SC.

For resource types FILE and TSQUEUE, the XML binding is required, and it specifies the data structures used by the resource named in RESOURCENAME, which supplies the data for the Atom document.

For resource type PROGRAM, an XML binding is optional, and you create it using the resource that the program accesses to obtain the data for the Atom entries, not the program itself. You must specify an XML binding for resource type PROGRAM if you are using the resource handling parameters in the DFHATOMPARMS container to pass information from the Atom configuration file to the program. If you are not doing this, do not specify an XML binding.

The name of the XML binding can be specified as an absolute path including all directories and beginning with a slash, for example, `/u/atom/atomictest.xsdbind`. Alternatively, it can be specified as a path relative to the HOME directory of the CICS region user ID; for example, `atom/atomictest.xsdbind` (with no leading forward slash). Up to 255 characters can be used.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

CONFIGFILE(*name*)
Specifies the fully qualified (absolute) or relative name of an Atom configuration file stored in z/OS UNIX System Services. The Atom configuration file contains XML that specifies metadata and field names for the Atom document that is returned for this resource definition. For details, see Creating an Atom configuration file.

The name can be specified as an absolute path including all directories and beginning with a slash, for example, `/u/atom/myfeed.xml`. Alternatively, it can be specified as a path relative to the HOME directory of the CICS region user ID, for example, `atom/myfeed.xml` (with no leading forward slash). Up to 255 characters can be used.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

RESOURCENAME(*name*)
Specifies the 1 - 16 character name of the CICS resource that provides the data for this Atom feed or collection. If the data for your Atom entries is held in a resource that is accessed by a service routine, specify the name of the service routine here. This attribute is not used for an Atom service or category document.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

RESOURCETYPE({FILE|PROGRAM|TSQUEUE})
Specifies the type of CICS resource that holds the data for this Atom feed or collection. This attribute is not used for an Atom service or category document. TSQUEUE is the default for this attribute.

FILE
A CICS file. A single record in the file provides the data for a single Atom entry. A file that holds Atom entries must have a unique key for the records, and you cannot use an alternate index file that has been defined with the NONUNIQUEKEY attribute. You can use any type of VSAM file to hold Atom entries, but note that ESDS (entry-sequenced data set) files are not suitable for a feed that you might want to set up as an editable collection, because you cannot delete records in an ESDS. You cannot use a BDAM file.

PROGRAM
A service routine, which is a CICS application program written to supply content for Atom entries.

TSQUEUE
A temporary storage queue. A single record in the temporary storage queue provides the data for a single Atom entry.

STATUS({ENABLED|DISABLED})
Indicates whether the Atom document specified by this resource definition is available or unavailable.

Installing ATOMSERVICE resource definitions

This procedure uses the CEMT and CEDA transactions to install an ATOMSERVICE resource definition. If the ATOMSERVICE resource already exists, it has to be disabled before it can be reinstalled.

Procedure

1. If the ATOMSERVICE resource already exists, ensure that it is disabled.

   Use the following command:

   ```
   CEMT SET ATOMSERVICE(name) DISABLED
   ```

   While the ATOMSERVICE resource is disabled, if a web client makes an HTTP request that requires the resource, CICS returns an HTTP 503 response (Service Unavailable) to the web client.
2. Install the ATOMSERVICE definition.

   Use the following command:

   ```
   CEDA INSTALL GROUP(groupname) ATOMSERVICE(name)
   ```
3. Optional: When you have successfully installed the ATOMSERVICE definition, use CEMT to enable the resource.

   Perform this step only if the ATOMSERVICE resource is not already defined as ENABLED, and you want to make the resource available for web clients. Use the following command:

   ```
   CEMT SET ATOMSERVICE(name) ENABLED
   ```

ATOMSERVICE resource definitions

ATOMSERVICE resource definitions specify the resources that CICS uses to produce an Atom service, feed, collection, or category document.

BUNDLE resources

A BUNDLE resource defines a CICS bundle, a unit of deployment for an application. A bundle is a collection of CICS resources, artifacts, references, and a manifest that you can deploy into a CICS region to represent a whole application or a component of an application.

The manifest is a file that describes the contents of the bundle, including what resources to create in the CICS region and the location of the supporting artifacts, what prerequisites are required for the application to run successfully, and any services that the application can offer to other applications.

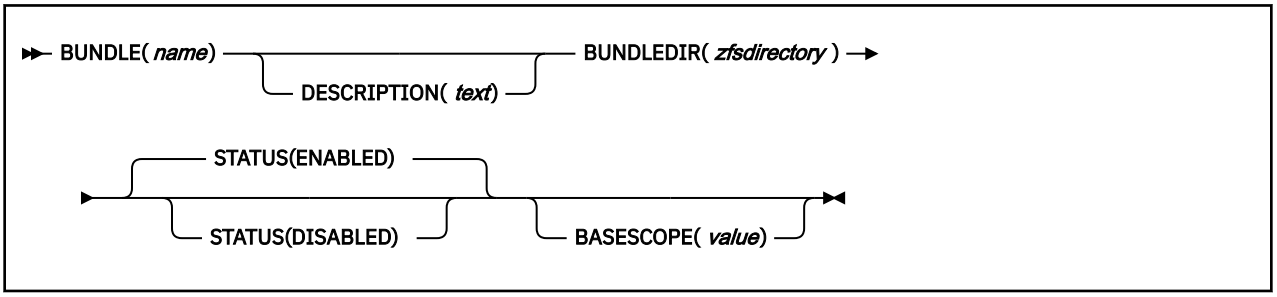
A bundle is deployed to z/OS UNIX and comprises a directory structure of artifacts. The BUNDLE resource defines where the bundle is deployed on z/OS UNIX and its status. When you enable a BUNDLE resource, CICS reads the manifest and dynamically creates the application resources that are defined in the manifest for you. Other resources that are defined as prerequisites for the application must be present in the CICS region to successfully enable the BUNDLE resource.

When you use CICS bundles, the lifecycle of the installed resources follows the lifecycle of the bundle; if you make available or unavailable, enable or disable, or discard a CICS bundle, the same action is applied to all resources that were dynamically created by the CICS bundle. For information about the implications of packaging resources in CICS bundles, see Characteristics of bundled resources.

Note: When you have created and deployed your bundle using the CICS Explorer®, you should ensure proper management of the bundle source code. The bundle cannot be reconstructed from the exported data in zFS, and a failure of your workstation would cause the data to be lost. You can use the Export function of CICS Explorer to export the bundle and check it in to a source code management system.

BUNDLE attributes

Describes the syntax and attributes of the BUNDLE resource.



BASESCOPE(*value*)
Specifies the 1 - 255 character string that defines the scope for a bundle. Use a uniform resource identifier (URI) where possible. Use the BASESCOPE attribute when you want to group similar bundles together or when you want to associate a CICS bundle with a particular application that is running in a platform. The URI for an application has the following format:

```
cicsapplication://Platform/ApplicationID/MajorVersion/MinorVersion/MicroVersion
```

Platform is the name of the platform in which the application is running, *ApplicationID* is the ID of the application bundle, followed by the major, minor, and micro version of the application.

The default value is empty so that all BUNDLE resources install in the same scope. See Scoping of bundles for more information.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

BUNDLE(*name*)
Specifies the 1 - 8 character name of the BUNDLE.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

It is preferable not to use names that start with DFH because these characters are reserved for use by CICS.

CICS can delete or discard them and replace them with CICS resources when you upgrade or provide new function or fixes.

BUNDLEDIR(*zfsdirectory*)
Specifies the 1 - 255 character fully qualified name of the root directory for the bundle on z/OS UNIX. The first character must be / and the end character /.

The value specified must be a valid name for a UNIX file:

- It must not contain embedded space characters.
- It must not contain consecutive instances of the / character.
- It is case-sensitive.

Acceptable characters:

```
A-Z a-z 0-9 . / _ # @ -
```

DESCRIPTION(*text*)
In this field, you can provide a description of the resource that you are defining. The description text can be up to 58 characters in length. No restrictions apply to the characters that you can use. However, if you use parentheses, ensure that each left parenthesis has a matching right one. If you use the CREATE command, for each single apostrophe in the text code, use two apostrophes.

STATUS(ENABLED|DISABLED)
Specifies the initial status of the BUNDLE resource when it is installed.

ENABLED
The BUNDLE is available for use. CICS checks that all prerequisites for the bundle are present in the region and attempts to install and enable all of the required resources that are defined in the bundle.

DISABLED
The BUNDLE is not available for use. CICS does not check for prerequisites and installs, but does not enable, the resources that are defined in the bundle.

BUNDLE resource definitions

BUNDLE resource definitions describe bundles, the unit of deployment for an application. The BUNDLE resource defines where the bundle is deployed on z/OS UNIX and its status.

CONNECTION resources

A CONNECTION defines a remote system with which your CICS system communicates, using *intersystem communication* (ISC) or *multiregion operation* (MRO).

ISC uses the APPC or LUTYPE6.1 communication protocol. MRO uses the IRC, XM, or XCF/MRO access method.

See also "IPCONN resources" on page 65. Like a CONNECTION, an IPCONN defines a communication link to a remote system, but in this case the connection uses the TCP/IP protocol.

When you define a CONNECTION, you give enough information to identify the system and specify its basic attributes. You put details in the SESSIONS definition about the sessions you use to communicate with the system. CICS uses the CONNECTION name to identify the other system when the definition has been installed. For other CICS systems connected via MRO, this name is typically the same as that specified in the other CICS system as the SYSIDNT system initialization parameter. For other systems connected via ISC, this name is typically based on an acronym that describes the location of or the organization that owns the system (for example, USA1 or IBMC).

The REMOTESYSTEM name on a TRANSACTION definition, or on a TERMINAL definition, can refer to a CONNECTION definition through its CONNECTION name (or to an IPCONN definition through its IPCONN name). These attributes are used for transaction routing.

The REMOTESYSTEM name on a PROGRAM definition can refer to a CONNECTION definition through its CONNECTION name (or to an IPCONN definition through its IPCONN name). This attribute is used for distributed program link.

The CONNECTION definition does **not** name associated SESSIONS.

Before you start creating definitions for intercommunication resources, see Defining intercommunication resources for further guidance. There you can find many useful examples of the attributes you must specify for different types of links and sessions.

Special considerations for different connection types are:

MRO links and sessions
You define an MRO link using one CONNECTION definition, and its associated parallel sessions using one SESSIONS definition.

ACCESSMETHOD
On the CONNECTION definition, specify this as IRC (for interregion communication), or XM (for cross-memory services). IRC is used to open and close the links.

PROTOCOL
On the SESSIONS definition, specify LU61 as the PROTOCOL. On the CONNECTION definition, leave the PROTOCOL value blank.

SENDPFX, SENDCOUNT, RECEIVEPFX, RECEIVECOUNT
In one SESSIONS definition, you specify a number of send sessions and a number of receive sessions. The values that you specify in these attributes are used to determine the names of the TCT entries created when the definition is installed. (See "Installing connection definitions" on page 19.)

APPC links and parallel sessions
For APPC, the sessions are grouped into modesets. You define each modeset with a SESSIONS definition, so you have as many SESSIONS definitions as you require modesets. You define the link as a CONNECTION definition. The following attributes are significant:

ACCESSMETHOD
On the CONNECTION definition, specify this as VTAM®.

MAXIMUM
Use this to control the number of sessions in the modeset.

MODENAME
On the SESSIONS definition for each modeset, name the modeset with the MODENAME. This is the name by which the modeset is known to CICS when the definition is installed in the active system.

PROTOCOL
On both the CONNECTION and SESSIONS definitions, specify APPC as the protocol.

APPC (LUTYPE6.2) single session terminal
You can define an APPC terminal as a CONNECTION-SESSIONS pair or as a TERMINAL-TYPETERM pair. The TERMINAL-TYPETERM method is described in "APPC (LUTYPE6.2) single session terminal " on page 176. If you want to use the CONNECTION-SESSIONS method, the following attributes are significant:

ACCESSMETHOD
On the CONNECTION definition, specify this as VTAM.

MAXIMUM
For a single session terminal, specifying 1,0 or 1,1 has the same effect. (For further information, see "CONNECTION attributes" on page 9.)

MODENAME
On the SESSIONS definition, specify the MODENAME. This is the name that CICS uses to identify the session when the definition is installed in the active system.

PROTOCOL
On both the CONNECTION and SESSIONS definitions, specify APPC as the protocol.

SINGLESESS
YES indicates that the CONNECTION definition is for a single session terminal.

LUTYPE6.1 links and sessions
LUTYPE6.1 links and sessions can be defined in one of two ways:

- In one CONNECTION and one SESSIONS definition
- In one CONNECTION and a number of SESSIONS definitions: one for each session needed

If your sessions are all to have **identical** attributes, define each link in one CONNECTION definition and all its associated sessions in one SESSIONS definition.

ACCESSMETHOD
On the CONNECTION definition, specify this as VTAM.

PROTOCOL
On the SESSIONS definition and on the CONNECTION definition, specify this as LU61.

RECEIVECOUNT, RECEIVEPFX, SENDCOUNT, SENDPFX
These attributes are used as for MRO links and sessions.

If your sessions are to have **different** attributes from each other, you must create a separate SESSIONS definition for each one. With the exception of NETNAMEQ, this method is the same as that for CICS-IMS sessions, described here.

Note: For CICS-CICS ISC links and sessions, you are recommended to use APPC rather than LUTYPE6.1.

LUTYPE6.1 CICS-IMS links and sessions
IMS needs each session to be defined in a separate SESSIONS definition, because each session must have a different NETNAMEQ.

You define the link as a CONNECTION definition, and create a number of SESSIONS definitions: one for each SEND session and one for each RECEIVE session.

ACCESSMETHOD
On the CONNECTION definition, specify this as VTAM.

NETNAMEQ
This is the name that the remote IMS system uses to identify the session.

PROTOCOL
On both the CONNECTION and SESSIONS definitions, specify LU61 as the protocol.

SESSNAME
This is the name that CICS uses to identify the session when the definition is installed in the active system.

RECEIVECOUNT
SENDCOUNT
Use these attributes to specify whether a session is a SEND session or a RECEIVE session.

A RECEIVE session is one in which the local CICS is the primary and is the contention loser. It is specified by defining RECEIVECOUNT(1) and leaving SENDCOUNT to default to blank. (You do not need to specify a SENDPFX or a RECEIVEPFX.)

A SEND session is one in which the local CICS is the secondary and is the contention winner. Specify it by defining SENDCOUNT(1) and leaving RECEIVECOUNT to default to blank.

INDIRECT connections
An INDIRECT connection is a remote system for which you have not defined a direct link with the local system. Instead, the two systems communicate with each other by way of one or more intermediate systems. You can use this method for transaction routing. The remote system, indirectly connected, is always the terminal-owning region; the local system is always the application-owning region or an intermediate region on the transaction routing path.

Indirect connections are required only if you use non-z/OS Communications Server terminals for transaction routing across intermediate systems. Optionally, you can use them with z/OS Communications Server terminals, where several transaction routing paths are possible, to identify the preferred path to the terminal-owning region. For information about why you might want to define indirect connections, and about the resources required for transaction routing, see Defining indirect links for transaction routing.

In the local system, you must have ordinary CONNECTION and SESSIONS definitions for the intermediate systems to which you are directly connected. The ACCESSMETHOD should be IRC or XM with PROTOCOL(LU61), or VTAM with PROTOCOL(APPC).

For the INDIRECT connection (also known as an indirect link or an indirect system) you need, in the local system, a CONNECTION definition only. You do not need a SESSIONS definition: the sessions that are used are those of the intermediate system. The following attributes of the CONNECTION definition are significant:

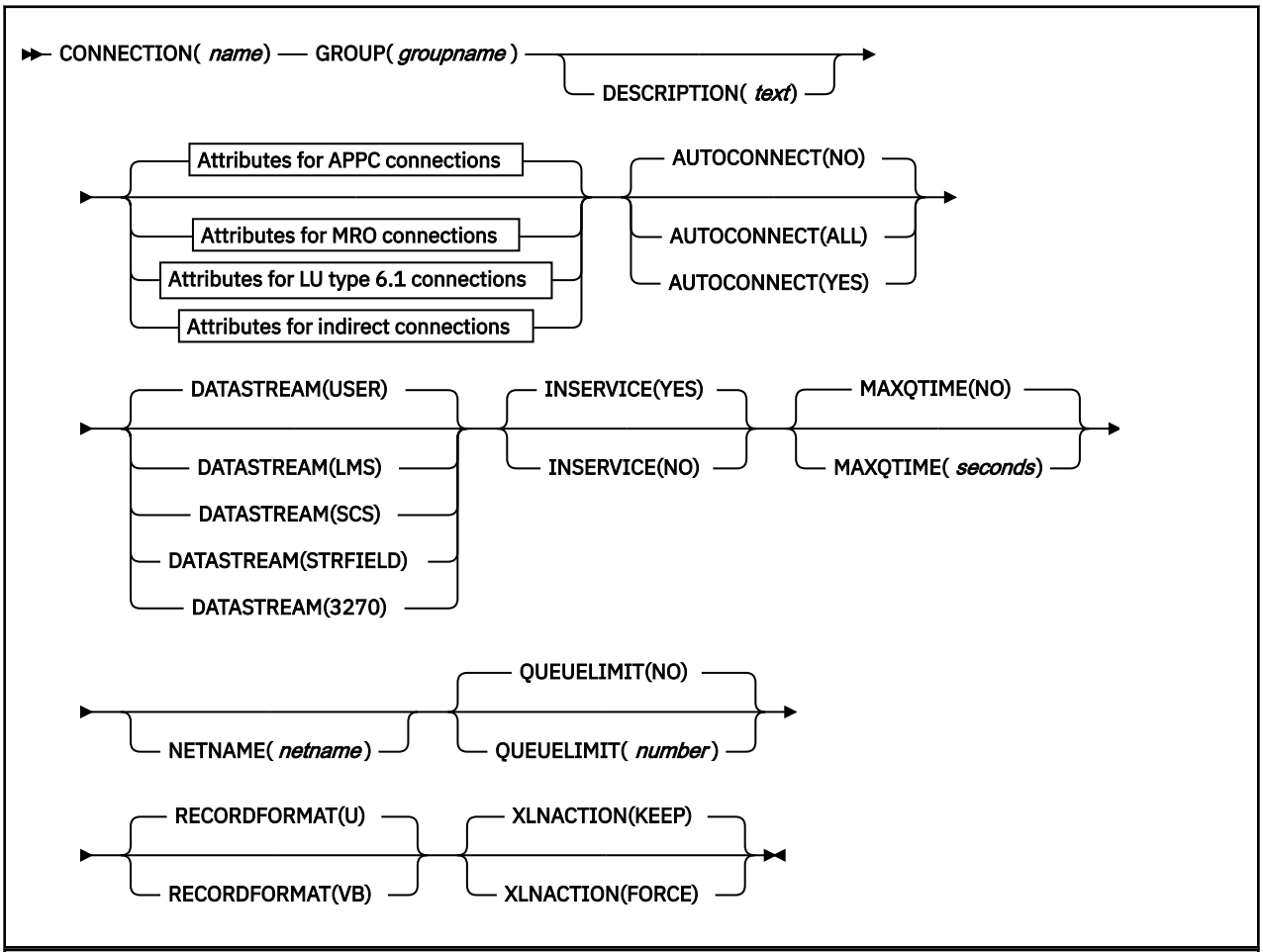
ACCESSMETHOD
Specify this as INDIRECT.

INDSYS
Specify the CONNECTION definition for the MRO or APPC link that is the start of a path to the terminal-owning system.

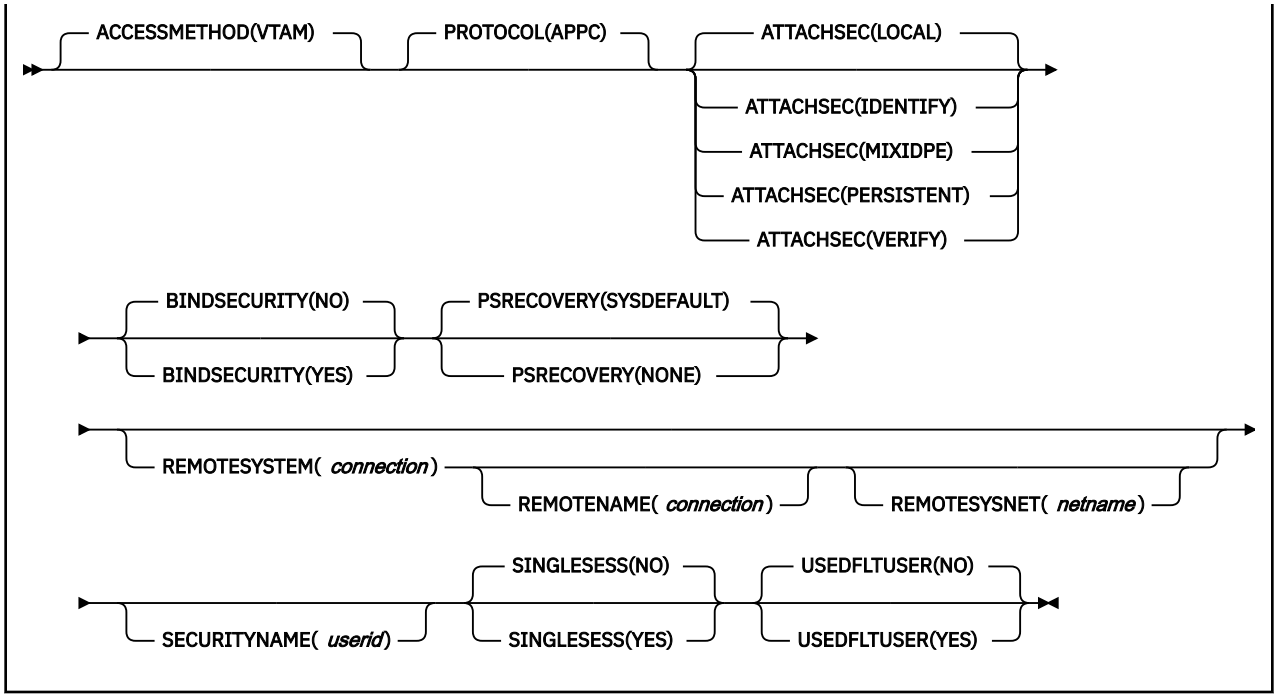
NETNAME
Specify the APPLID of the terminal-owning system.

CONNECTION attributes

Describes the syntax and attributes of the CONNECTION resource.

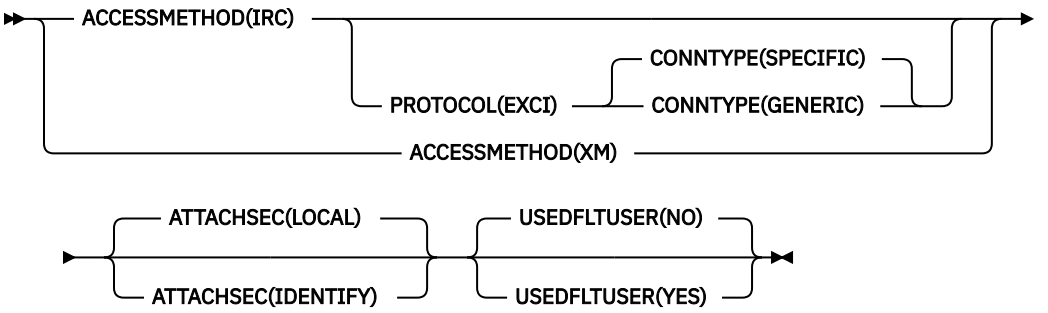


Attributes for APPC connections

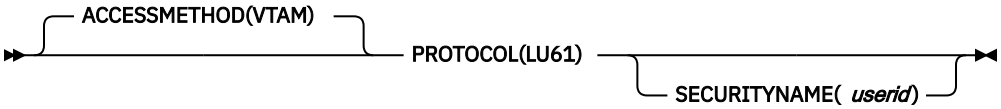


Note: VTAM is now z/OS Communications Server.

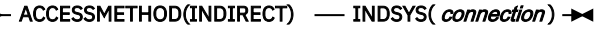
Attributes for MRO connections



Attributes for LU type 6.1 connections



Attributes for indirect connections



ACCESSMETHOD({VTAM|INDIRECT|IRC|XM})
specifies the access method to be used for this connection.

VTAM
Communication between the local CICS region and the system defined by this connection definition is through z/OS Communications Server. You can use z/OS Communications Server intersystem communication (ISC) for systems that are in different MVS™ images or in different address spaces in the same MVS image.

INDIRECT
Communication between the local CICS system and the system defined by this connection definition is through the system named in the INDSYS operand.

IRC
Communication between the local CICS region and the region defined by this connection definition is through the interregion communication (IRC) program DFHIRP, using the SVC (as opposed to cross-memory (XM)) mode of DFHIRP.

Note: This use of the term IRC is more specific than its general use.

You can use IRC for multiregion operation (MRO) for regions that are in the same MVS image or in different MVS images within a sysplex.

XM
MRO communication between the local CICS region and the region defined by its CONNECTION definition uses MVS cross-memory services. Initial connection is through the interregion communication (IRC) program DFHIRP, using the cross-memory (XM) (as opposed to the SVC) mode of DFHIRP. You can use XM for multiregion operation for regions that are in the same MVS image, or in different MVS images within a sysplex.

Note: The CICS type 3 SVC is still required with XM because DFHIRP is used when the link is opened. For further information about SVCs, see Installing the CICS SVCs in Installing.

MVS cross-memory services are used only if the ACCESSMETHOD of the other end of the link is also defined as XM.

If the MRO partners reside in different MVS images within a sysplex, and the CONNECTION specifies IRC or XM, CICS automatically uses XCF as the access method, and ignores the IRC or XM specification.

Note: You cannot define XCF explicitly; if you want to use XCF, you must specify IRC or XM. See Cross-system multiregion operation (XCF/MRO) for more information about XCF.

ATTACHSEC({LOCAL|IDENTIFY|VERIFY|PERSISTENT|MIXIDPE})
specifies the level of attach-time user security required for the connection.

IDENTIFY
Incoming attach requests must specify a user identifier. Enter IDENTIFY when the connecting system has a security manager; for example, if it is another CICS system.

LOCAL
CICS does not require the client to supply to supply a user identifier, or a password. All requests will run under the userid specified in the SECURITYNAME attribute. If the PROTOCOL attribute on the CONNECTION definition is LU6.1, you must specify LOCAL.

MIXIDPE
Incoming attach requests may be using either or both IDENTIFY or PERSISTENT security types. The security type used depends on the incoming attach request.

PERSISTENT
Incoming attach requests must specify a user identifier and a user password on the first attach request. Subsequent attach requests require only the user identifier. This should be used only between a programmable workstation, (for example, an IBM® Personal Computer) and CICS.

VERIFY
Incoming attach requests must specify a user identifier and a user password. Enter VERIFY when the connecting system has no security manager and hence cannot be trusted. Do not specify VERIFY for CICS-to-CICS communication, because CICS does not send passwords.

AUTOCONNECT({NO|YES|ALL})
For systems using ACCESSMETHOD(VTAM), you specify with AUTOCONNECT(YES) or (ALL) that sessions are to be established (that is, BIND is to be performed). Such sessions are set up during CICS initialization, or when you use the CEMT or EXEC CICS SET VTAM OPEN command to start communication with z/OS Communications Server. If the connection cannot be made at these times because the remote system is unavailable, you must subsequently acquire the link by using the CEMT

or EXEC CICS SET CONNECTION(sysid) INSERVICE ACQUIRED command, unless the remote system becomes available in the meantime and itself initiates communications.

For APPC connections with SINGLESESS(NO) specified, CICS tries to bind, on system start-up, the LU services manager sessions in mode group SNASVCMG.

For connection definitions with SINGLESESS(YES) specified, the AUTOCONNECT operand is ignored. Use the AUTOCONNECT operand of the session definition instead.

ALL
On this definition, ALL is equivalent to YES, but you can specify ALL to be consistent with the session definition.

AUTOCONNECT(ALL) should not be specified for connections to other CICS systems, because this can cause a bind-race.

NO
CICS does not attempt to bind sessions when the connection is established.

YES
CICS attempts to bind only contention-winning sessions when the connection is established.

The AUTOCONNECT option is not applicable on an LU6.1 connection definition. For LU6.1 connections, specify AUTOCONNECT(YES) on the SESSIONS definition if you want the connection to be established at initialization or CEDA install. Specify AUTOCONNECT(NO) on the SESSIONS definition if you do not want the connection to be established at initialization or CEDA install.

BINDPASSWORD
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS. For more information, see Obsolete attributes.

BINDSECURITY({NO|YES}) (APPC only)
specifies whether an ESM is being used for bind-time security.

NO
No external bind-time security is required.

YES
If security is active and the XAPPC system initialization parameter is set to YES, CICS attempts to extract the session key from RACF® in order to perform bind-time security. If no RACF profile is available, the bind fails.

CONNECTION(*name*)
specifies the name of this connection definition. The name can be up to four characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

This is the name specified as REMOTESYSTEM on file, terminal, transaction, and program definitions. You should not have a terminal definition and a connection definition with the same name.

CONNTYPE({SPECIFIC|GENERIC})
For external CICS interface (EXCI) connections, this specifies the nature of the connection.

GENERIC
The connection is for communication from a non-CICS client program to the CICS system, and is generic. A generic connection is an MRO link with a number of sessions to be shared by multiple EXCI users. For a generic connection you cannot specify the NETNAME attribute.

SPECIFIC
The connection is for communication from a non-CICS client program to the CICS region, and is specific. A specific connection is an MRO link with one or more sessions dedicated to a single user in a client program. For a specific connection, NETNAME is mandatory.

DATASTREAM({USER|3270|SCS|STRFIELD|LMS})
specifies the type of data stream.

LMS
The data stream is a Logical Message Services (LMS) data stream consisting of FMH4s and FMH8s as defined in the LUTYPE6.1 architecture.

SCS
The data stream is an SCS data stream as defined in the LUTYPE6.1 architecture.

STRFIELD
The data stream is a structured field data stream as defined in the LUTYPE6.1 architecture.

USER
Let DATASTREAM default to USER if the data stream is user-defined. If you are communicating between multiple CICS systems, always let DATASTREAM default to USER.

3270
The data stream is a 3270 data stream as defined in the type 6.1 logical unit (LUTYPE6.1) architecture.

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

INDSYS(*connection*)
specifies the name of another CONNECTION that defines an intermediate system used to relay communications between this system and the remote system. The name can be up to four characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

You may specify an intermediate system only if you specify ACCESSMETHOD(INDIRECT).

INSERVICE({YES|NO})
specifies the status of the connection that is being defined.

NO
The connection can neither receive messages nor transmit input.

YES
Transactions may be initiated and messages may automatically be sent across the connection.

MAXQTIME({NO|*seconds*})
specifies a time control on the wait time for queued allocate requests waiting for free sessions on a connection that appears to be unresponsive. The maximum queue time is used only if a queue limit is specified for QUEUELIMIT, and then the time limit is applied only when the queue length has reached the queue limit value.

NO
CICS maintains the queue of allocate requests that are waiting for a free session. No time limit is set for the length of time that requests can remain queued (though the DTIMOUT mechanisms can apply to individual requests). In this case, a value of X'FFFF' is passed on the XZIQUE parameter list (in field UEPEMXQT).

seconds
The approximate upper limit on the time that allocate requests can be queued for a connection that appears to be unresponsive. The number represents seconds in the range 0 through 9999.

CICS uses the maximum queue time attribute to control a queue of allocate requests waiting. When the number of queued allocate requests reaches the queue limit (QUEUELIMIT), and a new allocate request is received for the connection, if the rate of processing for the queue indicates that, on average, the new allocate takes more than the maximum queue time, the queue is purged, and message DFHZC2300 is issued. When the queue is purged, queued allocate requests return SYSIDERR.

No further queuing takes place until the connection has successfully freed a session. At this point, CICS issues DFHZC2301 and resumes normal queuing.

You can also control the queuing of allocate requests through an XZIQUE global user exit program. This allows you to use statistics provided by CICS, which report the state of the link. You can use these statistics, in combination with the queue limit and maximum queue time values you specify, to make more specialized decisions about queues.

The MAXQTIME value is passed to an XZIQUE global user exit program on the XZIQUE parameter list, if the exit is enabled. See XZIQUE exit for managing MRO and APPC intersystem queues for programming information about writing an XZIQUE global user exit program.

You can also specify the NOQUEUE|NOSUSPEND option on the ALLOCATE command to prevent an explicit request being queued. See CICS command summary for programming information about these API options.

NETNAME(*netname*)
specifies the network name that identifies the remote system. The name can be up to eight characters in length. The name follows assembler language rules. It must start with an alphabetic character.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

The NETNAME is the APPLID of the remote system or region, unless you are defining an LUTYPE6.1 or APPC link to a z/OS Communications Server generic resource group.

- If you are defining an LUTYPE6.1 link to a generic resource, NETNAME must specify the generic resource name, not the APPLID of one of the group members.

- If you are defining an APPC link to a generic resource, NETNAME can specify either the group's generic resource name or the APPLID (member name) of one of the group members. However, if you specify a member name, and this CICS is not itself a member of a CICS generic resource, the connection must always be acquired by this CICS ("this CICS " being the CICS region in which the connection definition is installed).

For z/OS Communications Server, the APPLID is the label of the remote VTAM VBUILD TYPE=APPL statement.

If you do not supply a NETNAME, the CONNECTION name is used by default.

There are some rules about duplicate NETNAMEs. You **cannot** have:

- Two or more APPC links with the same NETNAME
- An APPC link and an LUTYPE6.1 link with the same NETNAME
- Two or more IRC connections with the same NETNAME
- Two or more remote APPC connections with the same NETNAME.
- A remote APPC connection with the same NETNAME as any other connection or local terminal.

You **can** have:

- An IRC connection and an LUTYPE6.1 connection with the same NETNAME
- An IRC connection and an APPC connection with the same NETNAME
- Two or more LUTYPE6.1 connections with the same NETNAME
- Any connection with the same NETNAME as a remote terminal.

For connections that use the z/OS Communications Server LU alias facility:

- **APPC synclevel 1**: If the CICS region supports z/OS Communications Server dynamic LU alias (that is, LUAPFX=*xx* is specified on the CICS region's APPL statement) this NETNAME is assumed to be in the same network as the CICS region. If it is not the resource must have a local z/OS Communications Server CDRSC definition with LUALIAS=*netname* defined, where *netname* must match the NETNAME defined on this CONNECTION definition. Synclevel 1 APPC connections are generally work stations.

  Be aware that some synclevel 1 resources may become synclevel 2, depending on how they connect to CICS. For example, if TXSeries® does *not* use a PPC gateway, the connection is synclevel 1. If it does use a PPC gateway, it is synclevel 2.
- **APPC synclevel 2 and LUTYPE6.1**: This NETNAME is assumed to be unique. CICS matches it against the network name defined in the z/OS Communications Server APPL statement. These connections are generally CICS-to-CICS but could, for example, be TXSeries-connected through a PPC gateway.

Some rules about NETNAME and APPLID:

- – If an installed CONNECTION definition has the same name as an installed IPCONN definition, the APPLID of the IPCONN definition must match the NETNAME of the CONNECTION definition. If they do not, the message that results depends on the situation:
  - DFHIS3009 if the error is detected during IPCONN autoinstall
  - DFHAM4913 if the error is detected during IPCONN install
  - DFHZC6312 if the error is detected during CONNECTION install or autoinstall
- The IPCONN definition takes precedence over the CONNECTION definition: that is, if an IPCONN and a CONNECTION have the same name, CICS uses the IPCONN connection.
- a CONNECTION and an IPCONN with the same NETNAME and APPLID do not have to have the same name.

  This allows the possibility to use a distinct sysid for communication over TCP/IP rather than relying on the CICS default of routing all supported function via the IPCONN, if it exists.

PROTOCOL({APPC|LU61|EXCI|blank})
specifies the type of protocol that is to be used for the link.

APPC (LUTYPE6.2 protocol)
Advanced program-to-program communication, or APPC protocol. This is the default value for ACCESSMETHOD(VTAM). Specify this for CICS-CICS ISC.

blank
MRO between CICS regions. You must leave the PROTOCOL blank for MRO, and on the SESSIONS definition you must specify LU6.1 as the PROTOCOL.

EXCI
The external CICS interface. Specify this to indicate that this connection is for use by a non-CICS client program using the external CICS interface.

LU61
LUTYPE6.1 protocol. Specify this for CICS-CICS ISC or CICS-IMS ISC, but not for MRO.

PSRECOVERY({SYSDEFAULT|NONE})
In a CICS region running with persistent sessions support, this specifies whether, and how, LU6.2 sessions are recovered on system restart within the persistent session delay interval.

NONE
All sessions are unbound as out-of-service with no CNOS recovery.

SYSDEFAULT
If a failed CICS system is restarted within the persistent session delay interval, the following actions occur:

- User modegroups are recovered to the SESSIONS RECOVOPTION value.
- The SNASVCMG modegroup is recovered.
- The connection is returned in ACQUIRED state and the last negotiated CNOS state is returned

QUEUELIMIT({NO|*number*})
specifies the maximum number of allocate requests that CICS is to queue while waiting for free sessions:

NO
There is no limit set to the number of allocate requests that CICS can queue while waiting for a free session. In this case, a value of X'FFFF' is passed on the XZIQUE parameter list (in field UEPQUELM).

number
The maximum number of allocate requests, in the range 0 through 9999, that CICS can queue on the connection while waiting for a free session. When the number of queued allocate requests reaches this limit, subsequent allocate requests return SYSIDERR until the queue drops below the limit.

This queue limit is passed to an XZIQUE global user exit program on the XZIQUE parameter list if the exit is enabled.

You can also control the queuing of allocate requests through the MAXQTIME attribute, and through an XZIQUE global user exit program. See the MAXQTIME attribute for more information about controlling queues.

Note: BIND re-negotiation is not triggered, even if there are unused secondary sessions. Unless the CEMT SET MODE command is used to force re-negotiation, the queuelimit will come into play as soon as all the primary sessions are in use.

RECORDFORMAT({U|VB})
specifies the type of SNA chain.

U
Let RECORDFORMAT default to U if the SNA chain is a single, unblocked stream of data. You can have private block algorithms within the SNA chain. Let RECORDFORMAT default to U if you are communicating between multiple CICS systems.

VB
The SNA chain is formatted according to the VLVB standard as defined in the LUTYPE6.1 architecture.

REMOTENAME(*connection*)
specifies the name by which the APPC connection for transaction routing is known in the system or region that owns the connection. The name can be up to four characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

The remote system or region can be an APPC device (see APPC devices for transaction routing).

REMOTESYSNET(*netname*)
specifies the network name (APPLID) of the system that owns the connection. The name can be up to eight characters in length. It follows assembler language rules, and must start with an alphabetic character.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

Use REMOTESYSNET when transaction routing to remote APPC systems or devices, and there is no direct link between the region in which this definition is installed and the system that owns the connection to the remote device. You do not need to specify REMOTESYSNET if:

- You are defining a local connection (that is, REMOTESYSTEM is not specified, or specifies the sysid of the local system).
- REMOTESYSTEM names a direct link to the system that owns the connection. However, there is one special case: if the connection-owning region is a member of a z/OS Communications Server generic resources group and the direct link to it is an APPC connection, you may need to specify REMOTESYSNET. REMOTESYSNET is needed in this case if the NETNAME specified on the CONNECTION definition for the direct link is the generic resource name of the connection-owning region (not the applid).

REMOTESYSTEM(*connection*)
specifies the name that identifies the intercommunication link to the system that owns the connection. The name can be up to four characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

This is the CONNECTION name on the connection definition for the intercommunication link.

REMOTESYSTEM is used for transaction routing to remote APPC systems or devices. If it is not specified, or if it is specified as the sysid of the local system, this connection is local to this system.

If the name is that of another system, the connection is remote. You can therefore use the same definition for the connection in both the local system and a remote system.

If there are intermediate systems between this CICS and the region that owns the (connection to the) device, REMOTESYSTEM should specify the first link in the path to the device-owning region. If there is more than one possible path, it should specify the first link in the preferred path.

SECURITYNAME(*userid*)
For APPC and LU6.1 links only, this is the security name of the remote system.

In a CICS system with security initialized (SEC=YES), the security name is used to establish the authority of the remote system.

Note: If USERID is specified in the SESSIONS definition associated with the connection definition, it overrides the userid specified in the SECURITYNAME attribute, and is used for link security.

The security name (or USERID on the sessions definition) must be a valid RACF userid on your system. Access to protected resources on your system is based on the RACF user profile and its group membership.

SINGLESESS({NO|YES})
specifies whether the definition is for an APPC terminal on a single session APPC link to CICS.

NO
The definition is not for a single session APPC link to CICS.

YES
The definition is for an APPC terminal on a single session APPC link to CICS.

The MODENAME attribute of the SESSIONS definition can be used to supply a modename for the single session mode set.

An APPC single session terminal can also be defined as a TERMINAL-TYPETERM definition. Both the TERMINAL-TYPETERM definition and the CONNECTION definition can be autoinstalled. If you are considering using autoinstall, see Autoinstalling VTAM terminals.

USEDFLTUSER({NO|YES}) (APPC and MRO only)
specifies the action that is taken when an inbound FMH5 does not contain the security information implied by the ATTACHSEC attribute.

NO
The attach request is rejected, and a protocol violation message is issued.

YES
The attach is accepted, and the *default user ID* is associated with the transaction.

For more information, see Implementing LU6.2 security and Implementing MRO security.

XLNACTION({KEEP|FORCE}) (APPC and MRO only)
XLNACTION specifies the action to be taken when a new logname is received from the partner system. Receipt of a new logname indicates that the partner has deleted its recovery information.

Note: MRO here covers connections with ACCESSMETHOD set to either IRC or XM.

FORCE
The predefined decisions for indoubt UOWs (as defined by the indoubt attributes of the transaction definition) are implemented, before any new work with the new logname is started. CICS also deletes any information retained for possible resolution of UOWs that were indoubt at the partner system.

Attention: Data integrity may be compromised if you use this option.

KEEP
Recovery information is kept, and no action is taken for indoubt units of work.

For IRC, the connection continues with new work. Resolve indoubt UOWs using the CEMT or SPI interface.

For APPC, the connection is unable to perform new work that requires synclevel 2 protocols until all outstanding recoverable work with the partner (that is, indoubt UOWs, or information relevant to UOWs that were indoubt on the partner system under the old logname) is completed using the CEMT or SPI interface.

Installing connection definitions

To install new CONNECTION definitions, put them in a group of their own which does not contain CONNECTION definitions that have already been installed, then use CEDA INSTALL to install the whole group. You cannot install single CONNECTION definitions.

About this task

To modify and reinstall existing MRO CONNECTION definitions, or if you want new and existing MRO CONNECTION definitions to be in the same group, you must close down all interregion communication (IRC) and open it again, before you can use the definition. If you do not close IRC and open it again, you get message DFHIR3788 when you try to bring up the region with the new connection.

Procedure

1. Close IRC down.
   Use the following command:

   ```
   CEMT SET IRC CLOSED
   ```
2. Install the resource definitions.
   Use the following command:

   ```
   CEDA INSTALL GROUP(groupname)
   ```
3. When you have successfully installed the group containing the definitions, open IRC again.
   Use the following command:

   ```
   CEMT SET IRC OPEN
   ```

ISC/MRO connection resource definitions

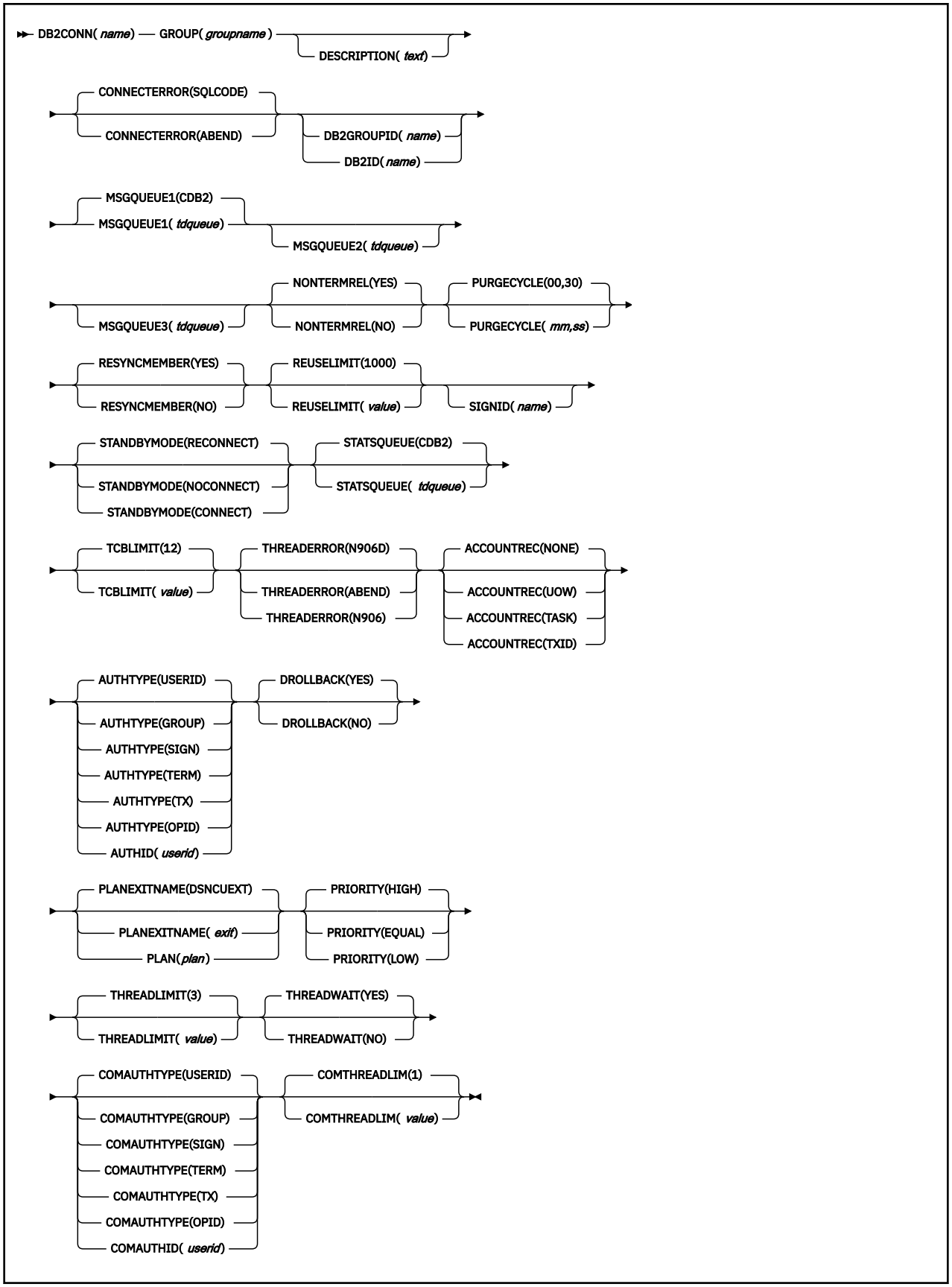
ISC/MRO connection definitions identify remote regions or systems that a CICS region communicates with using intersystem communication (ISC) or multiple region operation (MRO).

DB2CONN resources

A DB2CONN definition defines the attributes of the connection between CICS and DB2®, and of the *pool threads* and *command threads* used with the connection.

DB2CONN attributes

Describes the syntax and attributes of the DB2CONN resource.


DB2CONN(name) GROUP(groupname)
DESCRIPTION(text)
CONNECTERROR(SQLCODE)
CONNECTERROR(ABEND)
DB2GROUPID(name)
DB2ID(name)
MSGQUEUE1(CDB2)
MSGQUEUE1(tdqueue)
MSGQUEUE2(tdqueue)
MSGQUEUE3(tdqueue)
NONTERMREL(YES)
NONTERMREL(NO)
PURGECYCLE(00,30)
PURGECYCLE(mm,ss)
RESYNCMEMBER(YES)
RESYNCMEMBER(NO)
REUSELIMIT(1000)
REUSELIMIT(value)
SIGNID(name)
STANDBYMODE(RECONNECT)
STANDBYMODE(NOCONNECT)
STANDBYMODE(CONNECT)
STATSQUEUE(CDB2)
STATSQUEUE(tdqueue)
TCBLIMIT(12)
TCBLIMIT(value)
THREADERROR(N906D)
THREADERROR(ABEND)
THREADERROR(N906)
ACCOUNTREC(NONE)
ACCOUNTREC(UOW)
ACCOUNTREC(TASK)
ACCOUNTREC(TXID)
AUTHTYPE(USERID)
AUTHTYPE(GROUP)
AUTHTYPE(SIGN)
AUTHTYPE(TERM)
AUTHTYPE(TX)
AUTHTYPE(OPID)
AUTHID(userid)
DROLLBACK(YES)
DROLLBACK(NO)
PLANEXITNAME(DSNCUEXT)
PLANEXITNAME(exit)
PLAN(plan)
PRIORITY(HIGH)
PRIORITY(EQUAL)
PRIORITY(LOW)
THREADLIMIT(3)
THREADLIMIT(value)
THREADWAIT(YES)
THREADWAIT(NO)
COMAUTHTYPE(USERID)
COMAUTHTYPE(GROUP)
COMAUTHTYPE(SIGN)
COMAUTHTYPE(TERM)
COMAUTHTYPE(TX)
COMAUTHTYPE(OPID)
COMAUTHID(userid)
COMTHREADLIM(1)
COMTHREADLIM(value)

General attributes

Describes the general attributes of a DB2CONN resource.

DB2CONN(*name*)
The name to identify a DB2 connection definition. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

Connection attributes

Describes the attributes of the DB2CONN resource that relate to the connection.

CONNECTERROR({SQLCODE|ABEND})
Specifies the way that the information, that CICS is not connected to DB2 because the attachment facility is in 'standby mode', is reported back to an application that has issued an SQL request.

ABEND
The application abends with abend code AEY9.

SQLCODE
The application receives a -923 sqlcode. SQLCODE cannot be specified if STANDBYMODE is set to NOCONNECT.

DB2GROUPID(*name*)
Specifies the group ID (up to four characters) of a data sharing group of DB2 subsystems. The group attach facility connects CICS to any active member of this data sharing group. Match the group ID to the group attachment name defined in DB2. With DB2 Version 10, the group ID can be a subgroup attachment name defined to DB2 which defines a subset of the data sharing group. If the DB2GROUPID attribute remains blank, group attach is not used. You cannot specify both DB2GROUPID and DB2ID, the priorities are as follows:

1. Specifying a DB2GROUPID blanks out any DB2ID that is already set in the DB2CONN definition.
2. If you attempt to specify both a DB2GROUPID and a DB2ID on the same CEDA panel, the DB2ID is used.
3. If the DB2ID of an individual subsystem is specified in a CEMT or EXEC CICS SET DB2CONN command, or in a DSNC STRT command, this DB2ID overrides any DB2GROUPID attribute that is

set in the installed DB2CONN definition. The DB2GROUPID in the installed DB2CONN definition is blanked out, and must be set again (using CEDA or a SET DB2CONN command) to use group attach.

DB2ID(*name*)
Specifies the name of the DB2 subsystem to which the CICS DB2 attachment facility is to connect. By default this field is blank. If you want to use group attach, specify a DB2GROUPID in the DB2CONN definition, instead of a DB2ID. The DB2ID set in the installed DB2CONN definition can be overridden by a DB2 subsystem ID specified on a DSNC STRT command, or by a DB2ID specified in a SET DB2CONN command. If the DB2ID in the installed DB2CONN definition remains blank, and the DB2GROUPID also remains blank, you can specify a DB2 subsystem ID on the INITPARM system initialization parameter. If no DB2 subsystem ID is specified by any of these means, and no DB2GROUPID is specified, the default DB2ID of blanks is replaced by DSN when the connection is attempted. Hence, the hierarchy for determining the DB2 subsystem is as follows:

1. Use the subsystem ID if it is specified in a DSNC STRT command.
2. Use the DB2ID in the installed DB2CONN if it is not blank.
3. Use the DB2GROUPID in the installed DB2CONN for group attach, if it is not blank.
4. Use the subsystem ID if it is specified on the INITPARM when the DB2ID and DB2GROUPID in the last installed DB2CONN are blank (or have later been set to blanks). On any startup, INITPARM is always used if the last installed DB2CONN contained a blank DB2ID and a blank DB2GROUPID, even if the DB2ID or DB2GROUPID was later changed using a SET command.
5. Use a default subsystem ID of DSN.

You cannot specify both DB2GROUPID and DB2ID — if you attempt to specify both on the same CEDA panel, the DB2ID is used. If a DB2GROUPID is specified in a CEMT or EXEC CICS SET DB2CONN command, this overrides any DB2ID that is set in the installed DB2CONN definition, and the DB2ID is blanked out.

MSGQUEUE1({CDB2|*tdqueue*})
Specifies the first transient data destination to which unsolicited messages from the CICS DB2 attachment facility are sent. This first destination cannot be blank.

MSGQUEUE2(*tdqueue*)
Specifies a second transient data destination to which unsolicited messages from the CICS DB2 attachment facility are sent.

MSGQUEUE3(*tdqueue*)
Specifies a third transient data destination to which unsolicited messages from the CICS DB2 attachment facility are sent.

NONTERMREL({YES|NO})
Specifies whether a non-terminal transaction releases threads for reuse at intermediate sync points.

NO
Non-terminal transactions do not release threads for reuse at intermediate sync points.

YES
Non-terminal transactions release threads for reuse at intermediate sync points.

PURGECYCLE({00|*mm*},{30|*ss*})
Specifies the duration, in minutes and seconds, of the purge cycle for protected threads. The duration of the purge cycle is in the range 5 seconds to 59 minutes 59 seconds. If you do not specify a value for PURGECYLE, it defaults to 30 seconds; PURGECYCLE= 00, 30.

A protected thread is not terminated immediately when it is released. It is terminated only after two completed purge cycles, if it has not been reused in the meantime. Therefore, if the purge cycle is set to 30 seconds, a protected thread is purged 30 - 60 seconds after it is released. The first purge cycle after the attachment facility starts is always 5 minutes. After that the purge cycle values are applied. An unprotected thread is terminated when it is released (at sync point or end of task) if there are no other transactions waiting for a thread on that DB2ENTRY. Only threads belonging to a DB2ENTRY can be protected. Pool threads and command threads cannot be protected.

RESYNCMEMBER({YES|NO})

If you are using group attach, use the RESYNCMEMBER attribute to select the strategy that CICS adopts if outstanding units of work are being held for the last DB2 data sharing group member to which CICS was connected.

YES

Indicates that if outstanding units of work are held, you require resynchronization with the last DB2 data sharing group member to which CICS was connected. CICS ignores the group attach facility, and the CICS-DB2 attachment facility waits until it can reconnect to that last connected DB2 data sharing group member, to resolve the indoubt units of work. Units of work which are shunted indoubt are not included in this process, because CICS itself is unable to resolve those units of work at this time. Resynchronization for those UOWs occurs when CICS has resynchronized with its remote coordinator.

NO

Indicates that you do not require resynchronization. CICS makes one attempt to reconnect to the last connected DB2 data sharing group member. If this attempt is successful, the indoubt units of work (except for UOWs that are shunted indoubt) can be resolved. If it is unsuccessful, then CICS uses group attach to connect to any active member of the DB2 data sharing group, and a warning message (DFHDB2064) is issued stating that there can be unresolved indoubt units of work with the last member of the group to which CICS was connected.

REUSELIMIT(*value*)

Specifies a value in the range 0 - 10000 representing the maximum number of times a thread can be reused before it is terminated. The default is 1000. A value of 0 means that there is no limit on the number of times that a thread can be reused; this was the situation before CICS TS 4.2. However, long-running CICS DB2 threads that are constantly being reused build up resources in DB2 that can cause storage problems leading to abends and DB2 subsystem outages.

The reuse limit applies to unprotected threads both in the pool and on a DB2ENTRY, and to protected DB2ENTRY threads. An unprotected thread is reused if, at the time it is released from a transaction, there is a new transaction waiting. A protected thread is reused if a new transaction requires a thread during the time the thread is protected from being terminated. In either case, when the reuse limit is reached no further transactions can use the thread. When the transaction that is currently using the thread releases it, CICS terminates and re-creates the thread to free up DB2 resources before determining if there is new work for the thread to do, or whether the thread is to be protected.

Using the default of 1000 provides sufficient protection against over-allocating thread storage and EDM pool storage below the 2 GB bar when you are using a DB2 bind option of RELEASE(DEALLOCATE) without adversely affecting performance. If, however, DB2 monitoring and statistics show excessive DB2 thread storage, excessive EDM pool storage usage, or both, this limit can be lowered. Conversely, if CICS-DB2 statistics show that pool or entry threads are hitting the reuse limit frequently and there is sufficient virtual and real storage available to allow more DB2 thread storage, this limit can be raised.

Setting a low value for the reuse limit has a performance impact in terms of an increase in processor activity and a decrease in throughput. However, there are situations where you might choose to set a low value. For example, if you wanted to evaluate changing DB2 bind options from RELEASE(COMMIT) to RELEASE(DEALLOCATE) for a plan or package, you might use a low value temporarily to test the scenario.

SIGNID(*name*)

Specifies the authorization ID to be used by the CICS DB2 attachment facility when signing on to DB2 for pool and DB2ENTRY threads that specify AUTHTYPE(SIGN). The default is blanks which are replaced by the applid of the CICS system when the DB2CONN is installed. The ID that you specify can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

If you specify a user ID on the SIGNID attribute, CICS performs a surrogate user check against the user ID that is performing the installation. Similarly, the CICS region user ID is subject to a surrogate user check during group list installation on a CICS cold or initial start.

If the ID you specify matches the CICS region user ID, and you specify AUTHTYPE(SIGN) for any command, pool, or entry threads, the RACF access control environment element (ACEE) for the CICS region user ID is passed to DB2.

STANDBYMODE({RECONNECT|CONNECT|NOCONNECT})
Specifies the action to be taken by the CICS DB2 attachment facility if DB2 is not active when an attempt is made to connect CICS to DB2.

CONNECT
Specifies that the CICS DB2 attachment facility is to wait in 'standbymode' for DB2 to become active. If the connection is made, and DB2 later fails, the CICS DB2 attachment facility terminates.

NOCONNECT
Specifies that the CICS DB2 attachment facility is to terminate.

RECONNECT
Specifies that the CICS DB2 attachment facility is to go into 'standby mode' and wait for DB2. If DB2 later fails after the connection is made, the CICS DB2 attachment facility reverts to 'standby mode', and CICS then reconnects to DB2 when DB2 recovers.

STATSQUEUE({CDB2|*tdqueue*})
Specifies the transient data destination for CICS DB2 attachment facility statistics produced when the CICS DB2 attachment facility is shut down.

TCBLIMIT({12|*value*})

Specifies the maximum number of TCBs that can be used to process DB2 requests. The default is 12. The minimum number is 4 and the maximum is 2000. CICS uses L8 and L9 mode open TCBs to process DB2 requests. The TCBLIMIT attribute of the DB2CONN definition governs how many of the open TCBs can be used to access DB2, that is, how many of them can identify to DB2 and create a connection into DB2.

For more information about TCB limits and open TCB modes, see Open TCB management.

The TCBLIMIT value controls the total number of threads for the CICS region. For this reason, the recommended value for TCBLIMIT is the sum of all the thread limit values (that is, the sum of all THREADLIMIT attributes on the DB2 connection and DB2 entry resource definitions, plus the COMTHREADLIMIT value on the DB2 connection definition) up to the limit of 2000.

If the limit set automatically by CICS for the number of L8 and L9 mode open TCBs is reached, so no more open TCBs can be created, the task is suspended with HTYPE(DISPATCH) and HVALUE(OPEN_TCB). CICS sets this limit using the formula `(2 * MXT value) + 32`, with the MXT or MAXTASKS limit for the CICS region. If this limit is not exceeded but TCBLIMIT is exceeded, then the task is suspended with HTYPE(CDB2CONN). In this situation, although CICS has an open TCB available, the maximum allowed number of open TCBs are being used to access DB2 (as defined in TCBLIMIT).

When determining the number for TCBLIMIT, you must consider the amount you specified for the MAX USERS parameter on DB2 installation panel DSNTIPE.

THREADERROR({N906D|N906|ABEND})
Specifies the processing that is to occur following a create thread error.

ABEND
When the first SQL error is detected, CICS takes a transaction dump for abend code AD2S, AD2T, or AD2U, depending on the type of error. For the first error, the transaction does not abend. For a second or subsequent SQL error, the transaction abends with abend code AD2S, AD2T, or AD2U.

The transaction must be terminated and reinitialized before it is allowed to issue another SQL request.

N906D
A transaction dump is to be taken and the DSNCSQL RMI associated with the transaction is *not* to be disabled. The transaction receives a -906 SQLCODE if another SQL is issued, unless the transaction issues SYNCPOINT ROLLBACK. SYNCPOINT without the ROLLBACK option results in an ASP3 or ASP7 abend. The transaction dump records an abend of AD2S, AD2T, or AD2U.

N906
The DSNCSQL RMI associated with the transaction is *not* to be disabled. The transaction receives a -906 SQLCODE if another SQL request is issued, unless the transaction issues a SYNCPOINT ROLLBACK. SYNCPOINT without the ROLLBACK option results in an ASP3 or ASP7 abend.

Pool thread attributes

Describes the attributes of the DB2CONN resource that relate to the pool thread.

ACCOUNTREC({NONE|TASK|TXID|UOW})
Specifies the minimum amount of DB2 accounting required for transactions that use pool threads. The specified minimum might be exceeded, as described in the following options.

NONE
No accounting records are required for transactions that use pool threads.

DB2 produces at least one accounting record for each thread when the thread is ended. Authorization changes additionally cause accounting records to be produced.

TASK
The CICS DB2 attachment facility causes a minimum of one accounting record for each CICS task to be produced.

A transaction that contains multiple units of work (UOWs), and in which the threads are released at sync point, can use a different thread for each of its UOWs. As a result, an accounting record might be produced for each UOW.

TXID
The CICS DB2 attachment facility causes an accounting record to be produced when the transid that uses the thread changes.

Because pool threads are typically used by a number of different transaction IDs, there is an increased chance that a transaction containing multiple UOWs, and in which the threads are released at sync point, will use a different thread for each UOW. In this case, an accounting record might be produced for each UOW.

UOW
The CICS DB2 attachment facility causes an accounting record to be produced for each UOW, assuming that the thread is released at the end of the UOW.

AUTHID(*userid*)
Specifies the user ID that is used for security checking when using pool threads.

Do not specify AUTHID if you are using RACF for some or all of the security checking in your DB2 address space; use AUTHTYPE instead, with the GROUP, SIGN, or USERID options. You must use AUTHTYPE because threads that use an AUTHID do not pass the required RACF access control environment element (ACEE) to DB2.

The ID that you specify can be up to 8 characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

AUTHTYPE({USERID|OPID|GROUP|SIGN|TERM|TX})

Specifies the type of ID that can be used for threads on this DB2CONN.

If you are using RACF for some or all of the security checking in your DB2 address space, use the GROUP, SIGN, or USERID options. You must use one of these options because only threads defined with these options pass the required RACF access control environment element (ACEE) to DB2. However, if you specify the SIGN option, the ACEE is passed to DB2 only if the value specified for the SIGNID attribute on the DB2CONN definition matches the CICS region user ID.

The ACEE is not required if you are using DB2 internal security only; so, in this case, you can use any of the options.

USERID

The user ID associated with the CICS transaction is used as the authorization ID. If the user ID is less than 8 characters in length, it is padded on the right with blanks.

Important: Do not specify AUTHTYPE(USERID) when you use the DB2 sample sign-on exit DSN@SGN, because it might result in an SQL -922 failure. Specify COMMAUTHTYPE(GROUP) instead.

OPID

The operator identification that is associated with the user ID that is associated with the CICS transaction is used as the authorization ID. The 3-character operator identification is padded on the right with blanks to form the 8-character authorization ID.

GROUP

Specifies the user ID and the connected group name as the authorization ID. The following table shows how these two values are interpreted by DB2.

IDs passed to DB2	How DB2 interprets values
CICS sign-on user ID (USERID)	Represents the primary DB2 authorization ID.
RACF connected group name	If the RACF list of group options is not active, DB2 uses the connected group name supplied by the CICS attachment facility as the secondary DB2 authorization ID. If the RACF list of group options is active, DB2 ignores the connected group name supplied by the CICS attachment facility, but the value appears in the DB2 list of secondary DB2 authorization IDs.

To use the GROUP option, you must specify SEC=YES in the system initialization parameters for the region.

If no RACF group ID is available for this USERID, an 8-character field of blanks is passed to DB2 as the group ID.

SIGN

Specifies that the SIGNID attribute of the DB2 connection definition is to be used as the resource authorization ID.

TERM

Specifies the terminal identification as an authorization ID. The 4-character terminal identification is padded on the right with blanks to form the 8-character authorization ID.

If the transaction is not associated with a terminal (for example, if it is initiated with a START command), do not specify AUTHTYPE(TERM).

TX

Specifies the transaction identification as the authorization ID. The 4-character transaction identification is padded on the right with blanks to form the 8-character authorization ID.

DROLLBACK({YES|NO})
Specifies whether the CICS DB2 attachment facility initiates a SYNCPOINT ROLLBACK if a transaction that is involved in a deadlock resolution is selected.

YES
The attachment facility initiates a SYNCPOINT ROLLBACK before returning control to the application. Additionally the attachment facility changes the SQL return code returned by DB2 from -913 to -911, and returns -911 to the application.

Do not specify YES if the pool is used by transactions running enterprise beans as part of an OTS transaction; SYNCPOINT ROLLBACK is not allowed in an OTS transaction. Consider defining a DB2ENTRY that specifies DROLLBACK(NO) for use by transactions which run enterprise beans as part of an OTS transaction.

NO
The attachment facility does not initiate a rollback for a transaction. An SQL return code of -913 is returned to the application.

PLAN(*plan*)
Specifies the name of the plan to be used for all pool threads.

PLANEXITNAME({DSNCUEXT|*exit*})
Specifies the name of the dynamic plan exit to be used for pool threads. If you change PLAN and PLANEXITNAME while there are active transactions for the pool, the next time the transaction releases the thread the new values are applied.

PRIORITY({HIGH|EQUAL|LOW})
Specifies the priority of the pool thread TCBs relative to the CICS main TCB (QR TCB). The thread TCBs are CICS open L8 TCBs.

HIGH
Thread TCBs have a higher priority than the CICS QR TCB.

EQUAL
Thread TCBs have equal priority with the CICS QR TCB.

LOW
Thread TCBs have a lower priority than the CICS QR TCB.

THREADLIMIT({3|*value*})
Specifies the current maximum number of pool threads that the CICS DB2 attachment facility allows to be active before requests are made to wait or are rejected, subject to the THREADWAIT attribute. The default value of THREADLIMIT (3) is also the minimum that you can specify. The maximum value must not be greater than the value specified for TCBLIMIT.

THREADWAIT({YES|NO})
Specifies whether transactions wait for a pool thread or end abnormally if the number of active pool threads reaches the thread limit.

The CICS DB2 attachment issues a unique abend code AD3T, message DFHDB2011, when you code THREADWAIT=NO and the number of pool threads is exceeded.

YES
If all threads are busy, a transaction must wait until one becomes available. A transaction can wait as long as CICS allows it to wait, generally until a thread becomes available.

NO
If all threads are busy, the transaction ends abnormally with code AD2T or AD3T.

Command thread attributes

Describes the attributes of the DB2CONN resource that relate to the command thread. A command thread is used when the DSNC transaction is used to send a command to Db2® and when CICS issues a command to cancel a thread that is active in Db2 as part of the processing of a CICS task purge or forcepurge request.

The DB2 connection definition command thread attribute descriptions are:

COMAUTHID(*userid*)
Specifies what id the CICS DB2 attachment facility should use for security checking when using command threads.

Do not use COMAUTHID if you are using RACF for some or all of the security checking in your DB2 address space; use COMAUTHTYPE instead, with the USERID or GROUP option. You can also use COMMAUTHTYPE with the SIGN option when the SIGNID attribute on the DB2CONN definition matches the CICS region user ID. This is because threads using a COMAUTHID do not pass the required RACF access control environment element (ACEE) to DB2. The ACEE is not required if you are only using DB2 internal security, so in this case, you can use COMAUTHID.

The ID that you specify can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

COMAUTHTYPE({USERID|OPID|GROUP|SIGN|TERM|TX})
Specifies the type of id that can be used for security checking when using command threads.

If you are using RACF for some or all of the security checking in your DB2 address space, use the GROUP, SIGN or USERID options. This is because only threads defined with these options pass the required RACF access control environment element (ACEE) to DB2. However, if you specify the SIGN option, the ACEE is passed to DB2 only if the value specified for the SIGNID attribute on the DB2CONN definition matches the CICS region user ID.

USERID
The 1 to 8-character userid associated with the CICS transaction is used as the authorization ID. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

Important: Do not specify COMMAUTHTYPE(USERID) when you use the DB2 sample sign-on exit DSN@SGN, as this may result in an SQL -922 failure. Specify COMMAUTHTYPE(GROUP) instead.

OPID
The operator identification associated with the userid that is associated with the CICS transaction sign-on facility is used as the authorization ID (three characters padded to eight).

GROUP
Specifies the 1 to 8-character USERID and the connected group name as the authorization ID. The following table shows how these two values are interpreted by DB2.

IDs passed to DB2	How DB2 interprets values
CICS sign-on user ID (USERID)	Represents the primary DB2 authorization ID.

IDs passed to DB2	How DB2 interprets values
RACF connected group name	If the RACF list of group options is not active, DB2 uses the connected group name supplied by the CICS attachment facility as the secondary DB2 authorization ID. If the RACF list of group options is active, DB2 ignores the connected group name supplied by the CICS attachment facility, but the value appears in the DB2 list of secondary DB2 authorization IDs.

To use the CGROUP option the CICS system must have SEC=YES specified in the CICS system initialization table (SIT).

If no RACF group ID is available for this USERID, an 8-character field of blanks is passed to DB2 as the group ID.

SIGN
Specifies that the SIGNID attribute of the DB2CONN is used as the resource authorization ID.

TERM
Specifies the terminal identification (four characters padded to eight) as an authorization ID. An authorization ID cannot be obtained in this manner if a terminal is not connected with the transaction.

If a transaction is started (using a CICS command) and has no terminal associated with it, the COMAUTHTYPE(TERM) should not be used.

TX
Specifies the transaction identification (four characters padded to eight) as the authorization ID.

COMTHREADLIM({1|value})
The number specifies the current maximum number of command threads the CICS DB2 attachment facility allows active before requests overflow to the pool.

Installing DB2 connection definitions

This section describes the guidelines for installing and discarding DB2CONN definitions and the implications of interruptions in partial activity.

About this task

- Only one DB2CONN can be installed in a CICS system at any one time. An install of a second DB2CONN can implicitly DISCARD the existing DB2CONN and its associated DB2ENTRYs and DB2TRANs (unless reinstalling a DB2CONN of the same name) before proceeding with the installation.
- A DB2CONN must be installed before any DB2ENTRY or DB2TRAN definitions. DB2ENTRY and DB2TRAN definitions cannot exist on their own, and can only be associated with a DB2CONN that is already installed. Also, if you discard a DB2CONN, the associated DB2ENTRY and DB2TRAN resource definitions are also discarded. Note that there is no attribute on a DB2ENTRY or DB2TRAN that explicitly specifies the DB2CONN to which they belong. This allows DB2ENTRY and DB2TRAN definitions to be shared by DB2CONN definitions without alteration.

  Note: When DB2CONN, DB2ENTRYs, and DB2TRANs are defined in the same group, CICS automatically installs the DB2CONN first. If you install DB2 definitions from multiple groups (by means of a list or multiple INSTALL GROUP commands), the first group you install must contain the DB2CONN definition. Successive groups should not have any DB2CONN definitions. CICS issues an error message when installing a DB2ENTRY or DB2TRAN when no DB2CONN is installed. If multiple DB2CONN definitions are installed, all DB2 definitions installed before the final DB2CONN definition are discarded. CICS issues messages for all discards.

- A DB2CONN must be installed before the CICS DB2 connection can be started. Because it contains information regarding pool threads and command threads, as well as global type information, a DB2CONN represents the minimum required to start the CICS DB2 connection. There are no entry threads, and all transactions use the pool. You can add DB2ENTRYs and DB2TRANs after the CICS DB2 connection is active.
- A DB2CONN can be re-installed only if the CICS DB2 attachment facility is not connected to DB2, and therefore inactive.
- A DB2CONN can be discarded only when the CICS DB2 attachment facility is not connected to DB2.
- The discard of a DB2CONN implicitly discards all installed DB2ENTRYs and DB2TRANs.

  Note: There is no group commit or group discard of DB2CONNs, DB2ENTRYs, and DB2TRANs. However when a DB2CONN is discarded, the underlying control block is marked stating that a discard is in progress. The DB2ENTRYs and DB2TRANs are discarded before the DB2CONN. If the discard fails when half completed, a DB2CONN control block results, and a message is issued that a discard is in progress. A start of the CICS DB2 attachment facility fails with a message:

```
DFHDB2074 CICS DB2 ATTACHMENT FACILITY STARTUP
          CANNOT PROCEED AS THE CURRENTLY
          INSTALLED DB2CONN IS NOT USABLE
```

  when using a partially discarded DB2 resource definition. You must re-issue the discard. When a CICS system restarts after a failure when discarding, it knows that a discard took place. CICS does not recover the blocks from the catalog, and this effectively completes the discard. (Note that the definitions are removed from the catalog as well.)

  When you are installing, parts of any group or list install can fail, but messages are displayed that identify which resources have failed. You can proceed with a start of the CICS DB2 attachment facility when this happens.

Checks on definitions of DB2 connection resources

When you define a DB2CONN, CICS checks for consistency with other resource definitions in the same group or list.

For a DB2CONN object, the following checks are made:

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

DB2 connection resource definitions

A DB2 connection definition establishes the global characteristics of connections between CICS regions and a DB2 subsystem that use the DB2 attachment facility.

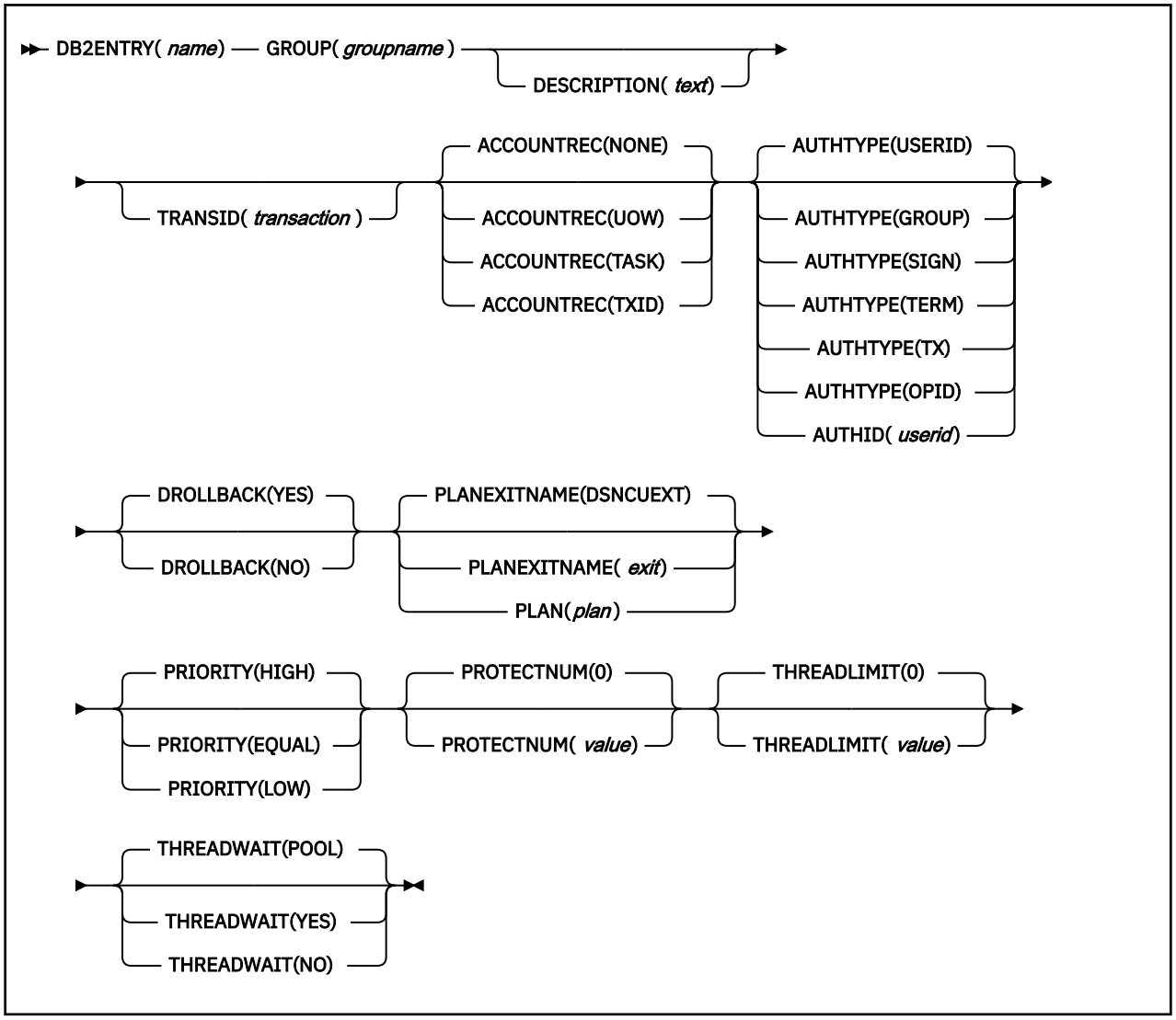
DB2ENTRY resources

A DB2ENTRY defines the attributes of *entry threads* used by the CICS DB2 attachment facility.

A transaction, or a group of transactions, can be associated with the DB2ENTRY; a group of transactions may be represented by the use of one or more wildcard characters (see “Wildcard characters for transaction IDs” on page 38). Also, further transactions can be associated with a DB2ENTRY by defining a DB2TRAN.

DB2ENTRY attributes

Describes the syntax and attributes of the DB2ENTRY resource.



General attributes

Describes the general attributes of a DB2ENTRY resource.

DB2ENTRY(*name*)
One to eight character name to identify a DB2 entry definition.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

Thread selection attributes

Describes the attributes of the DB2ENTRY resource that relate to thread selection.

TRANSID(*transaction*)
Specifies the transaction id associated with the entry. Only one transaction can be specified here. However, the use of one or more wildcard characters in the TRANSID (see "Wildcard characters for transaction IDs" on page 38) allows a group of transactions to be represented. Additional transactions can be defined for this DB2ENTRY by defining a DB2TRAN that refers to this DB2ENTRY. Transid is optional on a DB2ENTRY. All transactions can be associated with a DB2ENTRY by means of DB2TRAN definition instead. However, if only one transaction is associated with a DB2ENTRY, it is easier to specify it on the DB2ENTRY.

Note: Specifying a transaction id here causes an implicit DB2TRAN object to be created when the DB2ENTRY definition is installed. The implicit DBTRAN will have a name DFH concatenated with the transaction id specified. These implicit DB2TRANs will appear on CICS and CPSM views of DB2TRAN resources. If a DB2ENTRY is altered to specify a different transid, when the DB2ENTRY is reinstalled the previous implicit DB2TRAN will be deleted and a new implicit DB2TRAN created representing the new transid associated with the DB2ENTRY.

Thread operation attributes

Describes the attributes of the DB2ENTRY resource that relate to thread operations.

ACCOUNTREC({NONE|TASK|TXID|UOW})
Specifies the minimum amount of DB2 accounting required for transactions using this DB2 entry. The specified minimum might be exceeded, as described in the following options.

NONE
No accounting records are required for transactions using threads from this DB2ENTRY

However, DB2 produces at least one accounting record for each thread after the thread is terminated. Authorization changes additionally cause records to be produced.

TASK
The CICS DB2 attachment facility causes a minimum of one accounting record for each CICS task to be produced.

A transaction that contains multiple units of work (UOWs), and in which the threads are released at sync point, can use a different thread for each of its UOWs. As a result, an accounting record might be produced for each UOW.

TXID
The CICS DB2 attachment facility causes an accounting record to be produced when the transid using the thread changes.

This option applies to DB2 entry definitions that are used by more than one transaction ID. As threads are typically released at syncpoint, a transaction containing multiple UOWs might use a different thread for each UOW. As a result, an accounting record might be produced for each UOW.

UOW
The CICS DB2 attachment facility causes an accounting to be produced for each UOW, assuming that the thread is released at the end of the UOW.

AUTHID(*userid*)
Specifies the user ID that is used for security checking when using this DB2ENTRY.

Do not specify AUTHID if you are using RACF for some or all of the security checking in your DB2 address space; use AUTHTYPE instead, with the GROUP, SIGN, or USERID options. You must use AUTHTYPE because threads using an AUTHID do not pass the required RACF access control environment element (ACEE) to DB2.

The ID that you specify can be up to 8 characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

AUTHTYPE({USERID|OPID|GROUP|SIGN|TERM|TX})
Specifies the type of ID that can be used for security checking when using this DB2ENTRY.

If you are using RACF for some or all of the security checking in your DB2 address space, use the GROUP, SIGN, or USERID options. You must use one of these options because only threads defined with these options pass the required RACF access control environment element (ACEE) to DB2. However, if you specify the SIGN option, the ACEE is passed to DB2 only if the value specified for the SIGNID attribute on the DB2CONN definition matches the CICS region user ID.

The ACEE is not required if you are using DB2 internal security only; so in this case, you can use any of the options.

USERID
The USERID associated with the CICS transaction is used as the authorization ID. If the user ID is less than 8 characters in length, it is padded on the right with blanks.

When the DB2 sample sign-on exit DSN3@SGN is used with AUTHTYPE(USERID), the exit sends the user ID to DB2 as the primary authorization ID and the connected group name to DB2 as the secondary ID. When the sample sign-on exit is used, AUTHTYPE(USERID) and AUTHTYPE(GROUP) are the same.

OPID
The operator identification that is associated with the userid that is associated with the CICS transaction sign-on facility is used as the authorization ID. The 3-character operator identification is padded on the right with blanks to form the 8-character authorization ID.

GROUP
Specifies the 1- to 8-character USERID and the connected group name as the authorization ID. The following table shows how these two values are interpreted by DB2.

IDs passed to DB2	How DB2 interprets values
CICS sign-on user ID (USERID)	Represents the primary DB2 authorization ID.

IDs passed to DB2	How DB2 interprets values
RACF connected group name	If the RACF list of group options is not active, DB2 uses the connected group name supplied by the CICS attachment facility as the secondary DB2 authorization ID. If the RACF list of group options is active, DB2 ignores the connected group name supplied by the CICS attachment facility, but the value appears in the DB2 list of secondary DB2 authorization IDs.

To use the GROUP option, you must specify SEC=YES in the system initialization parameters for the region.

If no RACF group ID is available for this USERID, an 8-character field of blanks is passed to DB2 as the group ID.

SIGN
Specifies that the SIGNID attribute of the DB2CONN is used as the resource authorization ID.

TERM
Specifies the terminal identification (4 characters padded to 8) as an authorization ID. An authorization ID cannot be obtained in this manner if a terminal is not connected with the transaction.

If a transaction is started, using a CICS command, and has no terminal associated with it do not use AUTHTYPE(TERM).

TX
Specifies the transaction identification as the authorization ID. The 4-character transaction identification is padded on the right with blanks to form the 8-character authorization ID.

DROLLBACK({YES|NO})
Specifies whether the CICS DB2 attachment facility initiates a SYNCPOINT ROLLBACK in the event of a transaction being selected as victim of a deadlock resolution.

YES
The attachment facility initiates a SYNCPOINT ROLLBACK before returning control to the application. Additionally the attachment facility changes the SQL return code returned by DB2 from -913 to -911, and returns -911 to the application.

Do not specify YES if the DB2ENTRY is used by transactions running enterprise beans as part of an OTS transaction; SYNCPOINT ROLLBACK is not allowed in an OTS transaction.

Similarly, do not specify YES if the DB2ENTRY is used by transactions running as a DPL server without SYNCONRETURN; SYNCPOINT ROLLBACK is invalid in this scenario and would result in an AD2Z abend.

NO
The attachment facility does not to initiate rollback for this transaction. An SQL return code of -913 is returned to the application.

PLAN(*plan*)
Specifies the name of the plan to be used for this entry.

PLANEXITNAME(DSNCUEXT|*exit*)
Specifies the name of the dynamic plan exit to be used for this DB2 entry definition. If you change PLAN and PLANEXITNAME while there are active transactions for the DB2 entry definition, the next time the transaction releases the thread, the new values are applied.

PRIORITY({HIGH|EQUAL|LOW})
Specifies the priority of the thread TCBs for this DB2ENTRY relative to the CICS main TCB (QR TCB).If CICS is connected to DB2 Version 6 or later, the thread TCBs are CICS open L8 TCBs.

HIGH
Thread TCBs have a higher priority than the CICS QR TCB.

EQUAL
Thread TCBs have equal priority with the CICS QR TCB.

LOW
Thread TCBs have a lesser priority than the CICS QR TCB.

PROTECTNUM({0|*value*})
Specifies the maximum number of protected threads allowed for this DB2 entry definition. A thread that is released by a transaction when no other work is queued can be *protected*, meaning that it is does not end immediately. A protected thread ends after two complete purge cycles if it has not been reused in the meantime. For example, if the purge cycle is set to 30 seconds, a protected thread ends 30 - 60 seconds after it is released if it is not reused in the meantime. The first purge cycle after the CICS DB2 attachment facility has been started is 5 minutes, after which the PURGECYCLE value is applied. Threads are protected only while they are inactive. If a transaction reuses a protected thread, the thread becomes active, and the current number of protected threads is decremented.

THREADLIMIT({0|*value*})
Specifies the maximum number of threads for this DB2 entry definition that the CICS DB2 attachment allows active before requests are made to wait, are abended, or diverted to the pool.

THREADWAIT({POOL|YES|NO})
Specifies whether transactions wait for a DB2ENTRY thread, end abnormally, or overflow to the pool if the number of active DB2ENTRY threads reaches the THREADLIMIT number.

POOL
If all threads are busy, the transaction is diverted to use the pool of threads. If the pool is also busy, and you specified THREADWAIT(NO) on the DB2 connection definition, the transaction ends abnormally with abend code AD3T.

NO
If all threads are busy, the transaction ends abnormally with abend AD2P.

YES
If all threads are busy, the transaction waits until a thread becomes available.

Installing DB2 entry definitions

This section describes the guidelines for installing and discarding DB2ENTRY definitions and the implications of interruptions in partial activity.

About this task

- You can install a DB2ENTRY only if you have previously installed a DB2CONN.
- You can install a new DB2ENTRY at any time, even when the CICS DB2 adapter is connected to DB2.
- You can reinstall (by replacing an existing DB2ENTRY) only when the DB2ENTRY is disabled and no transaction is using it. Use the **SET DB2ENTRY DISABLED** command to quiesce activity and disable the entry. New transactions trying to use the DB2ENTRY are routed to the pool, abended, or returned an SQLCODE, dependent on the setting of the DISABLEDACT keyword when the DB2ENTRY is disabled.
- You can discard a DB2ENTRY only if it is disabled. Use the **SET DB2ENTRY DISABLED** command to quiesce activity and disable the entry. New transactions trying to use the DB2ENTRY are routed to the pool, abended, or returned an SQLCODE, dependent on the setting of the DISABLEDACT keyword when the DB2ENTRY is disabled.

  If you discard a DB2ENTRY, you could make the corresponding DB2TRAN an 'orphan'. If you then run a transaction, a message is sent to the CDB2 transient data destination, and the request is rerouted to the pool.

Checks on definitions of DB2 entry resources

When you define a DB2ENTRY, CICS checks for consistency with other resource definitions in the same group or list.

For a DB2ENTRY object, the following checks are made:

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

DB2 entry resource definitions

A DB2 entry definition specifies the resources required by CICS transactions that use the DB2 attachment facility to access a DB2 subsystem.

DB2TRAN resources

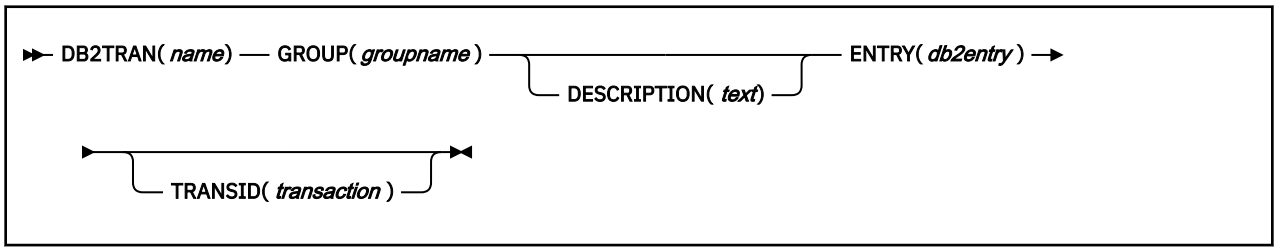
A DB2TRAN defines a transaction, or a group of transactions, associated with a DB2ENTRY, that are additional to the transactions specified in the DB2ENTRY itself.

Only one DB2TRAN definition can be installed for a specific transaction. An attempt to install a second DB2TRAN definition explicitly referring to the same transaction ID will fail.

The DB2TRAN definition allows a DB2ENTRY to have an unrestricted number of transactions associated with it, including names using wildcard characters. You can define any number of DB2TRANs to be associated with a single DB2ENTRY.

DB2TRAN attributes

Describes the syntax and attributes of the DB2TRAN resource.



DB2TRAN(*name*)
The one to eight character name to identify this DB2 transaction definition.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

ENTRY(*db2entry*)
Specifies the name of the DB2 entry definition to which this DB2 transaction definition refers. It is the DB2 entry definition with which this additional transaction should be associated.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

TRANSID(*transaction*)
Specifies the transaction id to be associated with the entry. If the TRANSID is not specified it defaults to the first four characters of the DB2 transaction definition name. The transaction id can include wildcard characters (see “Wildcard characters for transaction IDs” on page 38).

Wildcard characters for transaction IDs

When you define a DB2TRAN resource, you can specify a generic transaction ID using asterisk (*) and plus (+) symbols as wildcard characters.

- An asterisk (*) can be added to a transaction name, or used alone, to produce the effect on any value of making it 'wild'. The transaction name specified with an asterisk at the end of the name represents 0-3 unspecified characters in the transaction ID. For example, a TRansid of "T*" would represent transaction "T", "TA", "TAB", "TABE".
- A plus sign (+) is allowed in any position to represent any single character.
- An asterisk alone represents any transaction and can act as an alternative pool definition. It differs from a pool by having the additional attribute of the DB2ENTRY of overflowing to the pool when the thread allocation is exhausted.

The rules of matching are that the most specific match is taken. For example, transaction FRED will use DB2ENTRY(1) specifying a generic transaction ID of "FRE*", rather than DB2ENTRY(2) specifying a generic transaction ID of "F*". Also a '+' is more specific than a '*', eg "FRE+" is more specific than "FRE*".

If AUTHTYPE(TX) is specified, the actual TXID is passed to DB2 as the primary authorization ID, and not the name that used wildcard characters.

Note that if a DB2TRAN is defined using a generic transaction ID that includes a wildcard, the INQUIRE DB2TRAN command is unable to identify the individual transactions that match the generic transaction ID. For example, you can issue the command

```
CEMT INQUIRE DB2TRAN(*) TRANSID(ABCD)
```

to see details of the DB2TRAN with which transaction ABCD is associated. However, if the DB2TRAN is defined using a transaction ID "ABC*", the INQUIRE DB2TRAN command is unable to match the DB2TRAN to the transaction ID "ABCD", and returns a 'not found' response.

Installing DB2 transaction definitions

This section describes the guidelines for installing and discarding DB2TRAN definitions and the implications of interruptions in partial activity.

About this task

- You cannot install more than one DB2TRAN for the same transaction, if you have given the full transaction ID in the DB2TRAN definition. If you have used a generic transaction ID with a wildcard character, you can install more than one DB2TRAN that potentially matches the transaction, but CICS only uses the closest match (see "Wildcard characters for transaction IDs" on page 38).
- A DB2TRAN that refers to a non-existent DB2ENTRY cannot be installed. The DB2ENTRY must be installed first.
- Note that when DB2ENTRY and DB2TRAN definitions are defined in the same group, CICS installs the DB2ENTRY first at install time.
- You can install a new DB2TRAN at any time, even when the CICS DB2 adapter is connected to DB2.
- A DB2TRAN can be re-installed at any time, and can be discarded at any time.

Checks on definitions of DB2 transaction resources

When you define a DB2TRAN, CICS checks for consistency with other resource definitions in the same group or list.

For a DB2TRAN object, the following checks are made:

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

DB2 transaction resource definitions

A DB2 transaction definition identifies transactions that use the resources specified in a DB2 entry definition.

DOCTEMPLATE resources

A DOCTEMPLATE resource defines the attributes of a *document template*.

A document template is a unit of information that is used to construct a document. A document template can contain fixed text, and symbols that represent text whose value is supplied by an application program. Document templates can be created by a CICS application, or retrieved from an external source. For more information, see CICS documents and document templates.

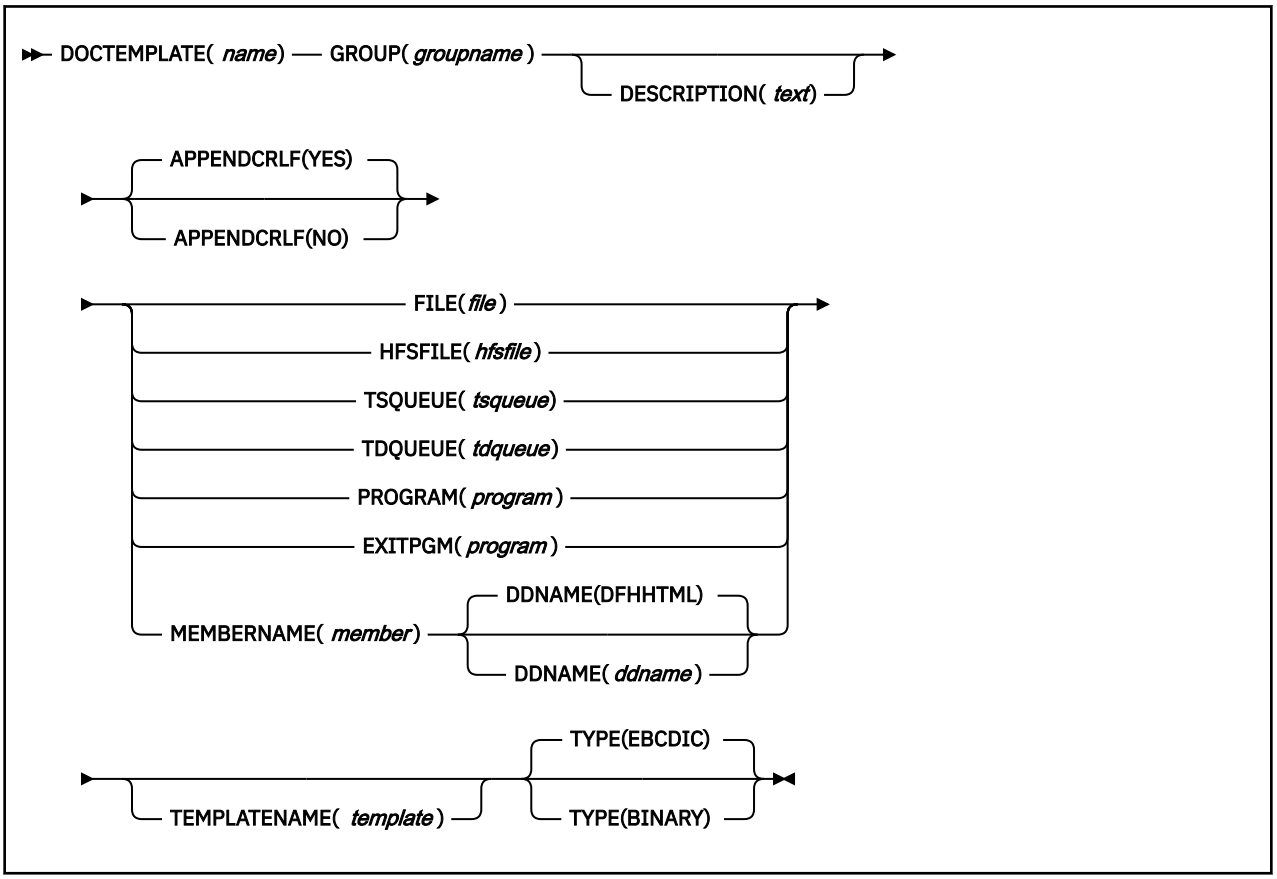
The template can reside in one of the following places:

- An MVS partitioned data set (specified by the DDNAME and MEMBERNAME attributes)
- A temporary storage queue (specified by the TSQUEUE attribute)
- A transient data queue (specified by the TDQUEUE attribute)
- A CICS program (specified by the PROGRAM attribute)
- A CICS file (specified by the FILE attribute)
- A z/OS UNIX System Services file (specified by the HFSFILE attribute)

The template can also be returned by an exit program (specified by the EXITPGM attribute).

DOCTEMPLATE attributes

Describes the syntax and attributes of the DOCTEMPLATE resource.



APPENDCRLF(YES|NO)
specifies whether CICS is to delete trailing blanks from and append carriage-return line-feed to each logical record of the template .

DDNAME(DFHHTML|*ddname*)
when the template resides in an MVS partitioned data set (PDS), specifies the DDname of the PDS. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

If you specify a value for the MEMBERNAME attribute, but do not specify a value for DDNAME, the default value of DFHHTML is taken.

If you specify this attribute, you cannot specify EXITPGM, FILE, HFSFILE, PROGRAM, TDQUEUE or TSQUEUE.

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

DOCTEMPLATE(*name*)
specifies the name of this document template definition. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

EXITPGM(*program*)
specifies the name of an exit program that generates a template. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

If you specify this attribute, you cannot specify DDNAME, FILE, HFSFILE, MEMBERNAME, PROGRAM, TDQUEUE or TSQUEUE.

FILE(*file*)
when the template resides in a CICS file, specifies the name of the file. The name can be eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

If you specify this attribute, you cannot specify DDNAME, EXITPGM, HFSFILE, MEMBERNAME, PROGRAM, TDQUEUE or TSQUEUE.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

HFSFILE(*hfsfile*)
When the template resides in a z/OS UNIX System Services file, this specifies the fully qualified (absolute) or relative name of the z/OS UNIX file. The name can be specified as an absolute name including all directories and beginning with a slash, for example, `/u/facts/images/bluefish.jpg`. Alternatively, it can be specified as a name relative to the HOME directory of the CICS region userid, for example, `facts/images/bluefish.jpg`. Up to 255 characters can be used.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

If you specify this attribute, you cannot specify DDNAME, EXITPGM, MEMBERNAME, PROGRAM, TDQUEUE or TSQUEUE.

Note: The CICS region must have permissions to access z/OS UNIX, and it must have permission to access the z/OS UNIX directory containing the file, and the file itself. For more information about how to grant these permissions, see Giving CICS regions access to z/OS UNIX directories and files.

MEMBERNAME(*member*)
when the template resides in an MVS partitioned data set (PDS), specifies the name of the member containing the template. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

If you specify this attribute, you cannot specify EXITPGM, FILE, HFSFILE, PROGRAM, TDQUEUE or TSQUEUE.

PROGRAM(*program*)
when the template resides in a CICS program, specifies the name of the program. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

If you specify this attribute, you cannot specify DDNAME, EXITPGM, FILE, HFSFILE, MEMBERNAME, TDQUEUE or TSQUEUE.

TDQUEUE(*tdqueue*)
when the template resides in a transient data queue, specifies the name of the queue. The name can be up to four characters in length.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

If you specify this attribute, you cannot specify DDNAME, EXITPGM, FILE, HFSFILE, MEMBERNAME, PROGRAM, or TSQUEUE.

TEMPLATENAME(*template*)
specifies the name by which the template is known to application programs that use it. The name can be up to 48 characters in length.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

If you do not specify a value for this attribute, the value of the DOCTEMPLATE attribute is used, extended on the right with blanks.

TSQUEUE(*tsqueue*)
when the template resides in a temporary storage queue, specifies the name of the queue. The name can be up to 16 characters in length.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

If you specify this attribute, you cannot specify DDNAME, EXITPGM, FILE, HFSFILE, MEMBERNAME, PROGRAM, or TDQUEUE.

TYPE({EBCDIC|BINARY})
specifies the format of the contents of the template.

BINARY
When the template is loaded from the template library, no parsing of the template's contents is done.

EBCDIC
When the template is loaded from the template library, the contents are parsed as EBCDIC text.

Document template resource definitions

Document template definitions define document templates for use in managed CICSregions.

ENQMODEL resources

An ENQMODEL defines a named resource for which the **EXEC CICS ENQ** and **EXEC CICS DEQ** commands have a sysplex-wide scope.

CICS uses ENQMODEL resources with z/OS global resource serialization to provide sysplex-wide protection of resources that are used by multiple applications. Local enqueues within a single CICS region are managed in the CICS address space. Sysplex-wide enqueues that affect more than one CICS region are managed by z/OS global resource serialization. For more information on global resource serialization, see z/OS MVS Planning: Global Resource Serialization.

When the **EXEC CICS ENQ** and **EXEC CICS DEQ** commands are issued for a resource, CICS checks for a matching installed ENQMODEL definition. The ENQSCOPE attribute of an ENQMODEL resource defines the set of regions that share the same enqueue scope. If the ENQSCOPE attribute remains blank (the default value), CICS treats any matching enqueue or dequeue request as local to the issuing CICS region. If the ENQSCOPE is non-blank, CICS treats the enqueue or dequeue as sysplex-wide, and passes a queue name and the resource name to z/OS global resource serialization to manage the enqueue.

z/OS global resource serialization can be configured in either a ring configuration, or a star configuration. For reasons of performance, a ring configuration is not recommended for production regions in a multisystem environment. You can also use the CICS system initialization parameter **NQRNL** to specify whether or not z/OS global resource serialization uses RNL processing for CICS enqueue requests, which can affect the scope of resources. For more information on configuring z/OS global resource serialization for CICS, see Global CICS enqueue and dequeue: improving performance .

You can display sysplex enqueues that are using z/OS global resource serialization facilities with the **DISPLAY GRS** command, for example:

```
D  GRS,RES=(DFHEqname|*,[  rname|,*])
```

where:

qname
specifies a 4-character ENQSCOPE defined by an ENQMODEL resource.

rname
specifies an ENQNAME defined by an ENQMODEL resource.

Sysplex-wide enqueue is supported only for a resource, and not for an enqueue on an address. It is not designed for the locking of recoverable resources.

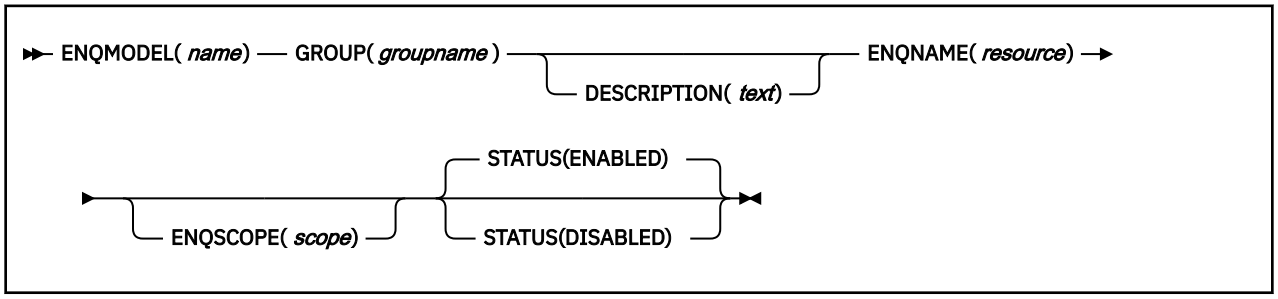
Access to ENQMODEL resources

The CICS regions that need to use sysplex-wide enqueue or dequeue function must all have the required ENQMODEL resources defined and installed. The recommended way to ensure this is for the CICS regions to share a CSD, and for the initialization group lists to include the same ENQMODEL groups.

Applications use sysplex enqueues when you define appropriate ENQMODEL resources; no change is needed to the application programs. For applications where the resource name is configured dynamically, so is not known in advance, you can use the enqueue EXEC interface program exits XNQEREQ and XNQEREQC to supply characters at the start of the resource name that match a suitable ENQMODEL resource definition. For more information on these user exits, see Enqueue EXEC interface program exits XNQEREQ and XNQEREQC.

ENQMODEL attributes

Describes the syntax and attributes of the ENQMODEL resource.



DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

ENQMODEL(*name*)
Specifies the name of this ENQMODEL definition. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

This name is used to identify the ENQMODEL definition on the CSD file. It is not used within the active CICS system.

ENQNAME(*resource*)
Specifies the 1 to 255-character resource name.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < > *
```

You can also use an * (asterisk) as the last character, to denote a generic name. An * (asterisk) anywhere else in the name is treated as a literal * character.

ENQSCOPE(*scope*)
Specifies the optional 4-character enqueue model scope name. If omitted or specified as blanks, matching enqueue models have a local scope.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

STATUS({ENABLED|DISABLED})
Specifies whether the enqueue model is to be installed in ENABLED or DISABLED status. ENABLED is the default.

ENABLED
Matching enqueue requests are processed in the normal way.

DISABLED
Matching enqueue requests are rejected, and the issuing task is abended. Matching INSTALL CREATE and DISCARD requests are processed.

Installing enqueue model definitions

As ENQMODELs usually have the default status of *ENABLED*, the order of installing ENQMODELs must follow the rules for ENABLING ENQMODELs.

About this task

When you enable ENQMODELs that form nested generic ENQnames you must do so starting with the most specific, and continue in order to the least specific.

For example, if you have ENQMODELS that have ENQnames of ABCD*, ABC*, and AB*, enable them in this order:

1. ABCD* (the most specific)
2. ABC*
3. AB* (the least specific)

Enqueue model definitions

Enqueue model definitions describe how enqueue models are to run in a CICS region.

FEPI node list resource definitions

FEPI node list definitions describe the physical and operational characteristics of FEPI nodes.

FEPI pool resource definitions

FEPI pool definitions describe the physical and operational characteristics of FEPI pools.

FEPI property set resource definitions

FEPI property set definitions describe the physical and operational characteristics of FEPI property sets.

FEPI target list resource definitions

FEPI target list definitions describe the physical and operational characteristics of FEPI targets.

FILE resources

A FILE resource defines the physical and operational characteristics of a file.

The FILE definition includes attributes that provide information about record characteristics, the types of operations allowed on the file, recovery attributes, and the operations that are to be journaled. CICS files usually correspond to physical data sets that must have been defined to VSAM before they are used. Using CICS files, your applications can:

- Access records in the data set directly
- Access records in a data table that has been loaded from the data set
- Access records in a coupling facility data table where there is no data set involved (because LOAD(NO) is specified on the CFDT file definition).

The following resources associated with CICS files can be managed using RDO:

- VSAM files (this includes files that refer to CICS-maintained, user-maintained, and coupling facility data tables as well as files that refer to VSAM data sets)
- Remote VSAM files
- Remote BDAM files
- VSAM local shared resource (LSR) pools, which are defined by LSRPOOL resource definitions

For the file to be used by an active CICS system, its definition must have been installed on to the system. CICS file control uses the installed definition to find the file when it needs to access it, to keep count of the number of tasks using the file, to capture processing statistics, and maintain other file control information.

FILE resources in CICS bundles

You can use a CICS bundle to create, edit, and install a FILE resource definition. If you create a FILE resource in this way, you can use the CICS bundle to manage the lifecycle of that resource.

The following file types are supported for definition in CICS bundles:

- VSAM files (including files that refer to CICS-maintained, user-maintained, and coupling facility data tables, as well as files that refer to VSAM data sets)
- Remote VSAM files
- Remote BDAM files

The initial status of a dynamically generated FILE resource is derived from the initial status of the bundle that defines the resource. As a result, it is not possible to define a FILE resource with a STATUS of UNENABLED to inhibit the implicit opening of files by applications.

You can inquire on a FILE resource that is dynamically generated by a CICS bundle, by using the **EXEC CICS INQUIRE FILE** or **CEMT INQUIRE FILE** command.

You cannot issue the **DISCARD** command against a FILE resource that is dynamically generated by a CICS bundle. You must discard the CICS bundle, and CICS applies the operation to the JVMSERVER resource.

To modify the attributes of a FILE resource that was defined and installed in a CICS bundle, use the resource editor in the CICS Explorer to modify the definition in the CICS bundle, and install a new version of the CICS bundle or of the application with which it was deployed. You can use the **SET FILE** command to change the attributes of the dynamically generated resource, but these changes are not cataloged and are not recovered across a warm restart of CICS.

To change the status of a FILE resource that was defined and installed in a CICS bundle, change the status of the CICS bundle or the application with which it is deployed. If you are experiencing a problem with disabling a CICS bundle that defines a FILE resource, you may use the **EXEC CICS SET FILE** or **CEMT SET FILE** commands on the dynamically generated resource, if this action is required. Follow the troubleshooting procedure in Diagnosing application errors to diagnose the problem and take suitable action.

For more information about defining resources in CICS bundles, see Defining CICS bundles.

Remote files

When multiple CICS systems are connected, they can share each other's files; all such files must be defined to CICS, including those belonging to another system. Files on other systems are defined as *remote*. Remote files are accessed through CICS function shipping of file control requests to the remote region that owns the file.

However, this does not apply to files accessed in RLS mode, or coupling facility data tables, which are always defined as local files.

The resource attributes needed by CICS for remote files are not specific to the access method used. You can therefore define both remote BDAM files and remote VSAM files using RDO.

If you name a REMOTESYSTEM, you may also supply a REMOTENAME, which is the name of the file used in the remote system.

If you specify a REMOTESYSTEM name that corresponds to the SYSIDNT of the CICS region in which the file definition is installed, CICS installs the definition as a local file. Otherwise, CICS installs the definition as a remote file.

When a file definition is installed as a remote file, only the following attributes are used:

REMOTESYSTEM
REMOTENAME
RECORDSIZE
KEYLENGTH

The other attributes are used when the same file definition is installed as a local definition. For more information, see Defining remote resources for function shipping.

Coupling facility data tables

Coupling facility data table support provides a method of file data sharing, using CICS file control, without the need for a file-owning region, and without the need for VSAM RLS support. Data is held in a coupling facility list structure, in a table that is similar in many ways to a shared user-maintained data table.

Unlike user-maintained data tables, coupling facility data tables do not have to be pre-loaded from a source data set. Loading a coupling facility data table is controlled by the DSNAME and LOAD attributes of

the file resource definition, which allows CFDTs to be populated entirely by the application programs that use them by specifying LOAD(NO).

The way you use LOAD(YES) and the DSNAME attributes allows you to control loading of a CFDT in various ways, such as:

1. Any CICS region can load the coupling facility data table. The first file open for the CFDT loads it, regardless of which CICS region issues the open. The data set is opened read-only by the loading CICS. All file definitions for the table specify LOAD(YES) and the DSNAME of a source data set. If you use this approach, ensure that the same data set is named on each file definition, otherwise the data set named on the first to be opened is the one that is loaded into the CFDT. CICS does not verify that the DSNAMEs are the same for all files that refer to the same CFDT.
2. One CICS region can be made responsible for loading the coupling facility data table. The loading region contains a file definition for the CFDT that specifies LOAD(YES) and the DSNAME for the data set, which is opened read-only by the loading CICS. Other CICS regions do not need access to the source data set, but they cannot open the CFDT until the loading region has opened it. The file definitions for the CFDT in non-loading regions must also specify LOAD(YES) but omit the DSNAME.

   You can restrict access to a coupling facility data table until after it has been loaded by using two (or more) file names that refer to the same coupling facility data table. To control access in this way:

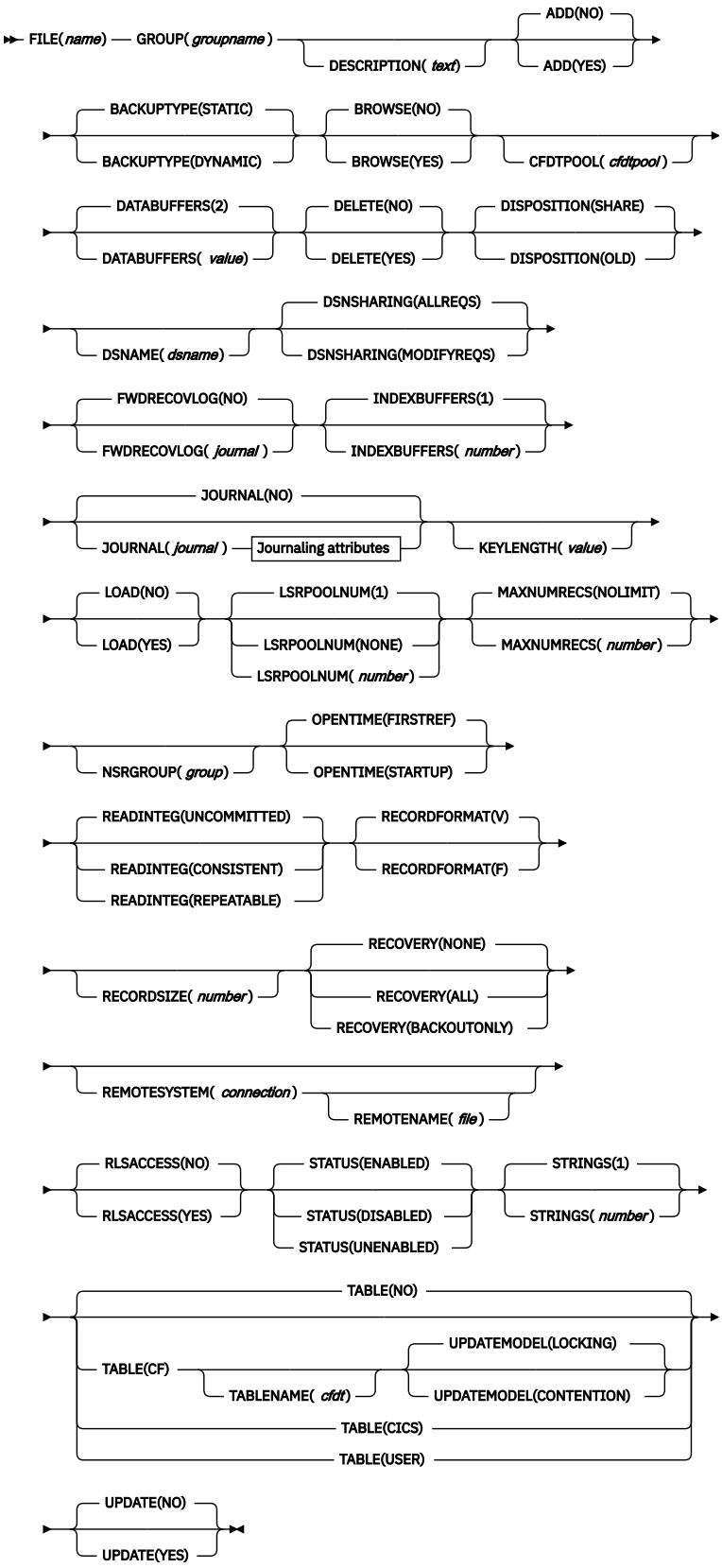
   - Define only one file name as being capable of loading the data table, by specifying LOAD(YES) and DSNAME(*datasetname*) Do not refer to this file name from application programs.
   - Define another file (or files) for application program use, ensuring that this file definition cannot initiate table loading. Specify LOAD(YES), but ensure the data set name is not specified on the DSNAME attribute, and that there is no DD statement for this file (or files).
   - Ensure that the load-capable file is opened before any application programs are likely to require access to the data table. For example, define the table-loading file name with OPENTIME(STARTUP) to ensure that the file is opened automatically toward the end of CICS initialization.
   - Ensure that application programs access the data table by referring to a filename that does not load the data.
3. Some hybrid of the previous two approaches can be used, where some CICS regions can load the table, and others require it to be loaded on their behalf.

Shared data tables

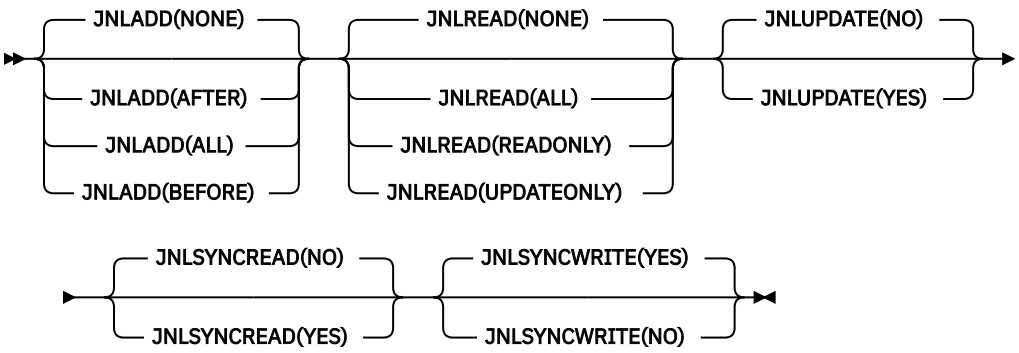
CICS-maintained data tables and user-maintained data tables are collectively referred to as *shared data tables*.

FILE attributes

Describes the syntax and attributes of the FILE resource.



Journaling attributes



ADD({NO|YES})
Specifies whether records can be added to the file.

BACKUPTYPE({STATIC|DYNAMIC})
CICS VSAM files can be defined as eligible for backup while open for update. This attribute is not used for files defined with RLSACCESS(YES), or if the recovery options are defined in the ICF catalog. To force CICS to use this attribute instead of the recovery options in the catalog, set the **NONRLSRECOV** system initialization parameter to FILEDEF. For files that are accessed in RLS mode, you must specify the backup type on the data set definition in the ICF catalog.

This attribute is ignored for coupling facility data tables and, if there are any recovery attributes defined in the ICF catalog for a source data set associated with the table, these also are ignored. A CFDT is not eligible for backup-while-open (BWO).

Possible values are:

DYNAMIC
Specify DYNAMIC together with the RECOVERY attribute of ALL to make the file eligible for backup while it is open for update.

STATIC
The file is not eligible for backup while open for update.

BROWSE({NO|YES})
Specifies whether records can be retrieved sequentially from the file.

CFDTPOOL(*cfdtpool*)
Specifies the name of the coupling facility data table pool containing the table defined by this file definition. This attribute is required if you specify TABLE(CF).

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

Coupling facility data tables can be separated (for purposes of accounting, security, administration, and so on) into groups (pools) of CFDTs. The names of all coupling facility data table pools must be unique within the sysplex, but can be the same as the names of other types of pools, such as TS data-sharing pools.

Opening a file that references a coupling facility data table requires that a coupling facility data table server for the named pool is running in the MVS in which the open request is issued. If the required server has not been started, the file open request fails.

Note: The CFDTPOOL attribute is meaningful only for CFDTs. You can specify a pool name for a file that is not defined as TABLE(CF), but CICS ignores it. If you then alter the file definition to reference a coupling facility data table, the CFDTPOOL name comes into effect.

DATABUFFERS({2|*value*})
Specifies the number of buffers to be used for data. Use a value in the range 2 (the default) through 32767. The minimum value you can specify is one more than the number of strings defined in the STRINGS attribute.

DELETE({NO|YES})
Specifies whether records can be deleted from the file.

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

DISPOSITION({SHARE|OLD})
Specifies the disposition of this file.

OLD
Equivalent to the DISP=OLD parameter in JCL.

SHARE
Equivalent to the DISP=SHR parameter in JCL.

DSNAME(*dsname*)
Specifies the data set name (as known to the operating system) to be used for this file. DSNAME can be 1 through 44 characters, conforming to the rules for MVS data set names (see the DSNAME parameter in the z/OS MVS JCL Reference).

Acceptable characters:

```
A-Z 0-9 $ @ # . -
```

At file open time, if no JCL statement exists for this file, the open is preceded by a dynamic allocation of the file using this DSNAME. If the file definition refers to a data table (CICS, USER, or CF) the DSNAME must be that of a VSAM base KSDS. It cannot be a path or alternate index data set.

The DSNAME specified on a DD statement for this file in the CICS start-up JCL takes precedence over the DSNAME specified in this file definition.

Coupling facility data tables
If the file definition specifies LOAD(YES) and it is not already opened, DSNAME specifies the name of the source data set from which the table is to be loaded. Alternatively, you can specify the source data set on a DD statement in the CICS startup JCL. The specified data set must be a VSAM base KSDS.

If there is a path or alternate index associated with the source data set, then any updates made via the file are not reflected in either the source data set or its associated alternate indexes. A coupling facility data table is entirely independent of its source data set after loading has completed.

If you want table loading to be initiated by the opening of another file specified by a different file definition, omit this attribute. In this case, also ensure that the file name is not specified on a DD statement in the CICS JCL. Attempts to open the file fail until CFDT loading has been initiated. For more information about loading a coupling facility data table from a data set, see Coupling facility data tables.

If LOAD(NO) is specified, this attribute is not required and is ignored.

DSNSHARING({ALLREQS|MODIFYREQS})
Specifies whether VSAM data set name sharing is used for the VSAM file. The possible values are:

ALLREQS
Data set name sharing is set in the ACB when the file is opened and is therefore used for all file requests.

MODIFYREQS
Data set name sharing is set in the ACB when the file is opened only if an operation of DELETE, ADD, or UPDATE is set for the file.

FILE(*name*)
Specifies the name of the file. The name can be up to 8 characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

The name must not start with a numeric character.

FWDRECOVLOG({NO|*journal*})
Specifies the journal that corresponds to the MVS system logger log stream that is to be used for forward recovery.

This attribute is ignored for coupling facility data tables and, if there are any recovery attributes defined in the ICF catalog for a source data set associated with the table, these also are ignored. A CFDT is not forward recoverable.

NO
Forward recovery logging is not required for this file.

journal
The number that identifies the journal that CICS is to use for the forward recovery log. CICS journal names are of the form DFHJ*nn* where *nn* is in the range 01 through 99. The after images for forward recovery are written to the MVS log stream that corresponds to journal name DFHJ*nn*.

Note: In CICS Transaction Server for z/OS, DFHJ01 is not the system log.

This attribute is used by CICS only if the following conditions are satisfied:

- RECOVERY(ALL) is specified
- RLSACCESS(NO) is specified
- No recovery attributes are defined in the ICF catalog

If you define the recovery attributes for a file in the ICF catalog entry for the corresponding data set, CICS always uses the ICF catalog recovery attributes and ignores those in the FILE resource. To force CICS to use the FILE resource attributes instead of the recovery options in the catalog, set the **NONRLSRECOV** system initialization parameter to FILEDEF. You can alter the recovery attributes defined in the ICF catalog by using the IDCAMS ALTER command. This is not prevented while there are ACBs open for a data set. However, if you change the recovery attributes, be aware of the possible effect on data integrity.

CICS takes a copy of the recovery attributes for a data set from the ICF catalog on the first open-for-update in a sequence of open requests for a data set. This means that a single CICS region is not affected by an update to recovery attributes. However, if a data set is opened in RLS mode and the attributes on the ICF catalog are modified, a second CICS region could open the same data set for update and copy a different set of attributes, with a risk to data integrity.

If you must alter recovery attributes defined in the ICF catalog (for example, to change the forward recovery log stream name), quiesce the data set before making any changes. This ensures that the data set cannot be used in RLS mode until you have made the change and unquiesced the data set.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

INDEXBUFFERS({1|*number*})
Specifies the number of buffers to be used for the index. Use a value in the range 1 through 32767. The minimum value you can specify is the number of strings defined in the STRINGS attribute.

JNLADD({NONE|BEFORE|AFTER|ALL})
Specifies the add operations you want recorded on the journal nominated by the JOURNAL attribute. Possible values are:

AFTER
Add the file control write operation to the journal after the VSAM I/O operation.

ALL
Add the file control write operation to the journal both before and after the VSAM I/O operation has completed.

BEFORE
Add the file control write operation to the journal before the VSAM I/O operation.

NONE
Do not add write operations to the journal.

JNLREAD({NONE|UPDATEONLY|READONLY|ALL})
Specifies the read operations you want recorded on the journal nominated by the JOURNAL attribute. Possible values are:

ALL
Add all read operations to the journal.

NONE
Do not add read operations to the journal.

READONLY
Add only READ ONLY operations (not READ UPDATE operations) to the journal.

UPDATEONLY
Add only READ UPDATE operations (not READ ONLY operations) to the journal.

JNLSYNCREAD({NO|YES})
Specifies whether you want the automatic journaling records, written for READ operations to the journal specified by JOURNAL, to be written synchronously or asynchronously.

JNLSYNCWRITE({YES|NO})
Specifies whether you want the automatic journaling records, written for WRITE operations to the journal specified by JOURNAL, to be written synchronously or asynchronously.

JNLUPDATE({NO|YES})
Specifies whether you want REWRITE and DELETE operations recorded on the journal nominated by the JOURNAL attribute.

JOURNAL({NO|*journal*})
Specifies whether you want automatic journaling for this file. The journaled data is in the format of the VSAM record and is used for user controlled journaling.

The data to be journaled is identified by the JNLADD, JNLREAD, JNLSYNCREAD, JNLSYNCWRITE, and JNLUPDATE attributes.

For a CICS-maintained data table, journaling is performed only for requests that result in VSAM I/O requests.

For a user-maintained data table or a coupling facility data table journaling is not performed for any file control operations. However, although automatic journaling for these tables is not supported, if you specify a journal number, CICS tries to open the log stream for the specified journal when opening the file.

Note: Automatic journaling is independent of logging to the system and forward recovery logs, as specified by the RECOVERY and FWDRECOVLOG attributes.

Possible values are:

NO
No automatic journaling is to take place for this file.

number
The number that identifies the journal that CICS is to use for the autojournal. CICS journal names are of the form DFHJ*nn*, where *nn* is in the range 01 through 99.

Note: In CICS Transaction Server for z/OS, DFHJ01 is not the system log.

KEYLENGTH(*value*)
Specifies the length in bytes of the logical key of records in remote files, and in coupling facility data tables that are specified with LOAD(NO).

If KEYLENGTH is not defined here, the KEYLENGTH option must be specified on file control commands in the application programs that refer to this file. If KEYLENGTH is not defined here and is not specified in the application program, and the key is longer than 4 characters, the default value is 4.

Remote files
The range for key length is 1 through 255.

Coupling facility data tables
The range for key length is 1 through 16. Key length is required only if LOAD(NO) is specified.

You can, optionally, specify a key length for coupling facility data tables specified with LOAD(YES), in which case you should be aware of the following:

- The key length is obtained from the ICF catalog entry for the data set from which the table is loaded. If you specify a key length, the key length must match that specified for the source data set, otherwise attempts to open the file fail with an error message.
- If, when opening the file, CICS finds that the CFDT has already been created, and the key length is different from that used when loading the data set, the open fails.

If you specify a key length for a file that is not a remote file, or does not refer to a CFDT, it has no effect unless the file is redefined, either as a remote file or to reference a CFDT. Note, however, that if you specify a key length, the value returned by an INQUIRE FILE command is as follows:

- If the file is open, CICS returns the value obtained from VSAM, which can be different from that specified on the file definition.
- If the file is closed, CICS returns the value specified on the file definition.

The value for this attribute must be the same throughout the sysplex in all file definitions that reference the same coupling facility data table.

LOAD({NO|YES})
Specifies whether the coupling facility data table is to be loaded from a source data set when first opened.

NO
Means that the coupling facility data table does not require loading from a source data set; it is fully usable by application programs as soon as it is open. The table is loaded by the application programs that use it, which is the default method for a coupling facility data table.

YES
Means that the coupling facility data table has to be loaded from a source data set before it is fully usable; the application programs that use this coupling facility data table rely on it containing

the records from a source data set. Loading does not have to be completed before data can be accessed.

This attribute is meaningful only for files defined with the TABLE(CF) attribute. You can specify the LOAD attribute for a file that is not defined as TABLE(CF), but CICS ignores it. (CICS-maintained and user-maintained tables are loaded automatically always from a source data set.) If you then alter the file definition to reference a coupling facility data table, the LOAD attribute comes into effect.

Ensure that the value for this attribute is the same throughout the sysplex in all file definitions that reference the same coupling facility data table.

For more information about using this attribute, see Coupling facility data tables.

LSRPOOLID({1|2|3|4|5|6|7|8|NONE})
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS.

The value specified for LSRPOOLID in existing definitions is transferred to the new option LSRPOOLNUM.

LSRPOOLNUM({1|*number*|NONE})
Specifies the identity of the local shared resource pool. The default value for LSRPOOLNUM is 1, unless a value has been specified for the NSRGROUP attribute, in which case the default value for LSRPOOLNUM is NONE.

NONE
Specifies that the data set associated with this file uses VSAM nonshared resources (NSR).

You cannot specify NONE for a CICS shared data table (CICS or user-maintained), because these types of data tables must use an LSR pool. However, this restriction does not apply to a coupling facility data table, for which you can specify NONE.

VSAM nonshared resources (NSR) are not supported for transactions that use transaction isolation. Specify ISOLATE(NO) when you define transactions that access VSAM files that use NSR. You can also function ship the file request to a remote region. The DFHMIRS program that carries out the request is defined with an EXECKEY of CICS. A CICS-key program has read and write access to CICS-key and user-key storage of its own task and all other tasks, whether or not transaction isolation is active.

number
Identifies the number of the VSAM shared resource pool that is used by the VSAM data set associated with this file. The value must be in the range 1 through 255. The data set is defined as using VSAM local shared resources (LSR). Define the buffers, strings, and other resources explicitly in an LSRPOOL resource definition that corresponds to the assigned LRPOOLNUM value.

By default, if the file definition specifies RLSACCESS(YES), the LSRPOOLNUM value is ignored when CICS opens the file. However, if you change a file definition that specifies an LSR pool from RLSACCESS(NO) to RLSACCESS(YES), you are advised to keep the LSRPOOLNUM value. LSRPOOLNUM ensures that, if the file is switched at any time from RLS to LSR mode, the file correctly references an LSR pool.

MAXNUMRECS({NOLIMIT|*number*})
Specifies the maximum number of records (entries) to be accommodated in the data table. You can use this attribute to prevent a runaway transaction from using:

- All the storage in the server's pool if the table is a coupling facility data table
- All the storage in the MVS data space if the table is a CICS- or user-maintained table.

This attribute is meaningful only for files defined with CICS, USER, or CF for the TABLE attribute. You can specify MAXNUMRECS for a file that is defined with TABLE(NO), but it has no effect. If you then alter the file definition to reference a data table, the MAXNUMRECS value comes into effect.

NOLIMIT
There is no user-specified limit placed on the number of records that can be stored in the table. CICS imposes a limit of 2,147,483,647, the maximum fullword value.

number
Specifies the maximum number of records allowed in the table, in the range 1 through 99999999. If you are setting a limit for a recoverable coupling facility data table, specify a value that is about 5 to 10% greater than the maximum number of records the table is expected to contain. This is to allow for additional records that can be created internally for processing recoverable requests. The margin you allow for this internal processing depends on the level of use of the coupling facility data table, and the nature of that use. An effect of this internal processing is that the NOSPACE condition with a RESP2 of 102 can be raised on a WRITE or REWRITE request to a recoverable coupling facility data table that apparently has fewer records than the MAXNUMRECS limit that has been specified for it.

NSRGROUP(*group*)
Specifies a symbolic name (up to 8 characters) to group file definitions that refer to the same VSAM base data set. The value is purely symbolic and need not refer to any particular file definition. It is merely necessary that all file definitions that must be grouped have the same name. You do not have to specify this attribute to ensure correct processing, but if you do not provide it, performance of your system might be degraded.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

The NSRGROUP attribute takes effect only for files referencing data sets that use VSAM nonshared resources. The NSRGROUP attribute must not be coded for a data table. It is associated with the VSAM concept of data set name sharing which causes VSAM to create a single control block structure for the strings and buffers required by all the files that relate to the same base data set.

When the first member of such a group of files is opened, the total number of strings to be allocated for all file entries in the group must be specified to VSAM (with the BSTRNO value in the Access Control Block). The VSAM control block structure is built this time regardless of whether the first file to be opened is associated with a path or base data set. The value of BSTRNO is calculated at this time by adding the STRINGS values in all the file definitions with the same NSRGROUP attribute. After the first file in the group is opened, any new files added to the group do not affect the VSAM control block structure already built. This would change only if all the files open against the base cluster were closed and then reopened.

Data set name sharing is forced by CICS as the default for all VSAM files. Data set name sharing is not in effect if a file is opened for read-only processing with DSNSHARING=MODIFYREQS. A file with DSNSHARING=MODIFYREQS still, however, contributes to the BSTRNO calculation.

If a file is using VSAM nonshared resources, and you do not provide an NSRGROUP attribute, the VSAM control block structure might be built with insufficient strings for later processing. When this happens, VSAM invokes the dynamic string addition feature to provide the extra control blocks for the strings as they are required. This mechanism is, however, inefficient and the extra storage is not released until the end of the CICS run.

For files specifying that VSAM local shared resources are to be used (LSRPOOLNUM=n, where n is in the range 1 - 255), NSRGROUP has no effect.

Figure 1 on page 57 Shows an example of how to specify the required file control definition for a VSAM base data set and alternate index path.

```
CEDA DEFINE FILE(VSAM10B)  GROUP(xxxxxx)
            DSNAME(DTGCAT.VSAM10B)
            DISPOSITION(SHARE)  ADD(YES)
            BROWSE(YES)  DELETE(YES)  READ(YES)
            UPDATE(NO)  RECORDFORMAT(F)
            STRINGS(8)  LSRPOOLNUM(NONE)
            RECOVERY(NONE)  NSRGROUP(GROUP1)
            INDEXBUFFERS(8)  DATABUFFERS(9)
CEDA DEFINE FILE(VSAM10P)  GROUP(xxxxxx)
            DSNAME(DTGCAT.VSAM10P)
            LSRPOOLNUM(NONE)  DISPOSITION(SHARE)
            STRINGS(5)  NSRGROUP(GROUP1)
            BROWSE(YES)  DELETE(NO)  READ(YES)
            ADD(NO)  UPDATE(NO)  RECORDFORMAT(F)
            RECOVERY(NONE)  INDEXBUFFERS(5)
            DATABUFFERS(6)
```

Figure 1. VSAM base data set and alternate index path definition.

OPENTIME({FIRSTREF|STARTUP})
Specifies when the file is opened. Possible values are:

FIRSTREF
The file remains closed until a request is made to open it by using one of the following methods:

- A main terminal command
- An EXEC CICS SET FILE OPEN command in an application program
- An implicit open

STARTUP
The file is opened immediately after CICS initialization by an automatically initiated CICS transaction (CSFU), unless the status of the file is UNENABLED when the file remains closed.

READ({YES|NO})
Specifies whether records on this file can be read.

READINTEG({UNCOMMITTED|CONSISTENT|REPEATABLE})
Specifies the level of read integrity required for files defined with RLSACCESS(YES). Read integrity does not apply to non-RLS access mode files, CICS shared data tables, or coupling facility data tables.

You can use READINTEG to set a default level of read integrity for a file. The default level of read integrity is used by programs that do not specify one of the API read integrity options UNCOMMITTED, CONSISTENT, or REPEATABLE on the READ, READNEXT, or READPREV commands. However, if an application program uses one of these explicitly to specify read integrity, the API option overrides any value specified on this READINTEG attribute.

Note: You can specify read integrity options only on CICS file control API commands or in CICS file resource definitions. You cannot use the equivalent parameter on the DD statement for files opened by CICS.

You can specify CONSISTENT or REPEATABLE in a file resource definition, to make read integrity available to programs written before these options were available on the API, and without having to modify those programs. However, if you do this, be aware that enforcing consistent or repeatable reads can introduce unexpected deadlocks. Programs might also encounter the LOCKED condition.

CONSISTENT
The record is read with consistent read integrity. If the record is being modified by another transaction, the READ request waits until the update is complete, the timing of which depends on whether the data set is recoverable or unrecoverable:

- For a recoverable data set, the READ request completes when the updating transaction completes its next sync point or rollback.
- For an unrecoverable data set, the READ completes as soon as the VSAM request that performing the update completes.

CONSISTENT is valid only if you also specify RLSACCESS(YES)—the resource definition is rejected with an error if you specify CONSISTENT for a non-RLS file.

REPEATABLE
The record is read with repeatable read integrity. If the record is being modified by another transaction, the READ request waits until the update is complete, the timing of which depends on whether the data set is recoverable or unrecoverable:

- For a recoverable data set, the READ request completes when the updating transaction completes its next sync point or rollback.
- For an unrecoverable data set, the READ completes as soon as the VSAM request that performing the update completes.

After the read completes, a shared lock remains held until sync point. This guarantees that any record read within a unit-of-work cannot be modified while the task makes further read requests. Error responses such as NOTFND might not be repeatable.

REPEATABLE is valid only if you also specify RLSACCESS(YES)—the resource definition is rejected with an error if you specify REPEATABLE for a non-RLS file.

UNCOMMITTED
The record is read without read integrity. CICS obtains the current value of the record as known to VSAM. No attempt is made to serialize this read request with any concurrent update activity for the same record. The record returned might be a version updated by another transaction, but not yet committed, and this record can change if the update is subsequently backed out.

Note:

1. UNCOMMITTED is the same level of integrity that is provided by those releases of CICS that do not support the READINTEG attribute.
2. Specify UNCOMMITTED for any kind of data table. Any value other than UNCOMMITTED is allowed if RLSACCESS(YES) but is ignored if TABLE(CF), TABLE(CICS), or TABLE(USER) is also specified for the file.

RECORDFORMAT({V|F})
Specifies the format of the records on the file.

F
The records are fixed length. For VSAM files, specify this only if the VSAM access method services definition specifies fixed size records (that is, the average size is equal to the maximum size), and all the records in the file are of that size.

F is invalid for user-maintained data tables and coupling facility data tables

V
The records are variable length. All user-maintained data tables and coupling facility data tables must be specified as variable length. Otherwise, CICS returns an error message stating that RECORDFORMAT(F) conflicts with TABLE(CF) or TABLE(USER) options and is ignored.

RECORDSIZE(*number*)
Specifies the maximum length in bytes of records in a remote file or a coupling facility data table. The size specified can be in the range 1 through 32767.

For coupling facility data tables only
This value is required if the file definition for the table specifies LOAD(NO).

You can also specify this attribute if LOAD(YES) is specified (for example, to make it easier for switching the file definition between LOAD(NO) and LOAD(YES)). However, if you specify LOAD(YES), the record size value must match that for the source data set, otherwise CICS fails to open the table. There are three conditions in which CICS can detect an error because of an incorrect record size with LOAD(YES):

1. Before opening the table, CICS verifies that the VSAM-defined record size for the data set from which the coupling facility data table is to be loaded is the same as the size, if any, in the file definition. If the record size is different, CICS returns error message DFHFC7081.

2. The record size (if specified) on the file definition is the same as that defined to VSAM for the data set, but on opening the table, CICS finds the table is already loaded with data of a different record size. This is probably because the data was loaded from a different data set from the one specified by this file definition. In this case CICS returns error message DFHFC7082.
3. The file definition for the table being opened specifies a record size, but not a data set name because the table is to be loaded by the opening of a different file. If the table has already been created, the open of a file specifying a different record size fails with message DFHFC7083.

To avoid these errors, ensure that the value for this attribute is the same throughout the sysplex in all file definitions that reference the same coupling facility data table, or omit it altogether for files that specify LOAD(YES).

If you specify a record size for a file that is not a remote file, or does not refer to a CFDT, it has no effect unless the file is redefined, either as a remote file or to reference a coupling facility data table. Note, however, that if you specify a record size, the value returned by an INQUIRE FILE command is as follows:

- If the file is open, CICS returns the value obtained from VSAM, which can be different from that specified on the file definition.
- If the file is closed, CICS returns the value specified on the file definition.

Note: For coupling facility data tables, if you can keep the record size to 63 bytes or less, there is a significant gain in performance as a result of the way records are held in the coupling facility.

RECOVERY ({NONE | BACKOUTONLY | ALL})
Specifies the type of recovery required for the file.

This attribute is not used for files defined with RLSACCESS(YES), or if the recovery options are defined in the ICF catalog. If LOG is defined in the ICF catalog, CICS ignores the RECOVERY option and takes the LOG value from the ICF catalog, even for files defined with RLSACCESS(NO). If LOG(ALL) is specified in the ICF catalog, CICS also takes the LOGSTREAMID and BWO values from the ICF catalog. To force CICS to use this attribute instead of the recovery options in the catalog, set the **NONRLSRECOV** system initialization parameter to FILEDEF.

For files that are accessed in RLS mode, you must specify the recovery parameters with the data set definition in the ICF catalog. See Defining files as recoverable resources for more information.)

For coupling facility data tables and user-maintained tables that are defined with a source data set, any recovery attributes in the ICF catalog are ignored. The recovery attributes are a property of the file not the associated data set.

For coupling facility data tables, the recovery attribute must be the same throughout the sysplex in all file definitions that reference the same coupling facility data table.

ALL
Except for coupling facility data tables, which manage their own recovery and do not use the services of log manager or recovery manager, before images are recorded in the system log, and after images in the journal specified in the FWDRECOVLOG attribute.

Records written to the FWDRECOVLOG are independent of any automatic journaling options that might be set.

RECOVERY=ALL together with FWDRECOVLOG provide a means of separating the needs of a forward recovery utility from those of automatic journaling. Additional information, unavailable via automatic journaling, is recorded on the FWDRECOVLOG. RECOVERY=ALL plus FWDRECOVLOG is the preferred way to provide forward recovery support.

Existing forward recovery utilities that used the JREQ=(WU,WN) and JID=FCT macro settings can still be used with these settings. The RDO equivalents of these automatic journaling settings are JNLADD=BEFORE, JNLUPDATE=YES, and the JOURNAL attribute.

For CICS-maintained data tables, the data table and its source data set are logged, journaled, and recovered together.

For user-maintained tables, specifying ALL has the same effect as specifying BACKOUTONLY: Only dynamic backout is provided. There is no forward recovery support for user-maintained tables.

For coupling facility data tables you cannot specify ALL.

Note: When ALL is specified for VSAM ESDS files, CICS is unable to perform backout of ADDs. To cope with this situation, code user exit XFCLDEL to avoid the file being closed because of the error.

BACKOUTONLY
Except for coupling facility data tables, which manage their own recovery and do not use the services of log manager or recovery manager, before images are recorded in the system log.

For CICS-maintained data tables, BACKOUTONLY specifies that the data table and its source data set are recoverable. They are both updated in step and, if required, recovered in step.

For user-maintained tables, BACKOUTONLY specifies only dynamic backout. No log records are written and, therefore, there is no recovery at emergency restart.

For coupling facility data tables, BACKOUTONLY is permitted only if the coupling facility data table is defined with UPDATEMODEL(LOCKING). You cannot specify this attribute for UPDATEMODEL(CONTENTION). Specifying BACKOUTONLY implies that a coupling facility data table is UOW-recoverable. This means that updates made to the CFDT within a unit of work are backed out if the unit of work fails, or if CICS or the CFDT server fails while the unit of work is in-flight, or if MVS fails.

Note: When BACKOUTONLY is specified for VSAM ESDS files, CICS is unable to perform backout of ADDs. To cope with this situation, code user exit XFCLDEL to avoid the file being closed because of the error.

NONE
There is no recovery logging for this file.

REMOTENAME(*file*)
Specifies, if the file resides on a remote system, the name by which this file is known in the system or region in which it is resident. The name can be up to 8 characters in length. If REMOTENAME is not specified, the name given in the FILE attribute is used.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

If you specify a remote name, CICSPlex® SM uses that name when assigning the file to a related system. If you specify a remote system but not a remote name, the local name (that is, the name of this file definition) is used in both the target and related systems.

REMOTESYSTEM(*connection*)
Specifies the name of the connection to the remote system in which the file is resident.

For IPIC connections, REMOTESYSTEM specifies the first 4 characters of the IPCONN name on the IPCONN definition that is in service and acquired.

For MRO and APPC connections, REMOTESYSTEM specifies the CONNECTION name on the CONNECTION definition.

If you specify REMOTESYSTEM, you can also supply a REMOTENAME, to specify the name of the file in the remote system.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

Note: If you modify a resource definition from RLSACCESS(NO) to RLSACCESS(YES), you must remove the remote system name. Otherwise CICS continues to function ship file requests.

RESSECNUM
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS. For more information, see Obsolete attributes.

RLSACCESS({NO|YES})
Specifies whether CICS is to open the file in RLS mode.

NO
The file is not to be opened in RLS mode. If you specify RLSACCESS(NO) or allow it to default, CICS opens the file in LSR or NSR access mode, depending on the LSRPOOLNUM attribute. If you also specify LSRPOOLNUM(NONE), the access mode is NSR; if you specify LSRPOOLNUM(*number*), the access mode is LSR.

YES
The file is to be opened in RLS mode. If you specify RLSACCESS(YES), it takes precedence over the LSRPOOLNUM attribute, which is ignored when the FILE is opened.

Specifying RLSACCESS(YES) alters the effect of some other attributes defined in the FILE definition, as shown in Table 1 on page 61.

Table 1. Effects of RLSACCESS(YES) on other FILE attributes

Attribute	Effect of RLSACCESS(YES)
PASSWORD	Ignored.
LSRPOOLNUM	Ignored.
DSNSHARING	Ignored.
STRINGS	Ignored; RLS access-mode files always have 1024 strings.
REMOTESYSTEM REMOTENAME RECORDSIZE KEYLENGTH	The meanings of these attributes are unchanged. However, dual FILE definitions of local and remote is of less value for RLS access-mode files. With RLS, there might be many file-owning regions (FORs) instead of one, and a local CICS region has the choice of several FORs.
DATABUFFERS INDEXBUFFERS	Ignored.
TABLE	TABLE(CICS) is not allowed; an error message is issued if specified. TABLE(USER) or TABLE(CF) is allowed. The source data set (if there is one) is accessed in RLS mode to load the data table, after which requests access the data table directly using data table services.
RECOVERY	Ignored. The recovery attribute is obtained from the ICF catalog for an RLS file.
FWDRCOVLOG	Ignored. The forward recovery log stream name is obtained from the ICF catalog for an RLS file.
BACKUPTYPE	Ignored. The type of backup is determined by the DFSMSdss backup utility for RLS.

Note:

1. Provided that a file is opened in RLS mode, any values specified for PASSWORD, LSRPOOLNUM, DSNSHARING, STRINGS, DATABUFFERS, and INDEXBUFFERS are ignored, as described in Table 1

on page 61. However, if you use a CEMT, or EXEC CICS, SET FILE command to change the value of RLSACCESS from YES to NO, these values are no longer ignored, and CICS uses them when the file is closed and reopened in non-RLS mode.

2. CICS always takes the RLS access mode from the file resource definition and you cannot override this using the RLS=NRI or RLS=CR parameter on a DD statement.

STATUS({ENABLED|DISABLED|UNENABLED})
Specifies the initial status of the file following a CICS initialization with START=COLD or START=INITIAL. You can change the status of a closed file with the main terminal transaction CEMT. The status of a file (ENABLED, DISABLED, or UNENABLED) following a CICS restart is recovered to its status at the previous shutdown.

DISABLED
Any request against this file from a command-level application program causes the DISABLED condition to be passed to the program.

ENABLED
Normal processing is allowed against this file.

UNENABLED
This prevents the file being opened by an implicit open from an application program. Any such attempt to access the file raises the NOTOPEN condition. By contrast, an explicit request to open the file (for example, a CEMT or EXEC CICS SET FILE OPEN command) changes the status to ENABLED before attempting to open the file.

This attribute is ignored for FILE resources that are dynamically generated by a CICS bundle. The initial status of a FILE resource is derived from the initial status of the bundle that defines the resource.

STRINGS({1|*value*})
Specifies the number, in the range 1 through 255, of concurrent requests that can be processed against the file. When the number of requests reaches this value, CICS queues any additional requests until one of the active requests terminates. This applies both to files using shared resources, and to those not using shared resources. Note that if the file definition specifies RLSACCESS(YES), STRINGS value is ignored; you always get 1024 strings with RLS mode access.

For files using local shared resources, this number is not used by VSAM. It is used by CICS, not only as described here, but also to calculate the default value in the buffer pool definition.

Notes:

1. When choosing a STRINGS value, be aware that a proportion (20%) of the specified number of strings is reserved by CICS for use in read-only requests
2. When choosing a STRINGS value for an ESDS, consider the following:
   - If an ESDS is used as an 'add-only' file (that is, it is used only in write mode), you must use a string number of 1. Any string number greater than 1 can significantly affect performance, because of exclusive control conflicts that occur when more than one task attempts to write to the ESDS at the same time.
   - If an ESDS is used for both writing and reading, with writing being 80% of the activity, it is better to define two file definitions—using one file for writing and the other for reading.
3. For user-maintained data tables and coupling facility data tables, the STRINGS value does *not* limit the number of concurrent requests against the table. However, the value does limit the number of concurrent requests during the loading of a user-maintained table.
4. For CICS-maintained data tables, the STRINGS value limits the number of concurrent requests to update the table. It does not limit the number of concurrent read-only requests.

TABLE({NO|CICS|USER|CF})
Specifies the type of data table that you require.

CF
A coupling facility data table (CFDT). This remains independent of its source data set, and changes to the table are not reflected in the corresponding source data set, if there is one. A source data set is optional for a CFDT, and is specified by LOAD(YES) on the file definition.

If you specify CF, also specify:

- CFDTPOOL, to give the name of the coupling facility pool in which the table resides
- LOAD, to specify whether the table is to be loaded from a source data set (or let this default to NO)
- UPDATEMODEL to specify whether the table is to use the CONTENTION or the LOCKING update model (or let this default to LOCKING)
- RECORDFORMAT as V (or let this default to V)
- MAXNUMRECS with the value you require.

A coupling facility data table requires a coupling facility data table server. For information about how to start a coupling facility data table server, see Setting up and running a coupling facility data table server.

CICS
A CICS-maintained data table. This automatically reflects all modifications made to the table in its source data set. If you specify CICS, also specify:

- LSRPOOLNUM with a value of 1 through 255
- MAXNUMRECS with the value you require.

NO
Data table not required.

USER
A user-maintained table. This remains independent of its source data set, and changes to the user-maintained table are not reflected in corresponding source data set. If you specify USER, also specify:

- LSRPOOLNUM with a value of 1 through 255
- RECORDFORMAT as VARIABLE (or let this default to VARIABLE)
- MAXNUMRECS with the value you require.

TABLENAME(*cfdt*)
Specifies the name of the coupling facility data table that is accessed through this file definition. The name can be up to 8 characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

If you omit this attribute when TABLE(CF) is specified, it defaults to the name specified for the FILE. To enable CICS regions to share a coupling facility data table, the file definitions installed in each region must specify the same CFDTPOOL name and TABLENAME (or FILE name when TABLENAME is not used). The TABLENAME need only be unique within its pool.

Note that the table name is not only an identifier for the table, but is also used as the resource name in security checks.

This attribute is meaningful only for files defined with the TABLE(CF) attribute. You can specify a table name for a file that is not defined as TABLE(CF), but CICS ignores it. If you then alter the file definition to reference a coupling facility data table, the TABLENAME attribute comes into effect.

UPDATE({NO|YES})
Specifies whether records on this file can be updated.

UPDATEMODEL({LOCKING|CONTENTION})
Specifies the type of update model to be used for a coupling facility data table.

LOCKING
Specifies that the CFDT is updated using the locking model. This means that records are locked when they are read for update, so that they cannot be changed by any other units of work until the update request has been completed. For recoverable tables, the update request is completed at sync point. For unrecoverable tables, the update request is completed when a REWRITE, DELETE, or UNLOCK command is completed. LOCKING is the default for a file that specifies TABLE(CF). With the LOCKING model, the CFDT can be defined as:

- **Unrecoverable**, meaning that CFDT updates are not backed out if a unit of work fails, and the locks are only held for the duration of a request. You specify that a CFDT is not recoverable by specifying RECOVERY(NONE).
- **Recoverable**, or UOW-recoverable, meaning that updates made to the CFDT within a unit of work are backed out if the unit of work fails, or if CICS or the CFDT server fails while the unit of work is in-flight, or if MVS fails. You specify that a CFDT is recoverable by specifying RECOVERY(BACKOUTONLY).

A recoverable CFDT that uses the locking model is like a recoverable file or data set, except that it does not survive a failure of the coupling facility in which it resides. There is no forward recovery for a coupling facility data table.

CONTENTION
Specifies that the CFDT is updated using the contention model. This means that records are not locked when they are read for update. An error is returned on a subsequent REWRITE or DELETE if the record was changed or deleted by another task after it was read for update. The CFDT must be unrecoverable (RECOVERY(NONE)), meaning that updates are not backed out if a unit of work fails.

The value for this attribute must be the same throughout the sysplex in all file definitions that reference the same coupling facility data table.

This attribute is meaningful only for files defined with the TABLE(CF) attribute. You can specify the update model for a file that is not defined as TABLE(CF), but CICS ignores it. If you then alter the file definition to reference a coupling facility data table, the UPDATEMODEL attribute comes into effect.

Installing file definitions

This procedure uses the CEMT and CEDA transactions to install a FILE definition. As an alternative to CEMT, you can use the **EXEC CICS SET FILE** command in a user-written transaction to disable and enable the file.

About this task

Procedure

1. If the file already exists, ensure that it is closed and disabled.
   Use the following command:

   ```
   CEMT SET FILE(filename) CLOSED DISABLED
   ```
2. Install the file definition.
   Use the following command:

   ```
   CEDA INSTALL GROUP(groupname) FILE(filename)
   ```
3. Optional: When you have successfully installed the group containing the file, use CEMT to open and enable the file.

Perform this step only if the file is not already defined as ENABLED, and you want to open the file explicitly. Use the following command:

```
CEMT SET FILE(filename) OPEN ENABLED
```

FILE resource definitions

FILE resource definitions describe the physical and operational characteristics of files.

IPCONN resources

An IPCONN resource defines a Transmission Control Protocol/Internet Protocol (TCP/IP) communication link to a remote system. This communication link is known as an IPIC connection.

Some of the inbound attributes of the IPIC connection are specified by the TCPIPSERVICE definition that is named on the TCPIPSERVICE option of the IPCONN definition.

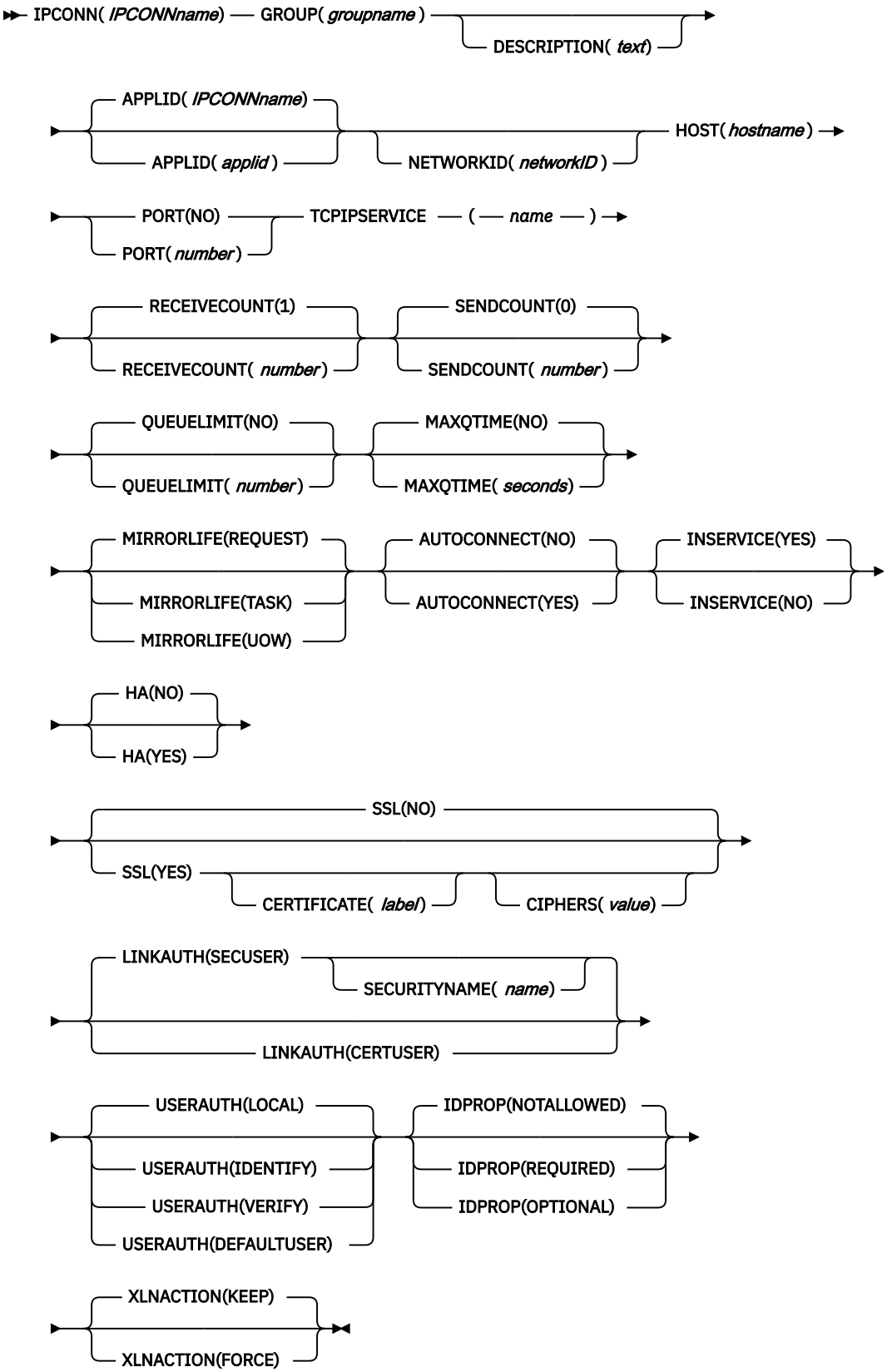
The REMOTESYSTEM name on a PROGRAM definition can refer to an IPCONN definition through its IPCONN name. This attribute is used for distributed program link.

For guidance on defining IPCONN resources, see Defining IPIC connections.

IPCONN attributes

Describes the syntax and attributes of the IPCONN resource.

Syntax



Attributes

APPLID({*IPCONNname*|*applid*})
Specifies the application identifier (APPLID) of the remote system. If the remote system is a CICS region, its APPLID is defined on the APPLID parameter. The value for *applid* can be up to 8 characters in length and must start with an alphabetic character.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

For connections to an extended recovery facility (XRF) CICS region, specify the generic APPLID of the remote region.

If you do not supply an APPLID, CICS uses the IPCONN name.

When the IPCONN is used to connect to an HA cluster, this field must contain the 1 to 8 character value that identifies the cluster. This value distinguishes this IPCONN from all other installed IPCONNs in the region. The name of an APPLID that points at a specific CICS region cannot have the same name as an HA cluster.

Duplicate APPLIDs follow these rules:

- You cannot install two or more IPCONN definitions that specify the same APPLID *and* the same NETWORKID. The combination of APPLID and NETWORKID can be used to ensure unique naming of systems across the network. See the description of the NETWORKID option.
- You *can* install an IPCONN definition that specifies the same APPLID as the NETNAME of an installed MRO, APPC, or LUTYPE6.1 CONNECTION definition.
- If an installed IPCONN definition has the same name as an installed CONNECTION definition, the APPLID of the IPCONN definition must match the NETNAME of the CONNECTION definition. If they do not, the resulting message depends on the situation:
  - DFHIS3009 if the error is detected during IPCONN autoinstall
  - DFHAM4913 if the error is detected during IPCONN installation
  - DFHZC6312 if the error is detected during CONNECTION installation or autoinstall

  The IPCONN definition takes precedence over the CONNECTION definition; that is, if an IPCONN and a CONNECTION have the same name, CICS uses the IPCONN.
- A CONNECTION definition and an IPCONN definition with the same NETNAME and APPLID do not require the same name to allow the possibility of using a distinct *sysid* for communication over TCP/IP rather than relying on the CICS default of routing all supported function with the IPCONN, if it exists.

These rules are validated at installation time.

AUTOCONNECT({NO|YES})
Specifies whether sessions are to be established when the IPCONN definition is installed (which can happen during CICS initialization, when you issue a subsequent CEDA INSTALL command, or when you use the CEMT or EXEC CICS SET TCPIP OPEN command to start communication with TCP/IP). If the connection cannot be made at these times because the remote system is unavailable, you can later acquire the link by using the CEMT or EXEC CICS SET IPCONN(*name*) INSERVICE ACQUIRED command, unless the remote system becomes available in the meantime and initiates communications.

NO
CICS does not try to establish sessions when the IPCONN is installed.

YES
CICS tries to establish sessions when the IPCONN is installed.

For connectivity to be achieved when you install the IPCONN definition, note these conditions:

1. The TCPIPSERVICE definition named on the TCPIPSERVICE option of this IPCONN definition must also be installed in this region and must specify PROTOCOL(IPIC).
2. Corresponding IPCONN and TCPIPSERVICE definitions must be installed in the remote region:
   a. The HOST option of the IPCONN definition on the remote region must specify this region.
   b. The PORT option of the IPCONN definition on the remote region must specify the same port number as that specified on the PORTNUMBER option of the local TCPIPSERVICE definition named by this IPCONN.
   c. The TCPIPSERVICE definition on the remote region (named by the IPCONN definition on the remote region) must specify PROTOCOL(IPIC) and, on its PORTNUMBER option, the same port number as that specified by the PORT option of this IPCONN.

You cannot specify AUTOCONNECT(YES) when PORT(NO) is specified.

CERTIFICATE(*label*)

Specifies the label of an X.509 certificate to be used as a client certificate during the SSL handshake when the IPIC connection is acquired, if the TCPIPSERVICE resource identified by the HOST and PORT definitions is defined with SSL(CLIENTAUTH). If this attribute is omitted, the default certificate defined in the key ring for the CICS region user ID is used.

Certificate labels can be up to 32 bytes long.

The certificate must be stored in a key ring in the external security manager database. For more information, see Building a key ring manually.

If you specify this attribute, you must also specify SSL(YES).

CIPHERS(*value*)

The CIPHERS attribute can be specified in either of two ways:

- **Recommended:** The name of the SSL cipher suite specification file, which is a z/OS UNIX file in the `security/ciphers` subdirectory of the directory that is specified by the **USSCONFIG** system initialization parameter. For example, if **USSCONFIG** is set to `/var/cicsts/dfhconfig` and **CIPHERS** is set to `strongciphers.xml`, the fully qualified file name is `/var/cicsts/dfhconfig/security/ciphers/strongciphers.xml`. For more information, see Creating an SSL cipher suite specification file.
- A string of up to 56 hexadecimal digits that is interpreted as a list of up to 28 2-digit cipher suite codes.

When you use the CEDA transaction to define the resource, CICS automatically initializes the attribute with a default list of acceptable codes. For CICS to initialize the attribute, the KEYRING system initialization parameter must be specified in the CICS region where you are running CEDA. If KEYRING is not set, CICS does not initialize the attribute. The default list of 2-digit ciphers is 3538392F3233.

You can reorder the cipher codes or remove them from the initial list. However, you cannot add cipher codes that are not in the default list for the specified encryption level. To reset the value to the default list of codes, delete all of the cipher suite codes. The field is automatically repopulated with the default list.

When you use DFHCSDUP to define the resource, the CIPHERS attribute is not populated with a default value. If the resource is used to establish a secure connection and the CIPHERS attribute is empty, CICS uses the default list of 2-digit ciphers. An INQUIRE on the resource that is created in this manner, returns an empty string for the CIPHERS value.

Any unsupported ciphers are removed at run time. A list of the removed ciphers is reported in messages DFHSO0145 and DFHSO0146.

For more information, see Cipher suites and cipher suite specification files.

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. No restrictions apply to the characters that you can use. However, if you use parentheses, ensure that each left parenthesis has a matching right one. If you use the CREATE command, code 2 apostrophes for each single apostrophe in the text.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

HA({NO|YES})
Specifies whether the IPCONN can be used to connect to a high-availability cluster.

NO
This is the default value. The IPCONN cannot be used to connect to a high-availability cluster.

YES
The IPCONN must connect to a region that is part of a high-availability cluster.

HOST(*hostname*)
Specifies the host name of the remote system or its IPv4 or IPv6 address. The name can be up to 116 characters long. You can specify IPv4 and IPv6 addresses in a number of acceptable formats. See IP addresses for more information about address formats.

If you specify an IPv6 address (or a host name that resolves to an IPv6 address), ensure that you are operating in a dual-mode (IPv4 and IPv6) environment and that the client or server that you are communicating with is also operating in a dual-mode (IPv4 and IPv6) environment. For more information about IPv6, see Understanding IPv6 and CICS.

The HOST attribute must contain only alphanumeric characters, hyphens (-), colons (:), or periods (.), although you cannot use colons when specifying a character host name instead of an IP address. CICS validates the host name at define time. The host name can be entered in uppercase, lowercase, or mixed case characters, but if a character host name is specified instead of an IP address, the host name is converted to lowercase in the IPCONN definition.

HOST is ignored when the SENDCOUNT attribute is zero. HOST is a required attribute when SENDCOUNT is greater than zero.

IDPROP({NOTALLOWED||OPTIONAL|REQUIRED})
Specifies whether the distributed identity is transmitted to the connected system by the sender. The IDPROP attribute is meaningful only if a connection extends outside a sysplex and is used primarily to prevent distributed identities being transmitted between enterprises. If the connection is between systems in the same sysplex, the connection operates as if IDPROP(OPTIONAL) is specified and ignores any other setting.

Distributed identities flow only if the connected systems meet all the following criteria:

- Both systems are participating in identity propagation. See Support and requirements for identity propagation.
- The systems are in the same sysplex or are connected by using SSL.
- The receiving system has USERAUTH(IDENTIFY) specified in the IPCONN resource definition.

NOTALLOWED
A user ID associated with the sending transaction is sent for requests that use this connection. NOTALLOWED is the default value.

REQUIRED
A distributed identity is required for requests that use this connection. If REQUIRED is specified, the receiving system must support distributed identities. The user ID associated with the sending transaction is not sent.

OPTIONAL
A distributed identity is sent, if available. The user ID associated with the sending transaction is also sent.

INSERVICE({NO|YES})
Specifies the status of the IPCONN resource when it is installed.

NO
The connection cannot receive messages or transmit output.

YES
The connection is available for use.

IPCONN(*IPCONNname*)
Specifies the name of this IPCONN definition. The name can be up to 8 characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

If this IPCONN is to be used for distributed program link (DPL) between CICS TS 3.2 or later regions, or transaction routing between CICS TS 4.1 or later regions, or function shipping file control, transient data, or temporary storage requests between CICS TS 4.2 or later regions, that use IPIC connectivity, its name must match the 4-character *local name* (SYSID) by which CICS knows the remote system, padded with four trailing blanks.

Note: You can specify the name (SYSID) of the remote, target region, of a DPL request in these ways:

- The REMOTESYSTEM option of the installed PROGRAM definition
- The SYSID option of the EXEC CICS LINK PROGRAM command
- The dynamic routing program

The IPCONN name can be the same as the name of an installed MRO or APPC CONNECTION definition.

LINKAUTH({CERTUSER|SECUSER})
Specifies how the user ID for link security is established in a CICS system with security initialized (SEC=YES).

CERTUSER
TCP/IP communication with the partner system must be configured for SSL and a certificate must be received from the partner system during SSL handshake.

The IPCONN resource must refer to a TCPIPSERVICE resource that is defined with SSL(CLIENTAUTH).

The received certificate must be defined to the external security manager so that it is associated with a user ID, which is used to establish link security.

SECUSER
Specifies that the user ID specified in the SECURITYNAME attribute is used to establish link security.

The default value is LINKAUTH(SECUSER)

MAXQTIME({NO|*seconds*})

Specifies the maximum time that queued allocate requests, waiting for free sessions on this connection, can wait before the queue is purged.

Note:

- The maximum queuing time is used only if a limit to the length of the queue is specified on the QUEUELIMIT option.
- The time limit is applied only when the queue length has reached the QUEUELIMIT value.

NO

CICS maintains the queue of allocate requests that are waiting for a free session. No limit is set on the length of time that requests can remain queued, although the DTIMOUT mechanisms can apply to individual requests.

seconds

The approximate maximum time, in seconds, that allocate requests waiting for a free session can be queued, when this connection appears to be unresponsive; *seconds* must be in the range 0 - 9999.

When the queue of allocate requests reaches its maximum length (specified by the QUEUELIMIT attribute), and a new allocate request is received for the connection, if the rate of processing for the queue indicates that, on average, the new allocate takes more than the maximum queue time, the queue is purged, and message DFHIS500 is issued. When the queue is purged, queued allocate requests return SYSIDERR.

No further queuing takes place until the connection has successfully freed a session. At this point, CICS issues a DFHIS5001 message and resumes normal queuing.

MIRRORLIFE({REQUEST|TASK|UOW})

Specifies the minimum lifetime of the mirror task for function-shipped file control, transient data, and temporary storage requests received by this region. Normally, mirror tasks are terminated as soon as possible to keep the number of active tasks to a minimum and to avoid holding on to a session for long periods. However, for some applications it is more efficient to retain both the mirror task and the session until the next sync point, although this retention is not required for data integrity. For example, a transaction that issues many READ FILE requests to a remote system might be better served by a single mirror task instead of a separate mirror task for each request. In this way, you can reduce the overhead of allocating sessions on the sending side and attaching mirror tasks on the receiving side. The specified MIRRORLIFE value is not reflected in the lifetime of the mirror task until a file control, transient data, or temporary storage request is function shipped.

REQUEST

The mirror task terminates as soon as possible. For unrecoverable work that does not require a mirror to hold on to a session in the resource-owning region (for example, a non-update READ FILE request) this is after the request has been processed. For unrecoverable work that requires the mirror to hold on to a session (for example, file browse requests), this is as soon as the need for the hold state has been completed (for example, the ENDBR is issued). For recoverable work, the mirror remains until the next sync point. This is the default value.

TASK

The mirror task remains available to the application that issues the remote request until the task of the application ends. This value saves the overhead of reallocating a session and reattaching the mirror task for a subsequent request following a user sync point. Do not specify this value when long-running tasks might be used to function ship file control, transient data, or temporary storage requests because once the session is allocated to a task it is not freed until the task ends, even it is not being used. The session is not available for use by other tasks until it is freed.

UOW

The mirror transaction remains available to the application that issues the remote request until the next sync point is issued. This value saves the overhead of reestablishing communication with the mirror transaction if the application has more function-shipping file control, transient data, or temporary storage requests in this unit of work.

This parameter is valid when it is specified on the IPCONN of the resource-owning region. Specify the TASK or UOW values with this parameter with caution, especially if DPL requests with SYNCONRETURN or TRANSID are used.

NETWORKID(*networkID*)
Specifies the network ID of the remote system. The remote system network ID is either its z/OS Communications Server NETID or, for VTAM=NO systems, the value of its UOWNETQL system initialization parameter.

If NETWORKID is not specified, CICS assumes that the remote system is in the same network as the local system. In this instance, CICS uses the z/OS Communications Server NETID, or the value of the UOWNETQL system initialization parameter, of this CICS (that is, the CICS on which this definition is installed).

Specify NETWORKID if you want to connect to a remote system that is in a different network and therefore has a different z/OS Communications Server NETID or UOWNETQL value. In this instance, it might be possible for two or more remote systems to have the same APPLID. Although CICS APPLIDs must be unique within a sysplex, you might, for example, want to connect to a system outside the sysplex or in a different sysplex. The combination of APPLID and NETWORKID attributes ensures that the remote system is referred to by a unique name.

The NETWORKID value must match the remote system network ID.

When it is not specified, the NETWORKID value is derived when the IPCONN resource is first installed and is not changed between warm starts, even if the local NETID value changes.

The name can be up to 8 characters in length and follows assembler language rules. It must start with an alphabetic character.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

PORT(NO|*number*)
Specifies the decimal number of the port on which the remote region listens.

NO
This IPCONN resource is not used for outbound requests (that is, it has no send sessions) when connecting from the CICS Transaction Gateway.

NO is not valid for CICS to CICS IPCONN resources.

NO forces the value of AUTOCONNECT to NO.

number
A value in the range 1 - 65535. The port number is combined with the HOST value to determine the destination for outbound requests on this IPCONN resource.

QUEUELIMIT({NO|*number*})
Specifies the maximum number of allocate requests that CICS is to queue while waiting for free sessions:

NO
No limit applies to the number of allocate requests that CICS can queue while waiting for a free session.

number
The maximum number of allocate requests, in the range 0 - 9999, that CICS can queue on the connection while waiting for a free session. When the number of queued allocate requests reaches this limit, subsequent allocate requests fail, returning SYSIDERR, until the queue drops below the limit.

RECEIVECOUNT({1|*number*})
Specifies, in the range 1 - 999, the number of receive sessions; that is, sessions that receive incoming requests. The number of receive sessions that are used depends also on the number of send sessions defined in the remote system. When the connection is established, these values are exchanged and the lower value is used.

SECURITYNAME(*name*)
Specifies the user ID used to establish link security.

In a CICS system with security initialized (SEC=YES), and with LINKAUTH(SECUSER) in use, the security name is used to establish the authority of the remote system.

The security name must be a valid RACF user ID on this region. Access to protected resources on this region is based on the RACF user profile and its group membership. If the security name is not a valid RACF user ID when the IPCONN is installed, CICS uses the default user ID for the security name.

The default value is the default user ID.

SENDCOUNT({0|*number*})
Specifies, in the range 0 - 999, the number of send sessions; that is, sessions that send outgoing requests. The number of send sessions that are used depends also on the number of receive sessions defined in the remote system. When the connection is established, these values are exchanged and the lower value is used. If 0 is specified, this IPCONN can process only incoming work. It cannot send requests to the connected system.

SENDCOUNT(0) is not valid for CICS to CICS IPCONN resources.

SENDCOUNT(0) forces PORT(NO). A SENDCOUNT value greater than zero requires PORT to have a numeric value.

An attempt to acquire an IPCONN resource that has SENDCOUNT(0) fails because no port is defined.

SSL({NO|YES})
Specifies whether Secure Sockets Layer (SSL) is used for encrypting the transmitted data. Note that, when using SSL, MAXSSLTCBs should be set to a value that is at least twice the number of IPIC connections that use SSL by the CICS region.

NO
The Secure Sockets Layer (SSL) is not used. No security checks are applied when the connection is being acquired. No encryption is applied to outbound messages.

YES
If the SEC system initialization parameter is set to YES, the Secure Sockets Layer (SSL) is used. If the TCPIPSERVICE resource identified by the HOST and PORT attributes is defined with SSL(CLIENTAUTH), CICS extracts the client certificate named in the CERTIFICATE attribute, and uses it when acquiring the IPIC connection to the partner system. SSL encryption processing is applied to all messages sent from this IPCONN resource. The level of encryption depends on the value of the CIPHERS option.

TCPIPSERVICE(*name*)
Specifies the name of a TCPIPSERVICE definition, with PROTOCOL(IPIC), that defines the attributes of the inbound processing for this IPCONN resource.

USERAUTH({LOCAL|IDENTIFY|VERIFY|DEFAULTUSER})
Specifies how the user ID for attach-time user security is established in a CICS system with security initialized (SEC=YES).

LOCAL
CICS does not accept a user ID or password from clients. All requests run under the link user ID or the default user ID if there is no link user ID.

IDENTIFY
Incoming attach requests must specify a user ID. Enter IDENTIFY when the connecting system is trusted to pre-authenticate users, for example, if it is another CICS or CICS TG system.

SSL client authentication must be in use or the connecting system must be in the same sysplex.

VERIFY
Incoming attach requests must specify a user ID and password. Specify VERIFY when connecting systems are unidentified and cannot be trusted.

DEFAULTUSER
CICS does not accept a user ID and password from the partner system. All requests run under the default user ID.

XLNACTION({KEEP|FORCE})
Specifies the action to be taken when a new logname is received from the partner system. Receipt of a new logname indicates that the partner has deleted its recovery information.

KEEP
Recovery information is kept, and no predefined actions are taken for indoubt units of work.

The connection cannot perform new work that requires sync level 2 protocols until all outstanding recoverable work with the partner (that is, indoubt UOWs, or information relevant to UOWs that were indoubt on the partner system under the old logname) is completed, using the CEMT or SPI interface. Therefore, because the connection is being used for CICS-to-CICS communication, which always uses the sync level 2 protocols, the connection cannot be acquired until all outstanding recoverable work with the partner has completed.

FORCE
Before any new work with the new logname is started, the predefined decisions for indoubt units of work (UOWs), as defined by the indoubt attributes of the TRANSACTION definition, are implemented. CICS also deletes any information retained for possible resolution of UOWs that were indoubt on the partner system.

Attention: Data integrity might be compromised if you use this option.

The XLNACTION parameter is ignored for IPCONN definitions used by the CICS Transaction Gateway.

Installing IPCONN definitions

To install new IPCONN definitions, put them in a group of their own which does not contain IPCONN definitions that have already been installed, then use CEDA INSTALL to install the group. You can also install IPCONN definitions individually.

About this task

For connectivity to be achieved when you install the IPCONN definition, consider the following resource requirements:

1. The TCPIPSERVICE definition named on the TCPIPSERVICE attribute of this IPCONN definition must also be installed in this region and must specify PROTOCOL(IPIC).
2. Corresponding IPCONN and TCPIPSERVICE definitions must be installed in the remote region."Corresponding" means that:
   a. The HOST attribute of the IPCONN definition on the remote region must specify this region.
   b. The PORT attribute of the IPCONN definition on the remote region must specify the same port number as that specified on the PORTNUMBER attribute of the local TCPIPSERVICE definition named by this IPCONN.
   c. The TCPIPSERVICE definition on the remote region, named by the IPCONN definition on the remote region, must specify PROTOCOL(IPIC) and, on its PORTNUMBER attribute, the same port number as that specified by the PORT attribute of this IPCONN.

IPCONN resource definitions

IPCONN resource definitions define the outbound attributes of an IP interconnectivity (IPIC) connection from a local CICS region to a remote CICS region or system.

JOURNALMODEL resources

A JOURNALMODEL resource defines the connection between a CICS journal name (or identifier) and the associated physical log streams managed by the MVS system logger, or between the journal name and the SMF log.

Although they are intended mainly for user journals, you can also define journal models for the system log and forward recovery logs (non-RLS only). However, for forward recovery logs, you are recommended to define all log stream names for forward recovery in the VSAM catalog. This is mandatory for VSAM files processed in RLS mode, but optional for non-RLS mode files.

Unlike the journal control table, you do not need to define a journal model for every journal that CICS uses. Instead, define some generic model definitions that describe the mapping to log stream names for the majority of your CICS journals. You may find that you can use the default models supplied by CICS and need not define any of your own. In addition to generic models, you can define the necessary specific models where special handling is required (for example, SMF logging, or merging with other log streams).

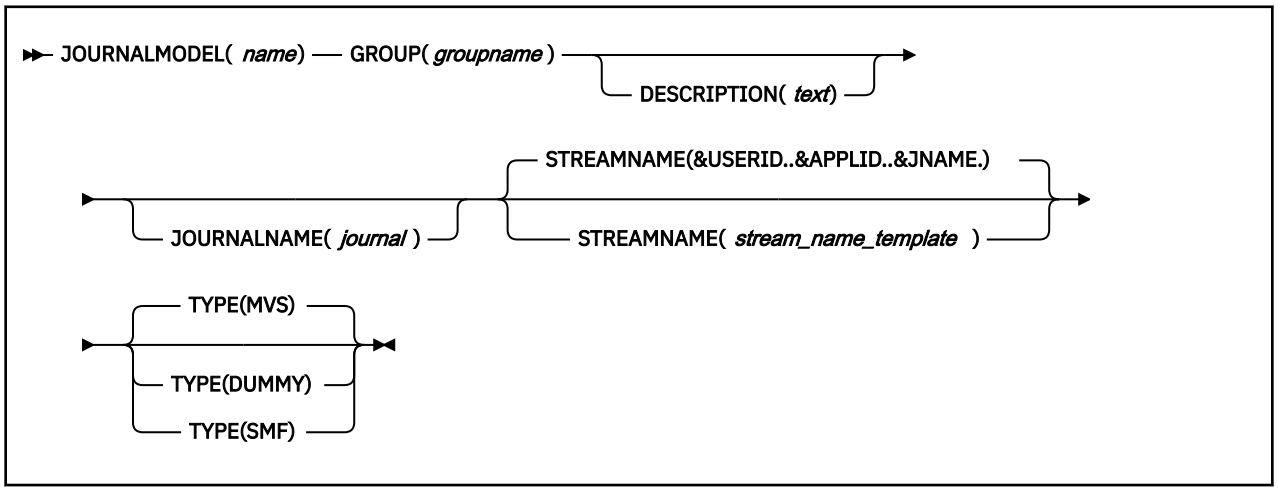
You can change JOURNALMODEL definitions at any time, but any journal entries that CICS has already created using model definition cannot reflect the change unless you first delete the existing entry using a DISCARD JOURNALNAME() command.

Compatibility note: For API compatibility with releases earlier than CICS Transaction Server for z/OS, CICS continues to support numeric journal identifiers in the range 01 through 99, for the following purposes:

- For file control autojournaling, as specified in FILE resources (or for BDAM files, on DFHFCT macro entries)
- For terminal control autojournaling, as specified in PROFILE resource definitions
- For forward recovery logging, as specified in FILE resources
- For user journaling using API journal commands, such as the EXEC CICS WRITE JOURNALNUM command.

JOURNALMODEL attributes

Describes the syntax and attributes of the JOURNALMODEL resource.



DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can

use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

GROUP (*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

JOURNALMODEL (*name*)
specifies the name of this JOURNALMODEL definition.

The journal model name is used to refer to a specific JOURNALMODEL definition in the CSD file—it does not have to correspond to a CICS journal name. However, the JOURNALMODEL name is also used as the JOURNALNAME if you omit the JOURNALNAME attribute.

The name can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

JOURNALNAME (*journal*)
specifies the journal names to which this definition applies. If you omit the JOURNALNAME attribute, the name you specify on the JOURNALMODEL attribute is used as the journal name. Name can be either the specific name of a journal or a generic name, although using a generic name for system log and log-of-logs models does not serve much purpose.

The name can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

The forms of the names you can define are as follows:

For system logs
To define a JOURNALMODEL for system logs, specify the name as DFHLOG for the primary system log stream, and as DFHSHUNT for the secondary log stream. Install one journal model only for each of these log streams in a CICS region.

CICS-supplied definitions for DFHLOG, DFHSHUNT, and DFHLGLOG are contained in group DFHLGMOD in DFHLIST.

For log-of-logs
To define a JOURNALMODEL for the log-of-logs, specify the name as DFHLGLOG. See Planning log streams for use by your log of logs (DFHLGLOG) for more information about the purpose of the log of logs.

For autojournals
For autojournals (file control and terminal control), the name must be of the form DFHJ*nn* where *nn* is a number in the range 1 through 99. The name can be either the specific name of a journal or a generic name.

For user journals
For user journals, the name can be up to 8 characters, and can be either the specific name of a journal or a generic name. If compatibility with releases earlier than CICS Transaction Server for z/OS, is required, the name must be of the form DFHJ*nn* where *nn* is a number in the range 1 through 99.

For forward recovery logs (non-RLS)
For non-RLS forward recovery logs, the name must be of the form DFHJ*nn* where *nn* is a number in the range 1 through 99. The name can be either the specific name of a journal or a generic name.

Note: You cannot define a journal model for use with VSAM RLS forward recovery logs. CICS obtains the fully-qualified LSN directly from the VSAM catalog, and therefore does not need a journal model to obtain the LSN.

You define generic names, using the special symbols %, +, and *, as follows:

- You can use the % or + symbols to represent any single character within a journal name.
- You can use the * symbol at end of a name to represent any number of characters. A default name of a single * is used to match any journal names that do match on a more specific name.

If there are several installed JOURNALMODEL definitions that match a journal name, CICS chooses the best match as follows:

1. If there is a JOURNALMODEL with a specific JOURNALNAME that exactly matches, CICS uses this model.
2. If there is no exact match, the journal name is compared with the matching generic entries and the most specific entry is used.

   In comparing names to see which one is more specific, the names are compared character by character. Where they first differ:

   - If one has a discrete character (not %, +, or *) and the other has a generic character (%, +, or *) the one with the discrete character is used.
   - If one has a % or a + and the other has a *, the one with % or + is used.
3. If there are duplicate JOURNALMODEL definitions (that is, definitions with the same JOURNALNAME), CICS uses the last one processed.

Attention: Take care when defining a completely generic journal name using only the single asterisk (*). This is particularly important if you have not defined a specific journal model for the system log (using journal name DFHLOG), and the log stream name is a fully-qualified literal name. If you define a journal model with JOURNALNAME(*) and do not define a journal model for the system log, CICS uses the log stream name defined on the generic model definition. This causes problems if other journals and forward recovery logs are assigned to the same log stream by means of the generic journal model.

STREAMNAME({&USERID..&APPLID..&JNAME.|*stream_name_template*})
specifies either an explicit MVS system logger log stream name, or a template used to construct the log stream name. STREAMNAME is applicable only to journal models defined with a LOGSTREAMTYPE of MVS.

The four symbolic names, from which you can use a maximum of three, are:

&USERID.
The symbolic name for the CICS region userid, which can be up to eight characters. If the region does not have a userid, the string 'CICS' will be used.

&APPLID.
The symbolic name for the CICS region APPLID as specified on the system initialization parameter, and which can be up to eight characters.

Note: If you are using XRF and you specify the APPLID system initialization parameter as APPLID=(generic_applid,specific_applid), it is the generic applid that CICS uses when resolving &APPLID..

&JNAME.
The symbolic name for a journal name that references, either by a specific or generic match, this journal model definition. &JNAME. can be up to eight characters in length.

&SYSID.
The symbolic name for the CICS region SYSID as specified on the SYSIDNT system initialization parameter. If SYSIDNT is not specified, the string 'CICS' will be used.

The default set of symbolic names is: &USERID..&APPLID..&JNAME.

For Example: &USERID..&APPLID..&JNAME. =

CICSHA##.CICSHAA1.DFHJ02

where:

CICSHA##
is the CICS region userid used by all the AORs.

CICSHAA1
is the applid of one AOR.

DFHJ02
is the journal name of an auto journal.

An alternative set of symbolic names could be:

&SYSID..&APPLID..&JNAME. =
SYSA.CICSHAA1.DFHJ02

where:

SYSA
is the character string as specified by the SYSIDNT system initialization parameter.

CICSHAA1
is the applid of one AOR.

DFHJ02
is the journal name of an auto journal.

CICS installs the JOURNALMODEL resource as defined, including the symbolic names.

stream_name_template
A log stream name can be either an unqualified name or a qualified name, as defined for MVS data set names:

- **Unqualified name**: 1 through 8 alphanumeric or national characters ($ # @), or a hyphen. The first character of the name must be alphabetic or national (A-Z $ # @).
- **Qualified name**: Multiple names joined by periods, up to a maximum of 26 characters. Each name in a qualified name must follow the rules for an unqualified name, with each qualified name (except the last) followed by a period. For example,

  ```
  name_1.name_2...name_n
  ```

  where the number of names is restricted by the 26-character limit.

For more information about the rules for qualified and unqualified data set names, see z/OS MVS JCL Reference.

You can construct log stream names consisting of a mixture of specific characters (from within the allowed set), and symbolic names for substitution. After substitution, the name must meet the rules for qualified and unqualified log stream names, and must not exceed 26 characters, including periods. Thus, if each name in a qualified name uses the maximum of eight characters, you are restricted to

three names only, with the first and second names, and the second and third names separated by a period. For example:

`CICSDA##.CICSDAA1.FWDRECOV`

for a forward recovery log stream. The log stream name is determined by symbolic substitution when a journal name is first resolved to a JOURNALMODEL definition.

By specifying the same log stream name for multiple CICS general logs, you can merge the log streams from different CICS regions. However, you cannot merge general log streams with the CICS system log, nor can you merge system logs from different CICS regions.

When merging log streams from different CICS systems, the log data blocks are written to their log streams in strict MVS system logger time-stamp sequence. However, the individual records from different CICS regions may not be in strict time-stamp sequence across different blocks

CICS log streams should not be merged with log streams generated by other products unless any programs that read the log stream are prepared to handle the formats.

Security note: When you have defined a log stream name to CICS and the MVS system logger, you must ensure that the required security authorizations are defined to RACF (or an equivalent external security manager). This security authorization is necessary before you attempt to start a CICS region that references a new log stream. RACF supports the LOGSTRM general resource class for this purpose.

TYPE({DUMMY|MVS|SMF})
specifies where the journal records are to be written. It can be up to five characters, and can have the following values:

DUMMY
No log records are to be written. For example, you can use this to suppress unwanted log records without changing an application, or without changing file or profile resource definitions.

If you do not want a system log or a log-of-logs, specify DUMMY on the JOURNALMODEL definitions for the DFHLOG, DFHSHUNT, and DFHLGLOG, as required.

MVS
Records are to be written to an MVS system logger log stream. The name of the log stream is specified in the STREAMNAME attribute.

SMF
Journal records are to be written in SMF format to the MVS SMF log instead of to an MVS system logger log stream.

Note: SMF is not allowed for the CICS system log or for forward recovery logs.

The default JOURNALMODEL

If CICS cannot find an installed JOURNALMODEL definition, it assumes a set of default attributes.

The attributes used are those of the following "built-in" default definition:

```
DEFINE JOURNALMODEL(OTHERS) GROUP(LOGS)
          JOURNALNAME(*)
          STREAMNAME(&USERID..&APPLID..&JNAME.)
          TYPE(MVS)
```

JOURNALNAME(*) is the default journal model that CICS uses if there is no matching JOURNALMODEL entry for a journal name.

Journal model resource definitions

Journal model definitions describe the association between a CICS journal name and the MVS system log streams or the SMF log.

Examples

Example of resource definition commands for CICS JOURNLMODELs.

Given the following set of definitions for a CICS AOR with applid CICSHAA3 and region userid CICSHA##:

```
1.  DEFINE JOURNALMODEL(USERJNL8) GROUP(LOGS)
           JOURNALNAME(DFHJ08)
           TYPE(SMF)
```

Records written to autojournal 08 or user journal 08 using an EXEC CICS WRITE JOURNALNAME(DFHJ08)... or EXEC CICS WRITE JOURNALNUM(08)... command are written in SMF format to the MVS SMF data set.

```
2.  DEFINE JOURNALMODEL(USERJNL9) GROUP(LOGS)
           JOURNALNAME(DFHJ09)
           TYPE(DUMMY)
```

Records written to autojournal 09 or user journal 09 using an EXEC CICS WRITE JOURNALNAME(DFHJ09)... or EXEC CICS WRITE JOURNALNUM(09)... command are not written to any log stream, though the application program receives a normal response.

```
3.  DEFINE JOURNALMODEL(UJ10TO19) GROUP(LOGS)
           JOURNALNAME(DFHJ1%)
           STREAMNAME(&USERID..MERGED.USRJRNLS)
           TYPE(MVS)
```

Records written to user journals 10–19 (DFHJ10–DFHJ19) are merged together on log stream CICSHA##.MERGED.USRJRNLS, together with records from any other CICS regions running under the CICSHA## userid and with the same JOURNALMODELs installed.

```
4.  DEFINE JOURNALMODEL(LOGOFLOG) GROUP(LOGS)
           JOURNALNAME(DFHLGLOG)
           STREAMNAME(CICSVR.SHARED.DFHLGLOG)
           TYPE(MVS)
```

File tie-up records and other records written by file control and the CICS log manager to journal DFHLGLOG for use by forward recovery products, such as CICSVR, are written to a shared log stream CICSVR.SHARED.DFHLGLOG. This log stream is shared by all the CICS regions in the sysplex in which this JOURNALMODEL resource definition is installed.

```
5. DEFINE JOURNALMODEL(JNLMODL1) GROUP(LOGS)
          JOURNALNAME(USERJNL*)
          STREAMNAME(&USERID..ANYCORP.&JNAME..UK)
          TYPE(MVS)
```

Records written to any user journals or autojournals that begin with the letters USERJNL are merged together on a log stream with a name that is obtained by substituting the CICS region userid for &USERID. and the journal name for &JNAME..

With only these examples installed, other forms of journaling, such as terminal control automatic message journaling defined with PROFILE ... JOURNAL(25), use the default JOURNALMODEL, with records written to log stream CICSHA##.CICSHAA3.DFHJ25. See “The default JOURNALMODEL” on page 79 .

JVMSERVER resources

A JVMSERVER resource defines the runtime environment for a JVM server.

The JVMSERVER resource defines the location of the JVM profile and the Language Environment® options that are required to create a Language Environment enclave and a JVM server in a CICS region.

JVMSERVER resources in CICS bundles

You can use a CICS bundle to create, edit, and install a JVMSERVER resource definition. If you create a JVMSERVER resource in this way, you must use the CICS bundle to manage the lifecycle of that resource, and you cannot manage the resource independently.

You can inquire on a JVMSERVER resource that is dynamically generated by a CICS bundle, by using the **EXEC CICS INQUIRE JVMSERVER** or **CEMT INQUIRE JVMSERVER** command.

You cannot issue the DISCARD command against a JVMSERVER resource that is dynamically generated by a CICS bundle. You must discard the CICS bundle, and CICS applies the operation to the JVMSERVER resource.

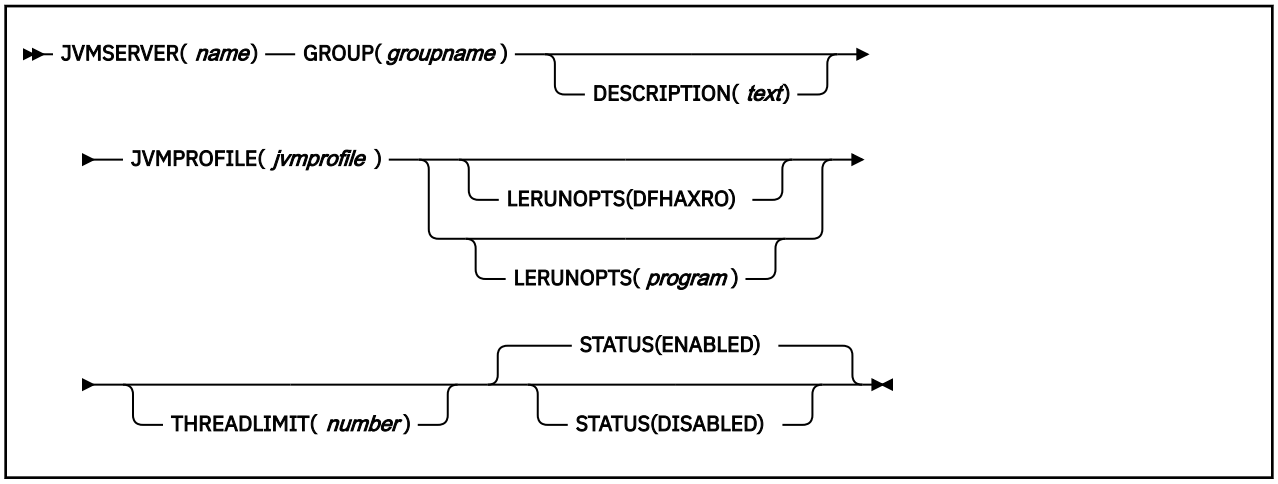
To modify the attributes of a JVMSERVER resource that was defined and installed in a CICS bundle, use the resource editor in the CICS Explorer to modify the definition in the CICS bundle, and install a new version of the CICS bundle or of the application with which it was deployed. You can use the SET JVMSERVER command to change the attributes of the dynamically generated resource, but these changes are not cataloged and are not recovered across a warm restart of CICS.

To change the status of a JVMSERVER resource that was defined and installed in a CICS bundle, enable or disable the CICS bundle. If you have disabled the CICS bundle, but you need to purge tasks that are still running in the JVM server, you can issue the SET JVMSERVER DISABLED command against the dynamically generated resource with the PURGE, FORCEPURGE, or KILL option to purge the tasks.

For more information about defining resources in CICS bundles, see Defining CICS bundles.

JVMSERVER attributes

Describes the syntax and attributes of the JVMSERVER resource.



JVMSERVER(*name*)

Specifies the 1 - 8 character name of the JVMSERVER resource.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . - _ % ? ! : | = , ;
```

It is preferable not to use names that start with DFH because these characters are reserved for use by CICS.

CICS can delete or discard them and replace them with CICS resources when you upgrade or provide new function or fixes.

DESCRIPTION(*text*)
In this field, you can provide a description of the resource that you are defining. The description text can be up to 58 characters in length. No restrictions apply to the characters that you can use. However, if you use parentheses, ensure that each left parenthesis has a matching right one. If you use the CREATE command, for each single apostrophe in the text code, use two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters that you enter are converted to uppercase.

The GROUP name can be up to 8 characters in length.

JVMPROFILE(*jvmprofile*)
Specifies the 1 - 8 character name of the JVM profile containing the JVM options for running a JVM server. The file containing the JVM profile must have a file extension of `.jvmprofile`.

- For a JVM profile for a JVMSERVER resource defined by online resource definition, the file path is specified by the system initialization parameter **JVMPROFILEDIR**.
- For a JVM profile for a JVMSERVER resource that is defined in a CICS bundle, the file path is relative to the root directory of the CICS bundle. The file containing the JVM profile must be packaged in the CICS bundle with the JVMSERVER resource. For more information, see Referencing zFS artifacts in a bundle.

The JVM profile name cannot consist of a single or double period. The set of acceptable characters in the JVM profile name is more restricted when the JVMSERVER resource is defined in a CICS bundle using CICS Explorer.

Acceptable characters for JVMPROFILE name in an online JVMSERVER resource:

```
A-Z a-z 0-9 @ # . - _ % & ? ! : | " = , ; < >
```

Acceptable characters for JVMPROFILE name in a JVMSERVER resource definition in a CICS bundle:

```
A-Z a-z 0-9 @ # - _ % ! =
```

LERUNOPTS(DFHAXRO|*program*)
Specifies the 1 - 8 character name of the program that defines the runtime options for the Language Environment enclave. DFHAXRO is a supplied program that provides a set of default values. The source for DFHAXRO is in the *hlq*.SDFHSAMP library if you want to change the defaults for any of the Language Environment runtime options.

If you want to use a different program, put the program in the *hlq*.SDFHLOAD library and specify the program name in uppercase characters.

STATUS(ENABLED|DISABLED)
Specifies the initial status of the JVMSERVER resource when it is installed.

ENABLED
The JVM server runtime environment is available for use.

DISABLED
The JVM server runtime environment is not available for use.

THREADLIMIT(15|*number*)
Specifies the maximum number of threads that are allowed in the Language Environment enclave for the JVM server. Each thread runs under a T8 TCB. You can specify a limit in the range of 1 - 256 threads.

If you specify a thread limit that exceeds the maximum of 2000 threads that is allowed for the CICS region, taking into account all other enabled and disabled JVMSERVER resources, CICS allocates the remaining threads up to 2000 to the resource as the thread limit value. If CICS is already at the maximum number of JVMSERVER threads, the resource installs in a disabled state.

Tip: Liberty JVM servers require a minimum of six threads to run. If you specify a threadlimit lower than six for a Liberty JVM server, the threadlimit and threadcount automatically raises to six to prevent your Liberty becoming unresponsive.

JVMSERVER resource definitions

A JVMSERVER resource definition defines the runtime environment for a JVM server. The JVMSERVER resource defines the location of the JVM profile, the Language Environment enclave options, and other configuration information to control the initialization of the JVM.

LIBRARY resources

Use the LIBRARY resource to define the physical and operational characteristics of a dynamic program LIBRARY concatenation.

The LIBRARY definition includes attributes that provide the name of the data set, or sets, within the LIBRARY resource and other details about the operational status of the LIBRARY.

DFHRPL is a special example of a LIBRARY which cannot be altered in a running CICS system, and does not appear as a resource in CEDA.

Data sets in a dynamic program LIBRARY concatenation can reside in the extended addressing space (EAS) of an extended address volume (EAV) DASD volume.

There is no 'group commit' for LIBRARY resources. Each LIBRARY in a CSD group is committed separately when the group is installed. To achieve an effect similar to group install, you can use the Disabled status to install a set of LIBRARY resources which would then be SET to an Enablestatus of Enabled when all resources are installed.

LIBRARY resources in CICS bundles

You can use a CICS bundle to create, edit, and install a LIBRARY resource definition. If you create a LIBRARY resource in this way, you must use the CICS bundle to manage the lifecycle of that resource, and you cannot manage the resource independently. You can inquire on a LIBRARY resource that is dynamically generated by a CICS bundle, by using the **EXEC CICS INQUIRE LIBRARY** or **CEMT INQUIRE LIBRARY** command. However, you cannot issue SET or DISCARD commands against a LIBRARY resource that is dynamically generated by a CICS bundle. You must issue the commands against the BUNDLE resource, and CICS applies them to the LIBRARY resource.

For more information about defining resources in CICS bundles, see Defining CICS bundles.

Private LIBRARY resources for applications

The LIBRARY resource is supported as a private resource for an application version. Each version of an application should include a private LIBRARY resource that represents a version-specific data set containing the load modules for the application.

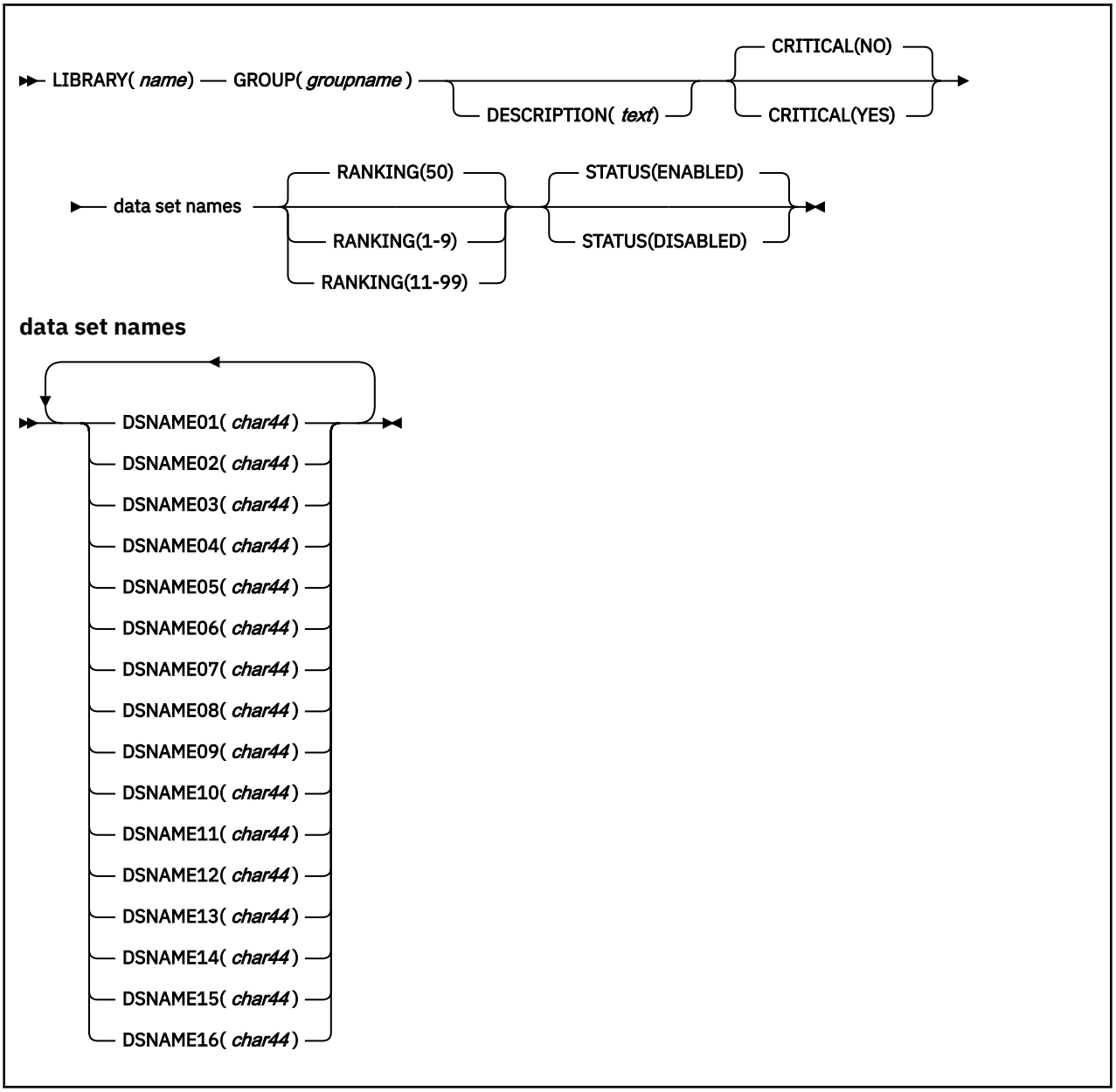
For a private LIBRARY resource that is defined in a CICS bundle that is packaged and installed as part of an application bundle or application binding bundle, the name of the LIBRARY resource is not used as the

DD name for the LIBRARY concatenation of data sets. Instead, CICS requests a unique DD name for the LIBRARY concatenation of data sets when the application is installed on the platform. The resource name can therefore be the same as LIBRARY names used elsewhere in the installation, or by different versions of the application. CICS issues message DFHLD0518 to state the DD name that z/OS has generated for the LIBRARY concatenation. You can also view the data set names for an installed application in the CICS Explorer.

For more information about private resources, see Private resources for application versions.

LIBRARY attributes

Describes the syntax and attributes of the LIBRARY resource.



CRITICAL({NO|YES})

Indicates whether this LIBRARY is critical to the running of CICS. This determines the behavior if the LIBRARY cannot be installed during startup, for example if a data set in the LIBRARY definition is not found or cannot be allocated for one of the following reasons:

- If the LIBRARY is CRITICAL

- If the LIBRARY is being created during CICS startup (through grouplist install, BAS install, a PLTPI program or restore from the catalog), a "GO or CANCEL" message is issued so that the operator can decide whether to override the criticality and allow CICS to start, or to cause the startup to be failed. If the reply is to continue with the startup, the LIBRARY is not recataloged as NONCRITICAL, but the critical status can be changed to NONCRITICAL if you decide that the LIBRARY is not to be regarded as CRITICAL in future.
- If the LIBRARY is not being created during CICS startup, the criticality of the LIBRARY has no impact on the behavior of the installation.

• If the LIBRARY is non-CRITICAL

- If the LIBRARY is being created during CICS startup (through grouplist install, BAS install, a PLTPI program or restore from the catalog), the LIBRARY is installed but disabled, a warning message is issued, and CICS startup continues. A later attempt can be made to resolve the problem and enable the LIBRARY.
- If the LIBRARY is not being installed during CICS startup, the criticality of the LIBRARY has no impact on the behavior of the installation.

DFHRPL, the static LIBRARY, is predefined as CRITICAL and cannot be changed. Any problems cause CICS startup to fail and an error message is issued.

For a private LIBRARY resource that is defined in a CICS bundle that is packaged and installed as part of an application bundle or application binding bundle, the CRITICAL attribute is ignored.

DSNAME01-16(Name)
The names of up to 16 data sets that contain program artifacts and that are to make up the LIBRARY concatenation. When the LIBRARY is installed, the data sets are concatenated together in the order in which they are named, for example, DSNAME01 before DSNAME02. It is not necessary for all of the data set attributes to be completed sequentially, for example, DSNAME01 might be blank. However, at least one of the data set attributes must have a value. See Defining a data set naming convention for further information.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

LIBRARY(*name*)
The name of the LIBRARY resource.

For a LIBRARY resource that is defined in a CICS region, a CICSPlex SM data repository, or a standalone CICS bundle, the resource name is used as the DD name for the LIBRARY concatenation of data sets. The name of the LIBRARY resource must therefore conform to the rules for a valid DD name (1-8 characters with the first character any of A-Z, $, # or @ and the remaining characters any of A-Z, 1-9, $, # or @). Avoid using names that conflict with other DD names used by the installation. The create of a public LIBRARY resource fails if the chosen name conflicts with a DD name already in use in the system for another public LIBRARY resource.

For a private LIBRARY resource that is defined in a CICS bundle that is packaged and installed as part of an application bundle or application binding bundle, the name of the LIBRARY resource is not used as the DD name for the LIBRARY concatenation of data sets. Instead, CICS requests a unique DD name for the LIBRARY concatenation of data sets when the application is installed on the platform. The resource name can therefore be the same as LIBRARY names used elsewhere in the installation, or by different versions of the application. CICS issues message DFHLD0518 to state the DD name

that z/OS has generated for the LIBRARY concatenation. You can also view the data set names for an installed application in the CICS Explorer.

The following reserved names are not accepted for a LIBRARY resource:

- DFHRPL or any name that starts with DFH
- Any name that starts with EYU
- CEEDUMP
- DUMMY
- JOBCAT
- JOBLIB
- STEPCAT
- STEPLIB
- SYSABEND
- SYSIN
- SYSMDUMP
- SYSOUT
- SYSUDUMP

LIBRARY resource names beginning with DFH or EYU are not accepted, to ensure that these names remain available in future for use by CICS. LIBRARY resource names beginning with CEE should be used with caution, but are accepted. SYSIN and SYSOUT and DUMMY are conventionally used as DD cards and are therefore inadvisable as LIBRARY resource names. The remainder are DD names reserved by MVS; DFHRPL is reserved for use in the CICS startup JCL as the name of the static LIBRARY.

RANKING(1-99)

RANKING is a fullword binary value that contains a decimal number that determines where this LIBRARY appears in the overall LIBRARY search order. A lower number indicates that this LIBRARY is searched for programs to load before other LIBRARY resources with higher ranking numbers. The ranking is analogous to the concatenation number of a data set in a LIBRARY concatenation, although it has some differences.

RANKING can take values 1 - 99, with a default of 50. The value 10 is reserved for DFHRPL, the static LIBRARY, and cannot be specified.

The DFHRPL concatenation is assigned a predefined ranking of 10, which cannot be changed. This value enables dynamically defined LIBRARY resources to be placed before the DFHRPL concatenation in the overall search order by giving them a ranking value less than 10.

Note:

- You should regard having LIBRARY resources with a ranking that places them before DFHRPL in the search order as a temporary situation.
- Although the predefined DFHRPL ranking of 10 is intended to discourage the placing of LIBRARY resources before DFHRPL in the search order, it does not limit the total number of LIBRARY resources that can be placed before DFHRPL providing the ranking between the LIBRARY resources themselves is not significant. If you specify a RANKING value of less than 10, a message to say that this LIBRARY appears ahead of DFHRPL in the search order is issued.
- No ranking change takes effect until a NEWCOPY or PHASEIN request is issued, or the program is loaded for the first time if it was not already loaded.
- The search order of LIBRARYs with the same ranking is undefined, and implies that the relative search order of the LIBRARY resources is unimportant. In practice, the search order is the order in which the LIBRARYs were installed.

- A LIBRARY resource that is defined in a CICS bundle as part of an application installed on a platform is private to that version of that application. These LIBRARY concatenations are only searched for tasks for that version of the application.
- LIBRARYs can have the same ranking but you should not use equal ranking values for two LIBRARY resources if their relative search positions are of any significance.
- The search positions of two LIBRARY resources of the same ranking are not reliable. The search order is related to the time each resource is installed, and can change after a COLD or INITIAL start or when changing the ranking of a LIBRARY. The resource that is installed first appears ahead of the other LIBRARY in the search order.

STATUS({ENABLED|DISABLED})
Indicates whether the LIBRARY is to be enabled or disabled when it is created. When a LIBRARY is created as enabled, CICS attempts to allocate and then concatenate the data sets, before finally opening the LIBRARY concatenation. If any of these steps fail, the steps that are already successful are undone, and the LIBRARY is installed as disabled. A message is issued that indicates which step failed.

When a LIBRARY is created with as disabled, CICS does not attempt to allocate or concatenate the data sets.

This attribute is ignored for LIBRARY resources that are dynamically generated by a CICS bundle. The initial status of a LIBRARY resource is derived from the initial status of the bundle that defines the resource.

Installing a LIBRARY resource definition by using CEDA

You install a new or updated LIBRARY resource definition to ensure that the LIBRARY resource is activated in the CICS region. This procedure shows how to use the CICS CEDA transaction to install a LIBRARY resource definition.

About this task

When you use CEDA to install a resource definition, you can set an initial status of either ENABLED or DISABLED, and the resource is installed in that status.

The following procedure installs the LIBRARY resource definition *libname* that is defined in the group *groupname*, and sets the initial resource status to ENABLED.

Procedure

Install the LIBRARY definition by using the following command:

```
CEDA INSTALL LIBRARY(libname) GROUP(groupname) STATUS(ENABLED)
```

What to do next

Defining the data sets to the CICS system
When you have successfully installed the LIBRARY resource definition and have also installed any programs, map sets, or other elements that comprise the application in the LIBRARY resource, you can then define the data sets to the CICS system, if you have not already done this.

Enabling a LIBRARY resource definition
If the LIBRARY resource was defined and installed as DISABLED, you can use the **EXEC CICS SET LIBRARY** or **CEMT SET LIBRARY** command to change its status to ENABLED.

LIBRARY resource definitions

LIBRARY definitions describe the attributes for dynamic program LIBRARY resources.

LSRPOOL resources

The LSRPOOL resource defines the size and characteristics of the local shared resources (LSR) pool. The LSR pool is a reserve of data buffers, strings, and Hiperspace buffers that VSAM uses when processing access requests for certain files.

A Hiperspace buffer may be used by VSAM to provide additional buffer capacity. For more information about MVS Hiperspace, see Creating and using hiperspaces in the z/OS MVS Programming: Extended Addressability Guide.

Up to 255 LSR pools can be defined concurrently in the system, each identified by its LSRPOOLNUM. This LSRPOOLNUM is used to associate a FILE with an LSR pool if that file is to use shared resources.

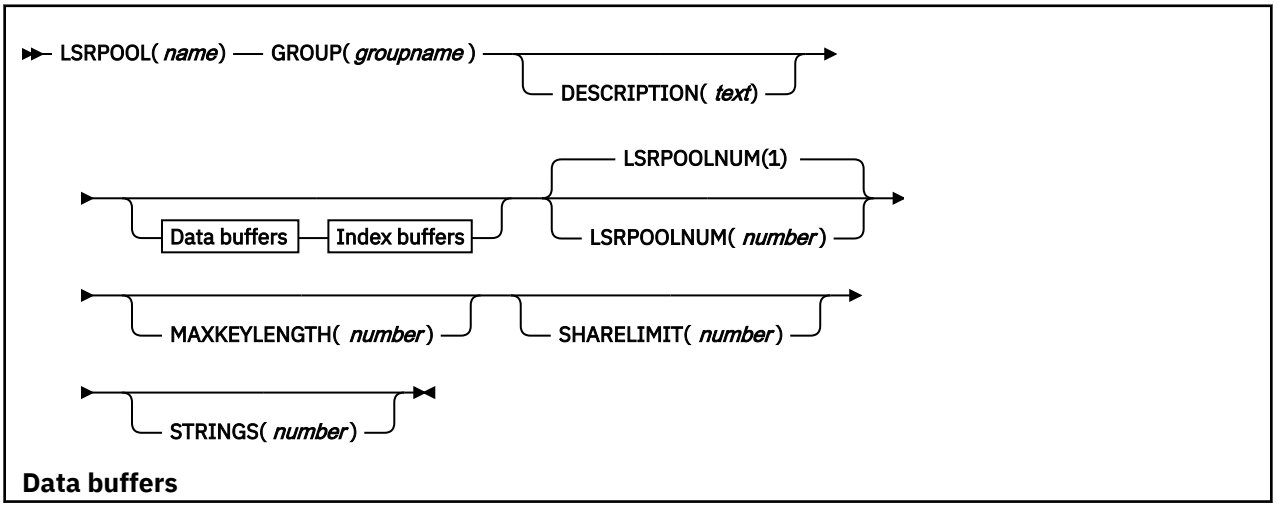
When the LSRPOOL definition is installed in the active system, its information is stored and used when the pool with the specified ID is next built. A pool is built when the first file that uses a particular LSR pool is opened, and is dynamically deallocated only when no files are currently open against that pool. This means that when an LSRPOOL definition is installed into the system it might not take effect immediately.

CICS sets default attributes if an LSRPOOL is not defined, but you are advised to define the LSRPOOL anyway, for reasons of performance. In a production system, for example, delay might be incurred while pool requirements are being calculated by CICS. Another possible problem is that if files are not allocated at the time the pool is built, the data set names are not known to CICS. In this case, the pool is built based on the information available, but the subsequent performance of the system can suffer or files might fail to open.

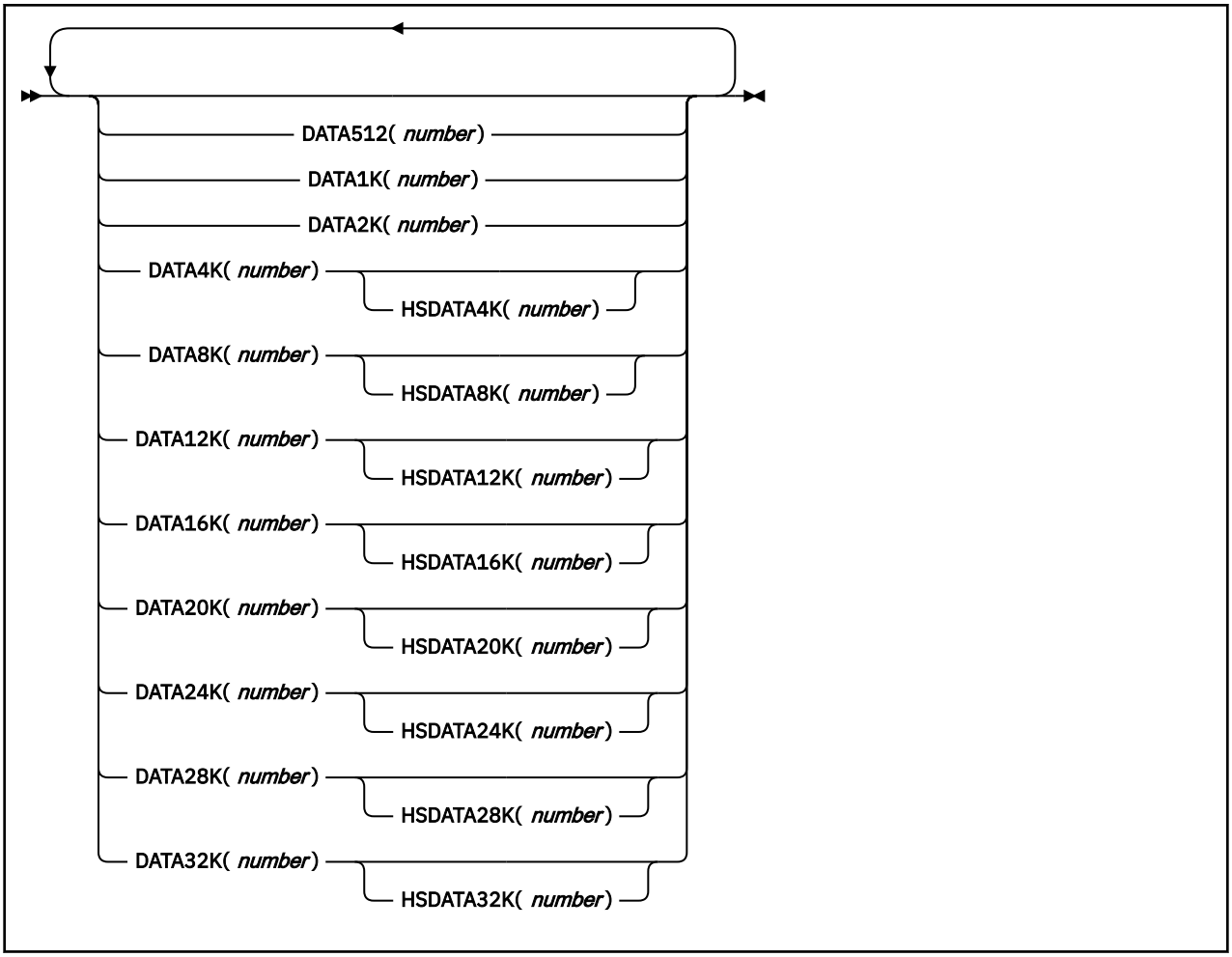
You can associate the CSD file with a particular LSRPOOL by specifying the **CSDLSRNO** system initialization parameter. The default is pool 1; ensure that sufficient buffers of an appropriate size are provided to permit the CSD file to be used by CICS. See Setting up shared data sets, CSD and SYSIN for further information about CSDLSRNO and for details about calculating the buffer requirements for the CSD file.

LSRPOOL attributes

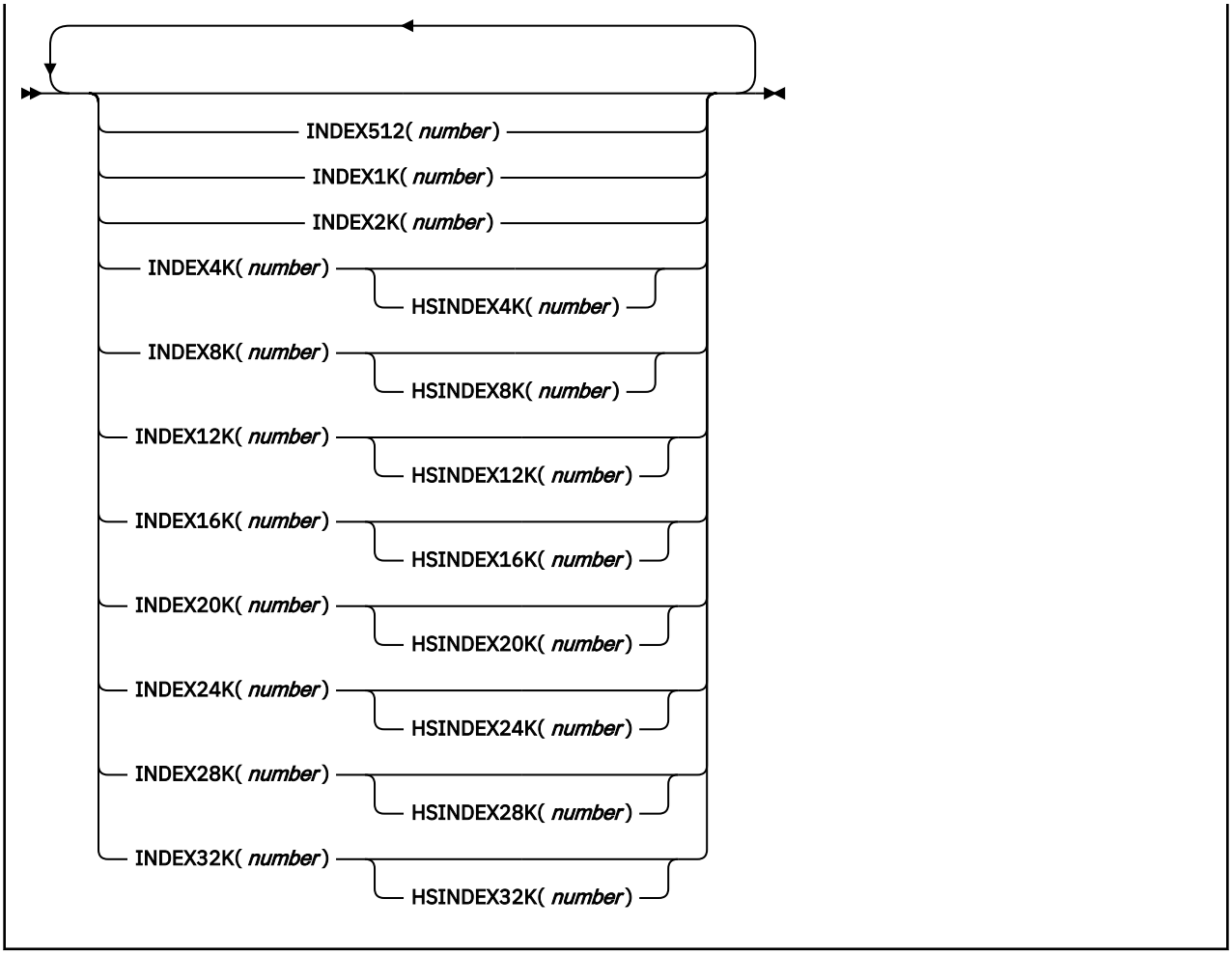
Describes the syntax and attributes of the LSRPOOL resource.



Data buffers



Index buffers



DATA BUFFERS

Specify the number of data buffers of each size that you require, in the range 3 through 32767. These attributes do not have default values.

DATA512(*number*)
Specifies the number, in the range 3 through 32767, of 512 byte data buffers you require.

DATA1K(*number*)
Specifies the number, in the range 3 through 32767, of 1 KB data buffers you require.

DATA2K(*number*)
Specifies the number, in the range 3 through 32767, of 2 KB data buffers you require.

DATA4K(*number*)
Specifies the number, in the range 3 through 32767, of 4 KB data buffers you require.

DATA8K(*number*)
Specifies the number, in the range 3 through 32767, of 8 KB data buffers you require.

DATA12K(*number*)
Specifies the number, in the range 3 through 32767, of 12 KB data buffers you require.

DATA16K(*number*)
Specifies the number, in the range 3 through 32767, of 16 KB data buffers you require.

DATA20K(*number*)
Specifies the number, in the range 3 through 32767, of 20 KB data buffers you require.

DATA24K(*number*)
Specifies the number, in the range 3 through 32767, of 24 KB data buffers you require.

DATA28K(*number*)
Specifies the number, in the range 3 through 32767, of 28 KB data buffers you require.

DATA32K(*number*)
Specifies the number, in the range 3 through 32767, of 32 KB data buffers you require.

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

HIPERSPACE DATA BUFFERS
Specify the number of Hiperspace data buffers of each size that you require, in the range 0 through 16777215. If you leave these fields blank, there are no default values.

Note: If you specify a value for a Hiperspace data buffer of a given size, you must also specify a value for the data buffer of the same size.

HSDATA4K(*number*)
Specifies the number, in the range 0 through 16777215, of 4 KB Hiperspace data buffers you require.

HSDATA8K(*number*)
Specifies the number, in the range 0 through 16777215, of 8 KB Hiperspace data buffers you require.

HSDATA12K(*number*)
Specifies the number, in the range 0 through 16777215, of 12 KB Hiperspace data buffers you require.

HSDATA16K(*number*)
Specifies the number, in the range 0 through 16777215, of 16 KB Hiperspace data buffers you require.

HSDATA20K(*number*)
Specifies the number, in the range 0 through 16777215, of 20 KB Hiperspace data buffers you require.

HSDATA24K(*number*)
Specifies the number, in the range 0 through 16777215, of 24 KB Hiperspace data buffers you require.

HSDATA28K(*number*)
Specifies the number, in the range 0 through 16777215, of 28 KB Hiperspace data buffers you require.

HSDATA32K(*number*)
Specifies the number, in the range 0 through 16777215, of 32 KB Hiperspace data buffers you require.

HIPERSPACE INDEX BUFFERS
Specify the number of Hiperspace index buffers of each size that you require, in the range 0 through 16777215. If you leave these fields blank, there are no default values.

Note: If you specify a value for a Hiperspace index buffer of a given size, you must also specify a value for the index buffer of the same size.

HSINDEX4K(*number*)
Specifies the number, in the range 0 through 16777215, of 4 KB Hiperspace index buffers you require.

HSINDEX8K(*number*)
Specifies the number, in the range 0 through 16777215, of 8 KB Hiperspace index buffers you require.

HSINDEX12K(*number*)
Specifies the number, in the range 0 through 16777215, of 12 KB Hiperspace index buffers you require.

HSINDEX16K(*number*)
Specifies the number, in the range 0 through 16777215, of 16 KB Hiperspace index buffers you require.

HSINDEX20K(*number*)
Specifies the number, in the range 0 through 16777215, of 20 KB Hiperspace index buffers you require.

HSINDEX24K(*number*)
Specifies the number, in the range 0 through 16777215, of 24 KB Hiperspace index buffers you require.

HSINDEX28K(*number*)
Specifies the number, in the range 0 through 16777215, of 28 KB Hiperspace index buffers you require.

HSINDEX32K(*number*)
Specifies the number, in the range 0 through 16777215, of 32 KB Hiperspace index buffers you require.

INDEX BUFFERS
Specify the number of index buffers of each size that you require, in the range 3 through 32767. If you leave these fields blank, there are no default values.

INDEX512(*number*)
Specifies the number, in the range 3 through 32767, of 512 byte index buffers you require.

INDEX1K(*number*)
Specifies the number, in the range 3 through 32767, of 1 KB index buffers you require.

INDEX2K(*number*)
Specifies the number, in the range 3 through 32767, of 2 KB index buffers you require.

INDEX4K(*number*)
Specifies the number, in the range 3 through 32767, of 4 KB index buffers you require.

INDEX8K(*number*)
Specifies the number, in the range 3 through 32767, of 8 KB index buffers you require.

INDEX12K(*number*)
Specifies the number, in the range 3 through 32767, of 12 KB index buffers you require.

INDEX16K(*number*)
Specifies the number, in the range 3 through 32767, of 16 KB index buffers you require.

INDEX20K(*number*)
Specifies the number, in the range 3 through 32767, of 20 KB index buffers you require.

INDEX24K(*number*)
Specifies the number, in the range 3 through 32767, of 24 KB index buffers you require.

INDEX28K(*number*)
Specifies the number, in the range 3 through 32767, of 28 KB index buffers you require.

INDEX32K(*number*)
Specifies the number, in the range 3 through 32767, of 32 KB index buffers you require.

LSRPOOL(*name*)
Specifies the name of the local shared resource pool being defined. The name can be up to 8 characters in length.

If only DATA BUFFERS is specified, one set of buffers is built for the pool to be used for both the index and the data components of a VSAM KSDS data set.

If no data buffers are specified, CICS calculates the buffers required for both data and index components, both components sharing the same set of buffers.

If INDEX BUFFERS is specified, two parts of the pool are built, one for data and the other for index buffers. If you specify INDEX BUFFERS, you must also specify DATA BUFFERS.

LSRPOOLID({1|*lsrpool*})
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS.

The value specified for LSRPOOLID in existing definitions is transferred to the new option LSRPOOLNUM.

LSRPOOLNUM({1|*number*})
Specifies the identifier of the local shared resource pool that is being defined. The value must be in the range 1 through 255.

MAXKEYLENGTH(*number*)
Specifies the maximum key length of any of the files that are to share resources. The value must be in the range 0 through 255. This value overrides part of the CICS resource calculation. If you do not specify it, CICS determines the maximum key length, recalculating it each time the LSR is rebuilt.

SHARELIMIT(*number*)
Specifies, as an integer, the percentage of the maximum amount of VSAM resources to be allocated. The number can be any value from 1 through 100.

Specify SHARELIMIT if CICS is to calculate the maximum amount of resources required by the VSAM files that are to share resources. Because these resources are to be shared, some percentage of this maximum amount of resources must be allocated. If this attribute is omitted, 50 percent of the maximum amount of resources is allocated.

If both the STRINGS and SIZE attributes are specified, SHARELIMIT has no effect.

STRINGS(*number*)
Specifies the limit, in the range 1 through 255, of all the strings of the files in the pool. For more information, see Number of buffers and strings for LSR and NSR pools.

LSR pool resource definitions

LSR pool definitions describe the size and characteristics of local shared resource pools that VSAM uses for certain files.

MAPSET resources

A MAPSET resource defines a BMS map sets.

Each interactive application using a display device can use specific screen layouts, or maps. These are not specified in the program itself. Instead, you use basic mapping support (BMS). This gives greater flexibility and allows the maps to be used by multiple invocations of the same program, or by several different programs. You specify maps, and the fields on them, using the DFHMSD, DFHMDI, and DFHMDF macros. For further guidance on this, see Creating the map.

Instead of using the BMS map definition macros, you can define maps interactively with the Screen Definition Facility II (SDF II) licensed program, program numbers 5665-366 (for MVS) and 5664-307 (for VM). SDF II allows you to paint a screen interactively. You can then generate the screen to get the equivalent of a CICS/BMS map set. The test facilities of SDF II also enable you to see your map in its run-time appearance. For background information, see Screen Definition Facility II Library.

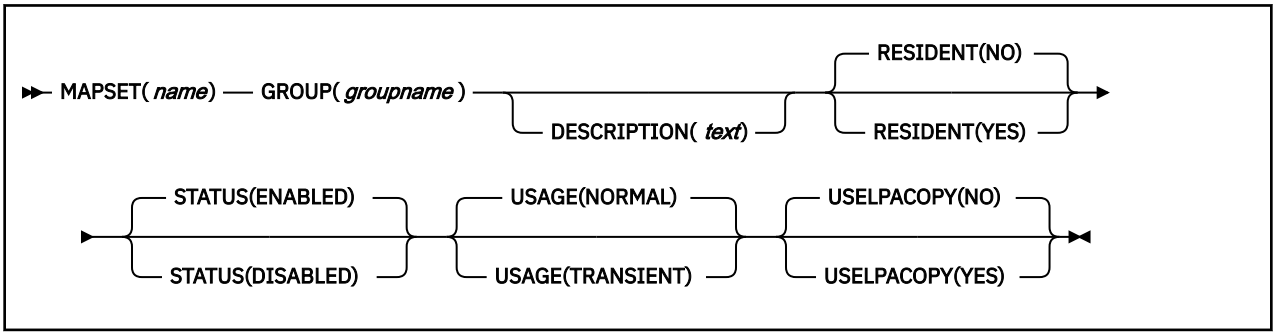
An application can use a series of related maps at different times during the interaction with the user. It can also use several related maps at the same time and on the same display, to build up a complete screen.

These related maps belong to a map set, which you specify in a MAPSET definition. Even if your program has only one map, this must still belong to a map set. You can define your MAPSETs either by using CEDA or DFHCSDUP, or by setting the appropriate system initialization parameters to enable them to be autoinstalled. See Autoinstalling programs, map sets, and partition sets in Configuring for further information about autoinstall.

There is no link through resource definitions between a program and its map sets. Instead, you specify the MAPSET name in the BMS SEND MAP and RECEIVE MAP commands in your program.

MAPSET attributes

Describes the syntax and attributes of the MAPSET resource.



DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

MAPSET(*name*)
specifies the name of this MAPSET definition. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

Do not use map set names beginning with DFH, because these characters are reserved for use by CICS.

For a BMS device-dependent map set, the map set name must be derived by adding the map set suffix to the original (1-to 7-character) map set name. The suffix depends on the parameter specified in the TERM operand of the DFHMSD macroinstruction that defined the map set.

To use device-dependent suffixes, you need to specify BMS=(,,,DDS) as a system initialization parameter. For information on map set suffixes, see Device dependent support.

RESIDENT({NO|YES})
specifies the residence status of the map set.

NO
The map set is not to be permanently resident.

YES
The map set is to be loaded on first reference and is then to be permanently resident in virtual storage, but is to be pageable by the operating system.

RSL
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS. For more information, see Obsolete attributes.

STATUS({ENABLED|DISABLED})
specifies the map set status.

DISABLED
The map set may not be used.

ENABLED
The map set may be used.

USAGE({NORMAL|TRANSIENT})
specifies when the storage for this map set will be released.

NORMAL
When the use count of the map set reaches zero, it will become eligible for removal from storage as part of the normal dynamic storage compression process.

TRANSIENT
When the use count for this map set becomes zero, the storage for this map set is released. This value should be specified for map sets that are referenced infrequently.

USELPACOPY({NO|YES})
specifies whether the map set is to be used from the link pack area (LPA).

NO
The map set is not to be used from the LPA. It is loaded into the CICS partition.

YES
The map set can be used from the LPA if LPA=YES is specified as a system initialization parameter. The use of the map set from the LPA requires that it has been installed there and that the map set is not named by the PRVMOD start-up option. For further guidance on this, see Controlling the use of modules from the MVS link pack area

Map set resource definitions

Map set definitions describe the characteristics of a group of related screen layouts, or maps.

MQCONN resources

An MQCONN resource defines the attributes of the connection between CICS and IBM MQ. You can install or discard an MQCONN resource only when CICS is not connected to IBM MQ.

Only one MQCONN resource can be installed at a time in a CICS region. When you define an MQCONN resource, CICS checks to ensure that there is only one MQCONN resource defined in the group or list. If more than one is found, even one with a different name, a warning message is issued. If you do install a second MQCONN resource, CICS implicitly discards the existing MQCONN resource before proceeding with the installation, unless you are reinstalling an MQCONN resource with the same name.

When you have installed the MQCONN resource, you can use the **CEMT SET MQCONN** or **EXEC CICS SET MQCONN** command, CICSPlex SM, or the CICS Explorer to start the connection between CICS and IBM MQ.

MQCONN and MQMONITOR resources

When you install an MQCONN resource that specifies a value in the **INITQNAME** attribute, CICS also dynamically creates and installs an MQMONITOR resource with the reserved name of DFHMQINI representing the default initiation queue.

DFHMQINI has the following attributes:

QNAME
Specifies the initiation queue name.

TRANSACTION
CKTI (by default)

MONUSERID
This attribute is obtained as follows:

- From the **PLTPIUSR** system initialization parameter, if available
- Otherwise, from the CICS region user ID

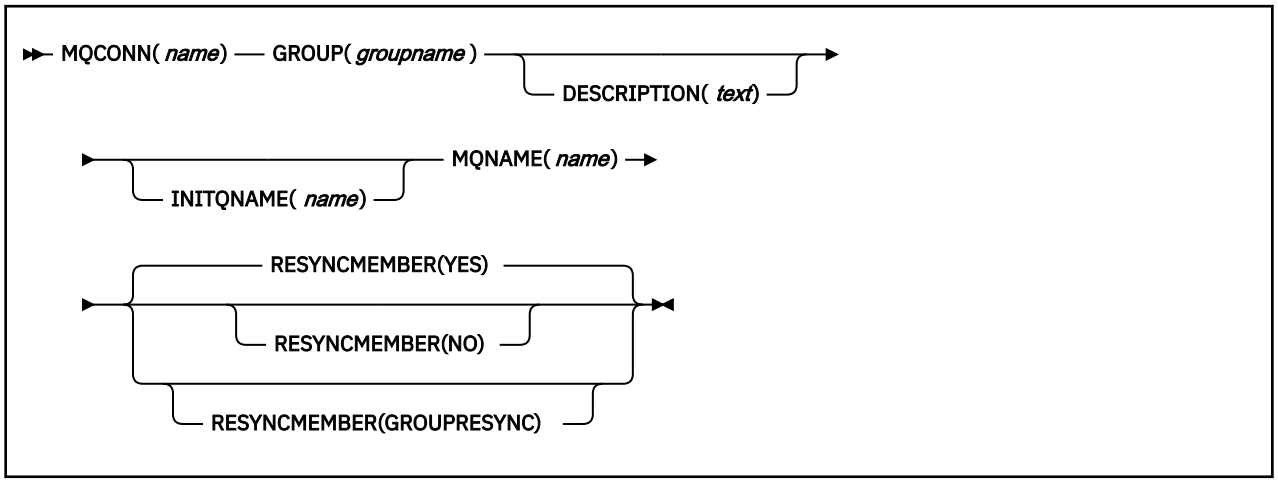
USERID
The value is the CICS default user ID.

DFHMQINI is automatically started when the MQ connection is established, and the user ID associated with the CKTI transaction is obtained from **MONUSERID** if security checking is active, or from the user ID of the transaction that set the state of the MQMONITOR resource to started if security checking is disabled.

You can use the **EXEC CICS INQUIRE MQMONITOR** or **CEMT INQUIRE MQMONITOR** command to inquire on the **QNAME** attribute of the dynamically created MQMONITOR resource. If you want to change this **QNAME** value, you must first change the **INITQNAME** attribute of the MQCONN resource and then reinstall the MQCONN resource. When you discard an MQCONN resource that includes a setting for the **INITQNAME** attribute, the dynamically created MQMONITOR resource and any user-defined MQMONITOR resources are also discarded.

MQCONN attributes

Describes the syntax and attributes of the MQCONN resource.



Attributes

DESCRIPTION(*text*)
In this field, you can provide a description of the resource that you are defining. The description text can be up to 58 characters in length. No restrictions apply to the characters that you can use.

However, if you use parentheses, ensure that each left parenthesis has a matching right one. If you use the CREATE command, for each single apostrophe in the text code, use two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters that you enter are converted to uppercase.

The GROUP name can be up to 8 characters in length.

INITQNAME(*name*)
Specifies the 1 - 48 character name of the default initiation queue for this CICS-MQ connection.

Acceptable characters:

```
A-Z a-z 0-9 . / _ %
```

When you specify the INITQNAME attribute, you can use a symbolic parameter, *&APPLID.*, to identify the APPLID of a CICS region; for example, `&APPLID..EXAMPLE.QUEUE.NAME`. When you install the resource in a CICS region, *&APPLID.* is replaced with the APPLID of the local region.

When you install the MQCONN resource definition, if you specify a non-blank INITQNAME attribute, CICS installs an MQMONITOR resource with the reserved name of DFHMQINI representing the default initiation queue. The QNAME attribute of DFHMQINI specifies the default initiation queue name. If you want to change the QNAME attribute, you must change the INITQNAME attribute of the MQCONN resource definition and then reinstall the MQCONN resource definition.

When you start the CICS-MQ connection by using the **CKQC START** command, if you specify the name of a default initiation queue , the INITQNAME value of the installed MQCONN resource definition and the QNAME value of the installed MQMONITOR resource definition are replaced with the settings that you specify.

MQCONN(*name*)
Specifies the 8-character name of this resource definition.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

MQNAME(*name*)
Specifies the 1 - 4 character name of either a single IBM MQ queue manager, or a queue-sharing group of IBM MQ queue managers.

- If you specify a single queue manager, when you start the CICS-MQ connection, CICS connects only to this queue manager.
- If you specify a queue-sharing group, when you start the connection, CICS connects to any active member of this group on the same LPAR. Use the RESYNCMEMBER attribute to specify what happens when CICS is holding outstanding units of work for the last queue manager to which it connected from the queue-sharing group.
- If you specify RESYNCMEMBER(GROUPRESYNC), MQNAME must be the name of a queue-sharing group.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

The name of the queue manager or queue-sharing group must not start with a numeric character.

IBM MQ uses the @ symbol to pad queue-sharing group names that are less than four characters long. When you specify a queue-sharing group that is less than four characters long, do not include the @ symbols at the end; CICS adds the padding before querying the queue-sharing group. For example, if the queue-sharing group is defined to IBM MQ as DEV@, specify DEV as the value of the MQNAME attribute.

If you specify the name of an alternative IBM MQ queue manager or queue-sharing group on the CEMT or EXEC CICS SET MQCONN command, or specify a queue manager name when you start the CICS-MQ connection by using the CKQC START command, CICS connects to that alternative queue manager or queue-sharing group. In addition, your setting for the MQNAME attribute in the installed MQCONN definition is replaced with the name of the queue manager or queue-sharing group that you specified on the command. If you want to revert to the original queue manager or queue-sharing group, set MQNAME again.

RESYNCMEMBER({YES|NO|GROUPRESYNC})

This attribute applies only if you use the MQNAME attribute to specify an IBM MQ queue-sharing group. RESYNCMEMBER specifies the strategy that CICS adopts if outstanding units of work are being held for the last queue manager to which CICS was connected from the queue-sharing group.

Changing the setting for RESYNCMEMBER must be done only when all resources are in a consistent state; that is, there are no indoubt units of work outstanding otherwise CICS is not able to resynchronize the IBM MQ units of work. It is important to ensure that all resources are in a consistent state before changing RESYNCMEMBER to or from GROUPRESYNC.

YES
CICS connects to the same queue manager, waiting, if necessary, until the queue manager becomes active to resolve the indoubt units of work. YES is the default.

NO
CICS makes one attempt to connect to the same queue manager. If that attempt fails, CICS connects to any member of the queue-sharing group and issues the warning message DFHMQ2064 about the outstanding units of work.

GROUPRESYNC
CICS connects to any member of the queue-sharing group. The queue manager is chosen by IBM MQ and it asks CICS to resolve indoubt units of work on behalf of all eligible queue managers in the queue-sharing group. This function is called *group unit of recovery*. The GROUPRESYNC option can be used only when you are running WebSphere® MQ 7.1, or higher, that supports group unit of recovery for CICS and when group unit of recovery is enabled in the queue managers.

The main use of the GROUPRESYNC option is with shared queues, but it can be used with private queues. Messages on shared queues are resolved immediately, but resolution of private queues occurs when the owning queue manager restarts. If using private queues, it can be convenient to define the queues with QSGDISP(GROUP). QSGDISP(GROUP) ensures that an instance of the private queue exists on every manager in the queue sharing group, so that in the event of failure and reconnection to a different queue manager, the application can continue using the new queue instance. The application would need to be able to tolerate using a new queue instance.

Do not change the setting for RESYNCMEMBER when units of work are outstanding in IBM MQ, because units of work cannot be resolved. A unit of work held in CICS is identified with a resource manager qualifier. When RESYNCMEMBER(GROUPRESYNC) is used the qualifier is the name of the queue-sharing group, otherwise the qualifier used is the name of the individual queue manager.

Units of work that are shunted indoubt are not included in this process, because CICS itself cannot resolve those units of work at this time. Resynchronization for those UOWs occurs when CICS has resynchronized with its remote coordinator.

MQCONN resource definitions

The MQCONN resource definition for the CICS region defines the attributes of the connection between CICS and IBM MQ.

MQMONITOR resources

An MQMONITOR resource defines attributes for IBM MQ message consumers, such as the trigger monitor transaction CKTI. Using an MQMONITOR enables the associated transaction that services an MQ queue to restart automatically when the connection to the IBM MQ queue manager is established.

You can use MQMONITOR resources to process incoming messages on MQ queues including initiation queues. You can have more than one MQMONITOR resource processing incoming messages on an MQ queue.

An installed and started MQMONITOR resource monitors the MQ queue that is either specified in the resource attribute **QNAME** or supplied to the transaction through the **MONDATA** attribute.

When you define an MQMONITOR resource for processing incoming messages on an MQ initiation queue, ensure that the **TRANSACTION** attribute specifies CKTI (the default) or is blank and that the **QNAME** attribute specifies the name of the MQ initiation queue.

When a user-written MQ monitor program, rather than the default CKTI, is to be used with an MQMONITOR resource, it is the responsibility of the user-written program to get messages directly from application queues and perform the required logic. When you code your program, ensure to follow the guidelines described in Developing and using user-written MQ trigger monitors and MQ message consumers.

In order for CICS to attempt to automatically start the transaction associated with an MQMONITOR, the MQMONITOR must be installed with attributes AUTOSTART(YES) and STATUS(ENABLED). When the connection between CICS and MQ has been established, CICS will start MQ monitors that have been installed with attributes AUTOSTART(YES) and STATUS(ENABLED) if the user ID associated with the task that set the MQCONN resource to CONNECTED has sufficient authority to start the associated transactions.

MQMONITORs are automatically stopped when CICS disconnects from the IBM MQ queue manager.

If z/OS Workload Manager Health service is active in the CICS region, MQMONITORs react to the region's health state and might be subject to the setting of the z/OS Workload Manager Health service open status. For details, see Effect of z/OS Workload Manager Health service on MQMONITORs.

Note:

- You can install an MQMONITOR only if you have previously installed an MQCONN. See Defining and installing MQMONITOR resources for instructions.
- When you define an MQMONITOR, CICS checks for consistency with other resource definitions in the same group or list. For more information, see Checks on definitions of MQMONITOR resources.

Security considerations

User ID for the transaction

If security checking is active (that is, the **SEC** system initialization parameter is set to YES), the user ID to be associated with the transaction is obtained from the **MONUSERID** attribute of the MQMONITOR resource.

If security checking is disabled (that is, **SEC** is set to NO), the user ID to be associated with the MQ monitor transaction is the user ID of the transaction that set the state of the MQMONITOR resource to started.

Security checks performed by CICS when setting the MQ monitor state to started

If security checking is active, when setting the MQ monitor state to started, CICS performs security checks on the user ID of the transaction that attempts to set the MQ monitor state to started, as follows:

Note: In the case of setting the MQ monitor state through a CICSPlex SM API interface such as the CICS Explorer, the user ID to be associated with the MQ monitor transaction is either the region user ID or the PLTPIUSR user ID (if specified).

- CICS verifies that the user ID is a surrogate of the user ID defined in **MONUSERID**.
- CICS verifies that the user ID is authorized to start transactions associated with the user ID defined in **MONUSERID**.

DFHMQINI

When you install an MQCONN resource that specifies a value in the **INITQNAME** attribute, CICS also dynamically creates and installs an MQMONITOR resource with the reserved name of DFHMQINI representing the default initiation queue.

DFHMQINI has the following attributes:

QNAME
Specifies the initiation queue name.

TRANSACTION
CKTI (by default)

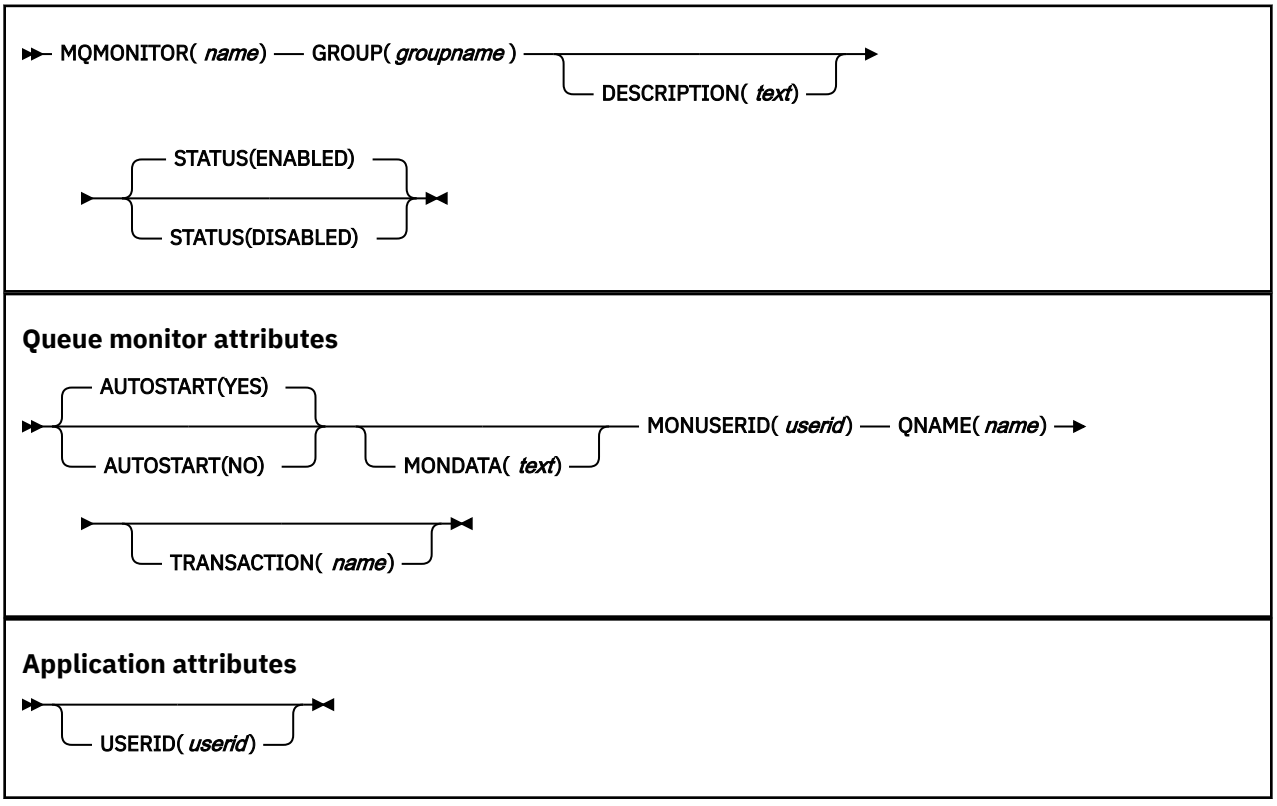
MONUSERID
This attribute is obtained as follows:

- From the **PLTPIUSR** system initialization parameter, if available
- Otherwise, from the CICS region user ID

USERID
The value is the CICS default user ID.

MQMONITOR attributes

Describes the syntax and attributes of the MQMONITOR resource.



Attributes

AUTOSTART({YES|NO})
Specifies whether to start the MQ monitor automatically.

YES
The MQ monitor is started automatically in either of the following situations:

- When the connection to the IBM MQ queue is established.
- If the z/OS Workload Manager (WLM) health service is active (see WLMHEALTH), every increment in the z/OS WLM **HEALTH** value of the CICS region from zero to 100%. For more information, see Effect of z/OS Workload Manager Health service on MQMONITORs and Alert monitor (CKAM).

NO
The MQ monitor is not started automatically.

DESCRIPTION(*text*)
In this field, you can provide a description of the resource that you are defining. The description text can be up to 58 characters in length. No restrictions apply to the characters that you can use. However, if you use parentheses, ensure that each left parenthesis has a matching right one. If you use the CREATE command, for each single apostrophe in the text code, use two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters that you enter are converted to uppercase.

The GROUP name can be up to 8 characters in length.

MQMONITOR(*name*)
Specifies the 8-character name of this resource definition.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

MONDATA(*text*)
You can specify data to be passed to the transaction monitoring the MQ queue. For example, you can use this field to pass parameters to the MQ bridge monitor transaction CKBR. The attribute can contain up to 200 mixed-case characters.

Note:

When displayed and retrieved by the monitoring task, the MONDATA data is prepended with the following 18 bytes:

Byte 1: < (left chevron)
Bytes 2 - 9: *MQMONITOR resource name*
Bytes 10 - 17: *USERID*
Byte 18: > (right chevron)

Bytes 19 - 218 contains MONDATA as entered by the user.

Therefore, user-written programs must allow for a maximum length of 218 bytes to retrieve **MONDATA**, and use the MQMONITOR name as specified in bytes 2 - 9 from the retrieved **MONDATA** for setting

the MONSTATUS attribute of the MQ monitor, thereby indicating its current status. Also note that when security checking is active, CICS performs security checks on the user ID associated with the transaction that attempts to set the MQ monitor state to started. For more information, see the security considerations described in MQMONITOR resources.

Any user-defined character strings `&applid.` or `&APPLID.` will be replaced by the APPLID of the local region when the MQMONITOR is installed. This will facilitate generic use of this resource.

MONUSERID(*userid*)
Specifies the user ID to be associated with the transaction monitoring the MQ queue. This attribute is only effective when security checking is active (that is, the **SEC** system initialization parameter is set to YES). If security checking is disabled (that is, **SEC** is set to NO), the user ID to be associated with the MQ monitor transaction is the user ID of the transaction that set the state of the MQMONITOR resource to started.

Note: If security checking is active, CICS verifies that the user ID associated with the transaction that attempts to set the MQ monitor state to started is a surrogate of the user ID defined in **MONUSERID** and is authorized to start transactions associated with the **MONUSERID**. In the case of setting the MQ monitor state through a CICSPlex SM API interface such as the CICS Explorer, the user ID to be associated with the MQ monitor transaction is either the region user ID or the PLTPIUSR user ID (if specified).

QNAME(*name*)
Specifies the name of the MQ queue to be monitored.

When you specify the **QNAME** attribute, you can use a symbolic parameter, *&APPLID.*, as a prefix in the value to identify the APPLID of a CICS region; for example, `&APPLID..EXAMPLE.QUEUE.NAME`. When you install the resource in a CICS region, *&APPLID.* is replaced with the APPLID of the local region.

If **QNAME** is omitted, the value is set to `&APPLID..INITIATION.QUEUE` by default when the resource is installed.

STATUS({ENABLED|DISABLED})
Specifies the initial status of the MQMONITOR resource when it is installed.

DISABLED
The MQMONITOR resource is not available for use.

ENABLED
The MQMONITOR resource is available for use.

TRANSACTION(*name*)
Specifies the 4-character ID of the CICS transaction that is used by the task monitoring the MQ queue.

If **TRANSACTION** is omitted, the value is set to CKTI by default.

USERID(*id*)
Specifies the user ID to be used by default for starting the transaction associated with the application if no suitable user ID is available from any other source.

If **USERID** is omitted, the CICS default user ID will be used.

MQMONITOR resource definitions

MQMONITOR resource definitions define attributes for WebSphere MQ message consumers, such as the trigger monitor transaction CKTI.

PARTITIONSET resources

A PARTITIONSET resource defines a partition set, that is, is a table that describes to CICS how to partition a display screen.

Partition sets are created by coding and assembling a series of commands.

The screen areas of some display devices (the 8775 Display Terminal and the IBM 3290 Information Panel, for example), can be divided into **partitions**, which can be treated as several different small displays. Different programs or program steps in a transaction can write to or receive input from different partitions.

You specify the partition set with DFHPSD and DFHPDI macros, described for programming purposes in Partition set definition.

You specify each different partition configuration as a PARTITIONSET. PARTITIONSET definitions are created using CEDA or DFHCSDUP, or they can be autoinstalled if the appropriate system initialization parameters have been set. See Autoinstalling programs, map sets, and partition sets in Configuring for information.

You can name the PARTITIONSET that you want the transaction to use in the TRANSACTION definition. When the transaction starts, the information is loaded into the internal buffer of the display device.

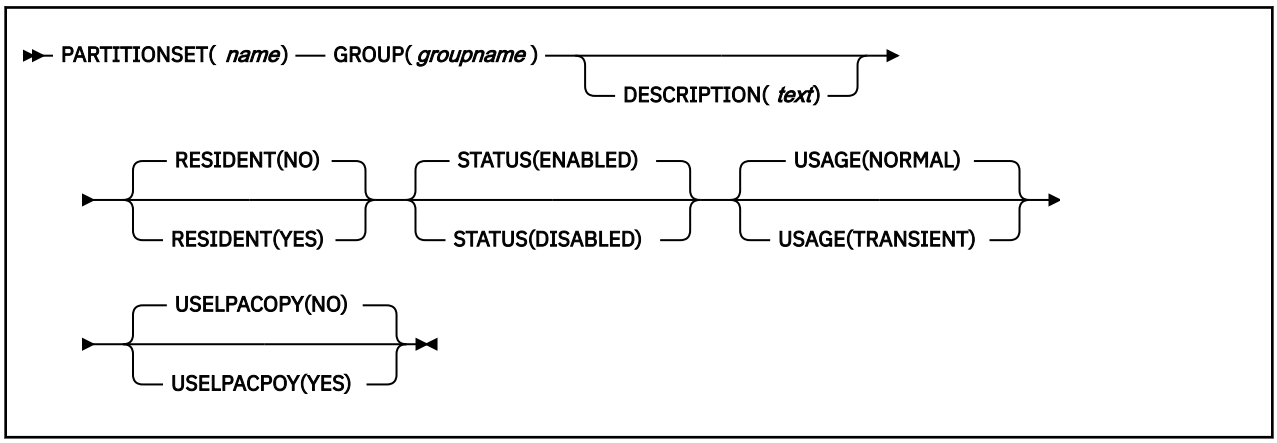
Alternatively, if you do not specify a PARTITIONSET, CICS sets the display device to its base state before the transaction is initiated.

The transaction may require CICS to load a PARTITIONSET, or change to a new one, by issuing the BMS SEND PARTNSET command. This loads the partition set dynamically, if its definition has been installed in the active CICS system.

Instead of using the BMS partition definition macros, you can define partitions interactively with the Screen Definition Facility II (SDF II) licensed program, program numbers 5665-366 (for MVS) and 5664-307 (for VM). SDF II allows you to paint a screen interactively. You can then generate the screen to get the equivalent of a CICS/BMS partitionset. The test facilities of SDF II also enable you to see your partition in its run-time appearance. For general information, see Screen Definition Facility II Library.

PARTITIONSET attributes

Describes the syntax and attributes of the PARTITIONSET resource.



DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

PARTITIONSET(*name*)
specifies the name of this PARTITIONSET definition. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

It is preferable not to use names that start with DFH because these characters are reserved for use by CICS.

CICS can delete or discard them and replace them with CICS resources when you upgrade or provide new function or fixes.

For a device-dependent partition set, the partition set name must be derived by adding the partition set suffix to the original (1- to 6-character) partition set name. The suffix depends on the parameter specified in the SUFFIX operand of the DFHPSD macro instruction that defined the partition set.

To use device-dependent suffixes, you need to specify BMS=(,,DDS) as a system initialization parameter.

RESIDENT({NO|YES})
specifies the residence status of the partition set.

NO
The partition set is not to be permanently resident.

YES
The partition set is to be loaded on first reference and is then to be permanently resident in virtual storage, but is to be pageable by the operating system.

RSL
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS. For more information, see Obsolete attributes.

STATUS({ENABLED|DISABLED})
specifies the partition set status.

DISABLED
The partition set may not be used.

ENABLED
The partition set may be used.

USAGE({NORMAL|TRANSIENT})
specifies when the storage for this partition set is released.

NORMAL
When the use count for this partition set reaches zero, it becomes eligible for removal from storage as part of the normal dynamic program compression process.

TRANSIENT
When the use count for this partition set becomes zero, the storage for this partition set is released. This value should be specified for partition sets that are referenced infrequently.

USELPACOPY({NO|YES})
specifies whether the partition set is to be used from the link pack area (LPA).

NO
The partition set is not to be used from the LPA. It is loaded into the CICS partition.

YES
The partition set can be used from the LPA if LPA=YES is specified as a system initialization parameter. The use of the partition set from the LPA requires that it has been installed there and that the partition set is not named by the PRVMOD start-up option. For more details on this, see Controlling the use of modules from the MVS link pack area.

Partition set resource definitions

Partition set definitions describe the characteristics of a display partition configuration.

PARTNER resources

A PARTNER resource enables CICS application programs to communicate, using APPC protocols, with a partner application program running on a remote logical unit.

The interaction between a CICS application program and a partner application program is called a *conversation*.

The PARTNER definition also facilitates the use of the call to the interface with the communications element of the System Application Architecture (SAA). For more information on the SAA communications interface, see the Common Programming Interface Communications (CPI-C) Reference.

To allow the SAA communications interface to be used, you must specify the following resources:

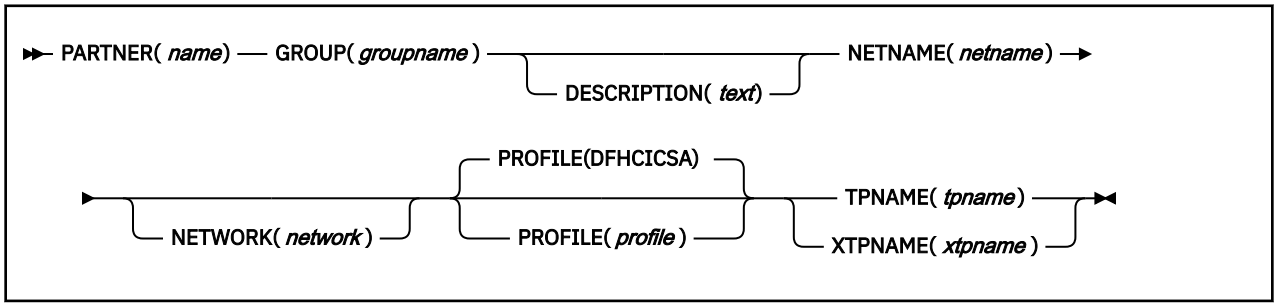
- A PROFILE definition (see "PROFILE resources" on page 114)
- A CONNECTION definition (see "CONNECTION resources" on page 6)
- A SESSIONS definition (see "SESSIONS resources" on page 131)

You can define your CICS partner information in one of two ways:

- Create a PARTNER definition
- In an application program, by setting SYMDESTNAME to a null value, and issuing the appropriate CPI SET calls. See the Common Programming Interface Communications (CPI-C) Reference for further details.

PARTNER attributes

Describes the syntax and attributes of the PARTNER resource.



DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

NETNAME(*netname*)
is the netname of the logical unit on which the partner application program is running. It matches the NETNAME attribute specified in the CONNECTION definition. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

NETWORK(*network*)
You can use this optional attribute to specify the name of the network on which the partner LU is located. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

PARTNER(*name*)
specifies the name of this PARTNER definition. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

It is preferable not to use partner names that start with DFH because these characters are reserved for use by CICS.

CICS can delete or discard them and replace them with CICS resources when you upgrade or provide new function or fixes.

A partner definition specifies the SAA communications interface information required to establish a conversation with a partner program. For further guidance on this, see Common Programming Interface Communications (CPI-C) Reference.

PROFILE(*profile*)
specifies the name of the PROFILE definition which is to be used for the session and conversation. The default PROFILE is DFHCICSA. The name can be up to 8 characters in length.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

TPNAME (*tpname*)
specifies the name of the remote transaction program that will be running on the partner LU. The definition of a remote TP name is mandatory; you must specify either TPNAME or its alternative, XTPNAME. This name can be up to 64 characters in length.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

If this range of characters is not sufficient for a name that you want to specify, you may use the XTPNAME attribute instead of TPNAME.

XTPNAME (*xtpname*)
This attribute may be used as an alternative to TPNAME; you **must** specify one of the two, because the definition of a remote TP name is mandatory.

Enter a hexadecimal string up to 128 characters in length, representing the name of the remote transaction program that runs on the partner LU. All hexadecimal combinations are acceptable **except X'40'**.

To specify an XTPNAME more than 72 characters long to DFHCSDUP, put an asterisk in column 72. This causes the following line to be concatenated.

Installing partner definitions

When you install a PARTNER definition, CICS attempts to resolve references to CONNECTION and PROFILE definitions.

About this task

CICS then creates an entry for the PARTNER definition in the *partner resource table* (PRT). See Defining remote resources for DTP for more information.

Because it is possible for programs to use the SAA communications interface SET calls to change the PARTNER name and the TPNAME name, CICS does not check that the CONNECTION definition is present when the PARTNER resource is being installed.

If you leave out the PROFILE attribute, CICS uses the default profile DFHCICSA. Because it is not possible for an SAA communications interface program to set a different profile, the PROFILE must be installed before the PARTNER resource. However, the PARTNER install does not fail if the PROFILE is missing; instead, a run-time error occurs when the partner conversation is attempted.

The MODENAME specified in the PROFILE need not be specified in a corresponding SESSIONS definition, as it is possible for the SAA communications interface program to set a different value for MODENAME.

CICS programs, and SAA communications interface programs that do not use the SET calls, require that all the relevant definitions be installed. You can use CEDA CHECK to help find out if all required definitions are in a group.

Partner resource definitions

Partner definitions enable CICS application programs to communicate with a partner application program running on a remote region using APPC protocols.

PIPELINE resources

A PIPELINE resource is used when a CICS application is in the role of a web service provider or requester. It provides information about the message handler programs that act on a service request and on the response. Typically, a single PIPELINE definition defines an infrastructure that can be used by many applications.

The information about the processing nodes is supplied indirectly: the PIPELINE specifies the name of a z/OS UNIX file containing an XML description of the nodes and their configuration.

An inbound web service request (that is, a request by which a client invokes a web service in CICS) is associated with a PIPELINE resource by the URIMAP resource. The URIMAP identifies the PIPELINE resource that applies to the URI associated with the request; the PIPELINE specifies the processing that is to be performed on the message.

For service providers deployed using the CICS web services assistant, URIMAP resources are created automatically when the pickup directory is scanned. This scan occurs when the PIPELINE resource is installed or as a result of a **PERFORM PIPELINE SCAN** command. The URIMAP resource that provides CICS with the information to associate the WEBSERVICE resource with a specific URI is a required resource. The attributes for this resource are specified by a web service binding file in the pickup directory. The URIMAP resource that provides CICS with the information to associate the WSDL archive file or WSDL document with a specific URI is an optional resource and is created if either a WSDL file or WSDL archive file are present in the pickup directory.

For service requesters, CICS does not create any URIMAP resources automatically when the PIPELINE resource is installed or as a result of a **PERFORM PIPELINE SCAN** command.

PIPELINE resources in CICS bundles

You can use a CICS bundle to create, edit, and install a PIPELINE resource definition. If you create a PIPELINE resource in this way, you must use the CICS bundle to manage the lifecycle of that resource, and you cannot manage the resource independently.

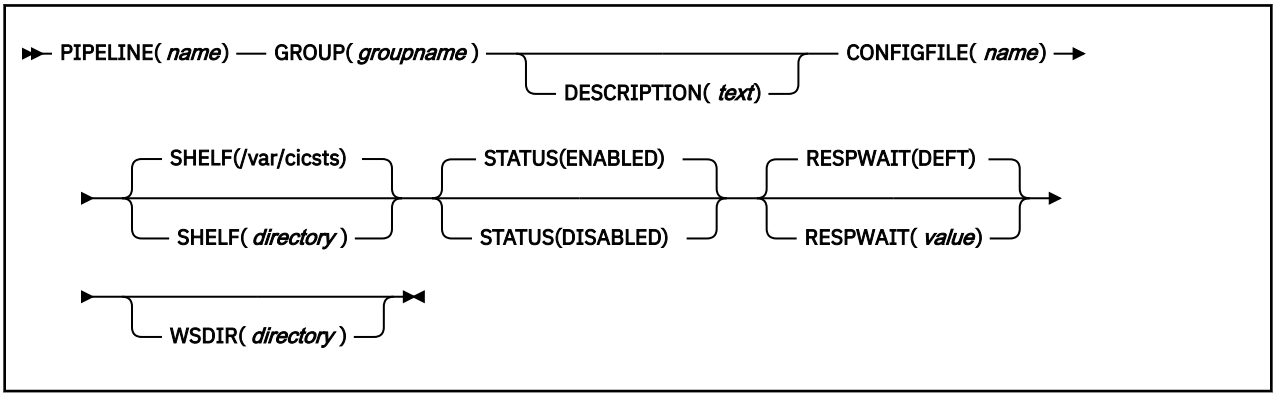
You can inquire on a PIPELINE resource that is dynamically generated by a CICS bundle, by using the **EXEC CICS INQUIRE PIPELINE** or **CEMT INQUIRE PIPELINE** command. However, you cannot issue a **DISCARD** command against a dynamically generated PIPELINE resource, or use a **SET** command to change the status of the resource. You must issue the commands against the BUNDLE resource, and CICS applies them to the PIPELINE resource.

You can modify the attributes of a dynamically generated PIPELINE resource, but the changes are not cataloged and will not be recovered across a warm restart of CICS. If you want to change an attribute of a PIPELINE resource that was installed by a bundle, disable and discard the CICS bundle, and install a new version of the bundle with the required changes.

For more information about defining resources in CICS bundles, see Defining CICS bundles.

PIPELINE attributes

Describes the syntax and attributes of the PIPELINE resource.



CONFIGFILE(*name*)
Specifies the fully qualified or relative name of a z/OS UNIX file that contains information about the processing nodes that will act on a service request, and on the response.

- For a configuration file for a PIPELINE resource defined by online resource definition, the file path is either fully qualified, if prefixed with a forward slash (/), or is relative to the HOME directory of the CICS region user ID.
- For a configuration file for a PIPELINE resource defined in a CICS bundle, the file path is relative to the root directory of the CICS bundle. The zFS file must be packaged in the CICS bundle with the PIPELINE resource. For more information, see Referencing zFS artifacts in a bundle.

The value specified must be a valid name for a UNIX file:

- It must not contain embedded space characters.
- It must not contain consecutive instances of the / character.
- It is case-sensitive.

Acceptable characters:

```
A-Z a-z 0-9 . / _ # @ -
```

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

PIPELINE(*name*)
Specifies the name of this PIPELINE. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

RESPWAIT(*value*)
Specifies the number of seconds that an application program should wait for a response message from a remote web service. The value can range from 0 to 9999 seconds.

To use the default timeout value of the transport protocol, specify DEFT. The default timeout value is also used if you do not specify any value for RESPWAIT.

- The default timeout value for HTTP is 10 seconds.
- The default timeout value for WebSphere MQ is 60 seconds.

SHELF({/var/cicsts/|*directory*})
Specifies the 1–255 character fully-qualified name of a directory (a *shelf*, primarily for web service binding files) on z/OS UNIX.

The value specified must be a valid name for a UNIX file:

- It must not contain embedded space characters.
- It must not contain consecutive instances of the / character.
- It is case-sensitive.

Acceptable characters:

```
A-Z a-z 0-9 . / _ # @ -
```

CICS regions into which the PIPELINE definition is installed must have full permissions to the shelf directory—read, write, and the ability to create subdirectories.

A single shelf directory can be shared by multiple CICS regions and by multiple PIPELINE definitions. Within a shelf directory, each CICS region uses a separate subdirectory to keep its files separate from those of other CICS regions. Within each region's directory, each PIPELINE uses a separate subdirectory.

You should not attempt to modify the contents of a shelf that is referred to by an installed PIPELINE definition. If you do, the effects are unpredictable.

A PIPELINE resource that is defined in a CICS bundle using CICS Explorer does not use a shelf directory.

STATUS({ENABLED|DISABLED})
Specifies the initial status of the PIPELINE when it is installed:

ENABLED
Web service requests for this PIPELINE are processed normally.

DISABLED
Web service requests for this PIPELINE cannot be processed.

WSDIR(*directory*)
specifies the 1–255 character fully-qualified name of the *web service binding directory* (also known as the *pickup directory*) on z/OS UNIX. Each PIPELINE installed in a CICS region must specify a different web service binding directory.

The value specified must be a valid name for a UNIX file:

- It must not contain embedded space characters.

- It must not contain consecutive instances of the / character.
- It is case-sensitive.

Acceptable characters:

```
A-Z a-z 0-9 . / _ # @ -
```

The web service binding directory contains web service binding files that are associated with a PIPELINE, and that are to be installed automatically by the CICS scanning mechanism. When the PIPELINE definition is installed, CICS scans the directory and automatically installs any web service binding files it finds there. Note that this happens regardless of whether the PIPELINE is installed in enabled or disabled state.

If you specify a value for the WSDIR attribute, it must refer to a valid z/OS UNIX directory to which the CICS region has at least read access. If this is not the case, any attempt to install the PIPELINE resource will fail.

If you do not specify a value for WSDIR, no automatic scan takes place on installation of the PIPELINE.

Pipeline resource definitions

A PIPELINE resource definition is used when a CICS application is acting as a web service provider or requester. It provides information about the message handler programs that act on a web service request and on the response.

PROCESSTYPE resources

A PROCESSTYPE resource defines a BTS process-type. It names the CICS file which relates to the physical VSAM data set (repository) on which details of all processes of this type (and their activity instances) are to be stored.

Using the CICS business transaction services (BTS) API, you can define and execute complex business applications called *processes*. A process is represented in memory as a block of storage containing information relevant to its execution. It also has associated with it at least one additional block of information called an activity instance. When not executing under the control of the CICS business transaction services domain, a process and its activity instances are written to a data set called a *repository*.

You can categorize your BTS processes by assigning them to different process-types. This is useful, for example, for browsing purposes. The activities that constitute a process are of the same process-type as the process itself.

Note: Records for multiple process-types can be written to the same repository data set.

Figure 2 on page 112 shows the relationship between PROCESSTYPE definitions, FILE definitions, and BTS repository data sets:

- More than one PROCESSTYPE resources can refer to the same FILE.
- More than one FILE can refer to the same BTS repository data set.

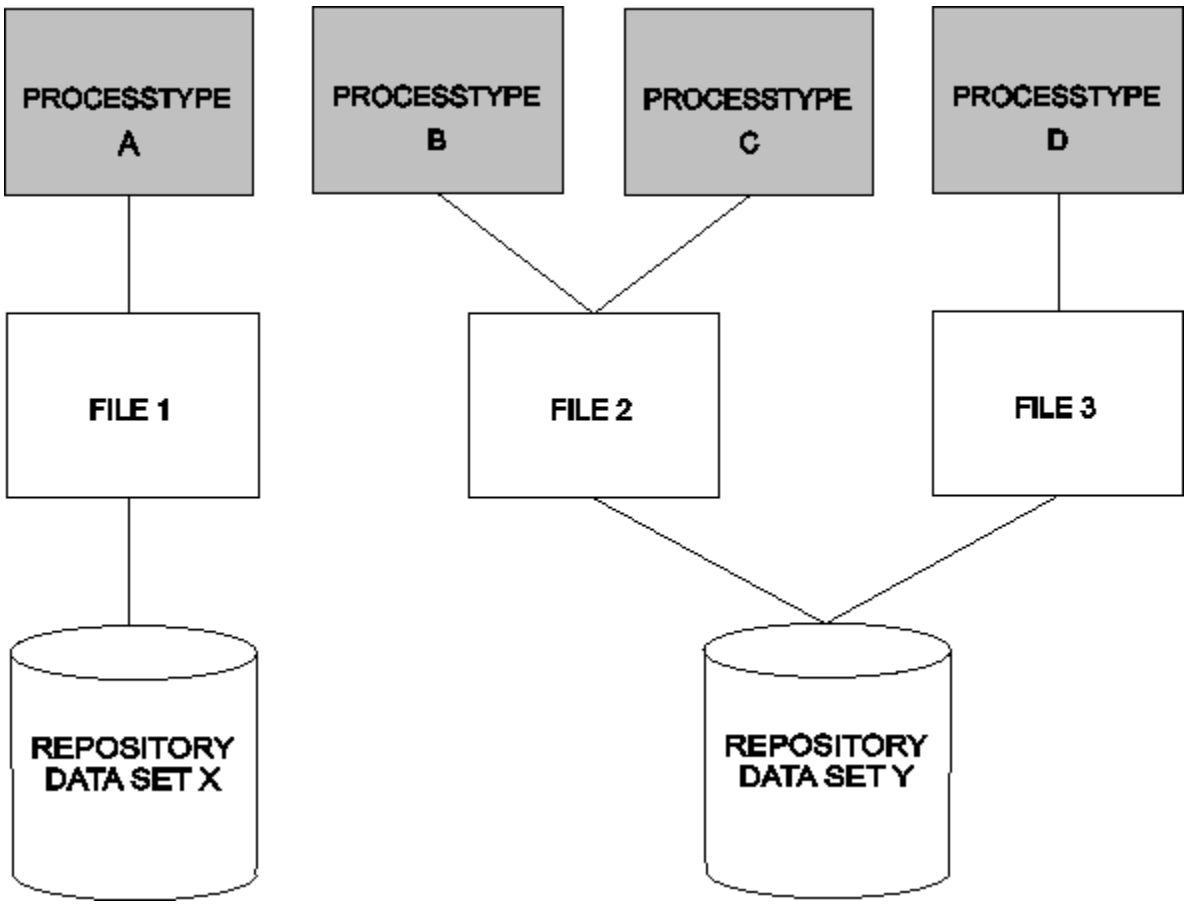
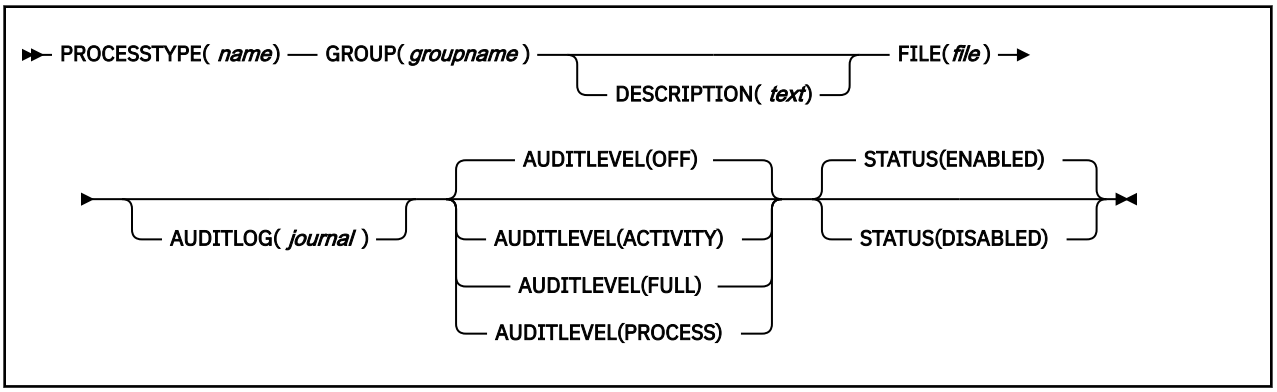


Figure 2. PROCESSTYPE definitions, FILE definitions, and BTS repository data sets

You may want to record the progress of BTS processes and activities for audit purposes, and to help diagnose errors in BTS applications. If so, you can name the CICS journal to which audit records are to be written, and the level of auditing that is required, for processes of the specified type.

PROCESSTYPE attributes

Describes the syntax and attributes of the PROCESSTYPE resource.



AUDITLEVEL({OFF|PROCESS|ACTIVITY|FULL})
specifies the initial level of audit logging for processes of this type. If you specify any value other than OFF, you must also specify the AUDITLOG option.

ACTIVITY
Activity-level auditing. Audit records will be written from:

1. The process audit points

2. The activity primary audit points.

FULL
Full auditing. Audit records will be written from:

1. The process audit points
2. The activity primary *and* secondary audit points.

OFF
No audit trail records will be written.

PROCESS
Process-level auditing. Audit records will be written from the process audit points only.

For details of the records that are written from the process, activity primary, and activity secondary audit points, see Specifying the level of audit logging

AUDITLOG(*journal*)
specifies the name of a CICS journal to which audit trail records will be written, for processes of this type and their constituent activities. The name can be up to eight characters long. If you do not specify an audit log, no audit records will be kept for processes of this type.

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

FILE(*file*)
specifies the name of the CICS file definition that will be used to write the process and activity records of this process-type to its associated repository data set. The name can be up to eight characters long.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

PROCESSTYPE(*name*)
specifies the name of this PROCESSTYPE definition.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

The name can be up to eight characters in length.. Leading and embedded blank characters are not permitted. If the name supplied is less than eight characters, it is padded with trailing blanks up to eight characters.

STATUS({ENABLED|DISABLED})
specifies the initial status of the process-type following a CICS initialization with START=COLD or START=INITIAL. After initialization, you can use the CEMT SET PROCESSTYPE command to change the status of the process-type. The status of the process-type following a restart is recovered to its status at the previous shutdown.

DISABLED
Processes of this type cannot be created. An EXEC CICS DEFINE PROCESS request that tries to create a process of this type results in the INVREQ condition being returned to the application program.

ENABLED
Processes of this type can be created.

PROCESSTYPE definitions

PROCESSTYPE resource definitions describe the physical and operational characteristics of CICS business transaction services (BTS) process types.

PROFILE resources

A PROFILE resource specifies options that control the interactions between transactions and terminals or logical units. The PROFILE is a means of standardizing the use of such options as screen size and printer compatibility, and the use of such functions as message journaling and the node error program.

MODENAME
A profile is associated with the communication between a transaction and an LUTYPE6.1 or APPC session to another system. For APPC sessions, you refer on the PROFILE definition to the MODENAME that is also named on the SESSIONS definitions. This MODENAME is the name of the mode set to which the sessions belong. See "CONNECTION resources" on page 6.

When installed in CICS, the information from the PROFILE definition creates an entry in the profile table. This entry is later used by each transaction that references that PROFILE.

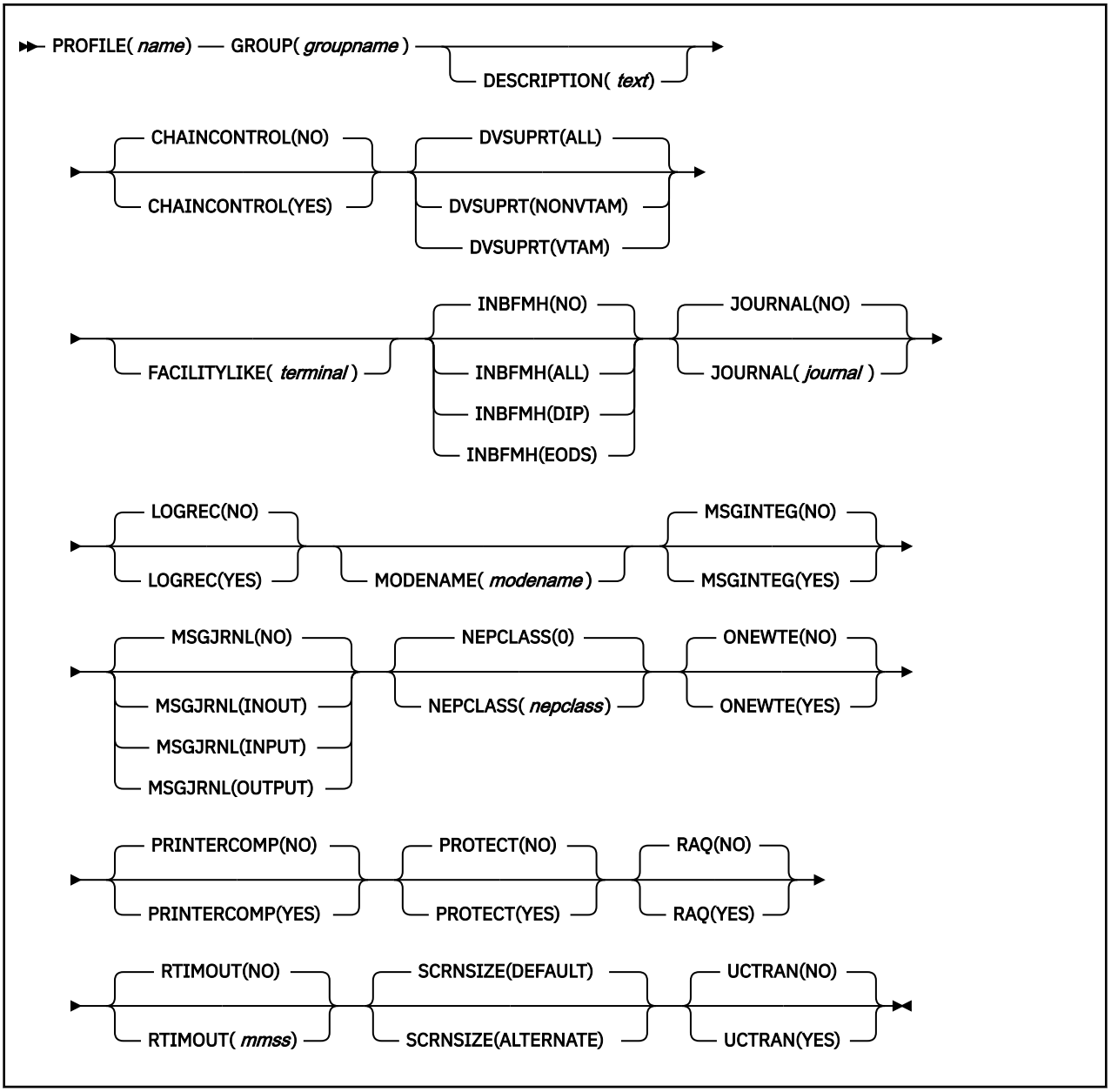
There are CICS-supplied PROFILE definitions suitable for most purposes. Each TRANSACTION definition names the PROFILE to be used. If you do not specify a PROFILE, the transaction uses the PROFILE supplied for using a terminal in a standard way.

With CICS intercommunication facilities (for example, function shipping), a PROFILE is needed for the communication between the transaction and the session. The attributes of the CICS-supplied profiles are shown in "PROFILE definitions in group DFHISC" on page 448. Defining communication profiles gives further information about the CICS-supplied PROFILEs, and tells you about defining your own profiles.

PROFILE attributes

Describes the syntax and attributes of the PROFILE resource.

Note: VTAM is now z/OS Communications Server.



Attributes

CHAINCONTROL({NO|YES})
specifies whether the application program can control the outbound chaining of request units. If you specify CHAINCONTROL(YES), ONEWTE(YES) means one chain and not one terminal control output request.

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

DVSUPRT({ALL|NONVTAM|VTAM})
specifies the devices (terminals or logical units) that are to be supported. The access method used by a particular terminal or logical unit is specified in its associated TCTTE.

ALL
The profile can be used with any terminal or logical unit.

NONVTAM
The profile can be used only with non-z/OS Communications Server terminals.

VTAM
The profile can be used only with logical units.

FACILITYLIKE(*terminal*)
This is an optional attribute that specifies the name of an existing (four-character) terminal resource definition to be used as a template for the bridge facility. It can be overridden by specifying FACILITYLIKE in the bridge exit.

There is no default value for this attribute.

If you are running in a CICS system started with the VTAM=NO system initialization (SIT) parameter, the resource definition specified by FACILITYLIKE must be defined as a remote terminal.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

INBFMH({NO|ALL|DIP|EODS}) (SNA LUs only)
specifies, for profiles used with logical units, whether a function management header (FMH) received from a logical unit is to be passed to the application program.

ALL
All FMHs (except APPC FMHs and LU6.1 ATTACH and SYNCPOINT FMHs that are processed by CICS) are passed to the application program. This value is required for function shipping transactions such as CSMI, transactions which use distributed transaction processing, and for distributed program link requests.

DIP
The batch data interchange program (DFHDIP) is to process inbound FMHs. BMS issues a batch data interchange receive request if a BMS receive request has been issued, and a batch data interchange receive request is issued instead of a terminal control receive request.

EODS
An FMH is passed to the application program only if it indicates end of data set (EODS).

NO
The FMHs are discarded.

JOURNAL({NO|*journal*})
specifies that you want automatic journaling of messages to take place, by giving the identifier of the journal.

NO
No automatic journaling of messages is to take place.

journal
The journal identification to be used for automatic journaling. This can be any number in the range 01 through 99. This number is appended to the letters DFHJ to give a journal identification of the form DFHJ*nn* and this maps to an MVS system logger general log stream.

Note: In CICS Transaction Server for z/OS, DFHJ01 is not the system log.

In a transaction routing environment, message journaling is performed in the application-owning region (AOR). Therefore, you should specify the JOURNAL attribute on the transaction profile in the AOR.

LOGREC({NO|YES})
specifies whether the design of the application requires that each EXEC CICS RECEIVE request is to be satisfied by a logical record. This option allows existing 2770-and 2780-based application programs to be attached to a batch logical unit (for example, 3790 or 8100) without modification to the program.

MODENAME(*modename*)
specifies the name that identifies a group of sessions for use on an APPC connection. The name can be up to eight characters in length, and must be the name of a z/OS Communications Server LOGMODE entry defined to z/OS Communications Server. It must not be the reserved name SNASVCMG. If you omit the modename, it defaults to blanks. See Defining groups of APPC sessions for more information about z/OS Communications Server modenames.

If a transaction that specifies this profile has been started using an EXEC CICS START command, the MODENAME is used for allocation of the principal facility. If a transaction performs an EXEC CICS ALLOCATE command specifying this profile, the MODENAME is used for allocation of the alternate facility.

If you do not specify a MODENAME, CICS selects a session from any one of the mode sets that have been defined.

The CICS-supplied profile DFHCICSA is used, if PROFILE is not specified on an EXEC CICS ALLOCATE command. For function shipping, the profile DFHCICSF is always used. MODENAME is not specified on the definition for either of these profiles, but you can add a MODENAME if you make your own copy. You must then ensure that the mode sets using your MODENAME have been defined in the TERMINAL or SESSIONS definition for all the systems with which communication takes place using APPC.

If a MODENAME is specified and you want to remove it, delete completely the value previously specified by pressing the ERASE EOF key.

MSGINTEG({NO|YES}) (SNA LUs only)
specifies whether a definite response is to be requested with an output request to a logical unit. You cannot specify YES for a pipeline transaction.

MSGJRNL({NO|INPUT|OUTPUT|INOUT})
specifies which messages are to be automatically journaled. If you specify a value other than NO, you must also supply a value for the JOURNAL attribute.

NO
No message journaling is required.

INPUT
Journaling is required for input messages.

OUTPUT
Journaling is to be performed for output messages.

INOUT
Journaling is to be performed for input and output messages.

In a transaction routing environment, message journaling is performed in the application-owning region (AOR) for routed APPC (LU type 6.2) sessions, and you should specify the MSGJRNL attribute on the transaction profile in the AOR. For other routed sessions, message journaling is performed in the terminal-owning region (TOR). In this case, you should specify the MSGJRNL attribute on the transaction profile in the TOR.

NEPCLASS({0|*nepclass*}) (z/OS Communications Server only)
specifies the node error program transaction class. This value overrides the value specified on the TYPETERM and SESSION definitions.

0
This results in a link to the default node error program module for z/OS Communications Server devices, or is the default value for non-z/OS Communications Server devices.

value
The transaction class for the (nondefault) node error program module. The value can be in the range 1 through 255. For programming information on the node error program, see Writing a node error program.

The NEPCLASS attribute applies only to user transactions, and is ignored for SNASVCMGR sessions.

ONEWTE({NO|YES})
specifies whether the transaction is permitted only one write operation or EXEC CICS SEND during its execution. YES has the effect of forcing the LAST option on the first write of the transaction. Any additional write requests are treated as errors, and the task is made ready for abnormal termination.

You must specify YES for a PIPELINE transaction.

PRINTERCOMP({NO|YES})
specifies the level of compatibility required for the generation of data streams to support the printer compatibility option for the BMS SEND TEXT command.

NO
Each line of output starts with a blank character, so that the format is equivalent to that on a 3270 display where an attribute byte precedes each line.

YES
No blank character is inserted, so that forms-feed characters included as the first character of your data are honored and the full width of the printer is available for your data.

If you use the BMS forms feed option, specify YES.

PROFILE(*name*)
specifies the name of this PROFILE definition. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

It is preferable not to use names that start with DFH because these characters are reserved for use by CICS.

CICS can delete or discard them and replace them with CICS resources when you upgrade or provide new function or fixes.

Note: If you use a comma (,) in a name, you will be unable to use those commands such as

```
CEMT INQUIRE PROFILE(value1,value2)
```

where the comma serves as a list delimiter. See List of resource identifiers for information about using lists of resource identifiers.

A profile specifies the options that control the interaction between CICS and a terminal or logical unit. A profile name is specified on the transaction definition to indicate the set of options that control the communication between the transaction and its principal terminal. You can also specify a profile name on an EXEC CICS ALLOCATE command to indicate the options that control communication between the transaction and the allocated session.

CICS supplies a number of profile definitions that are suitable for most purposes. For more information, see Defining communication profiles.

PROTECT
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS. For more information, see Obsolete attributes.

RAQ({NO|YES}) (SNA terminals only)
specifies whether the 'read ahead queuing' option is required.

NO
The transaction obeys SNA protocols and only SEND and RECEIVE when in the correct mode. If it does not follow the protocol, it may be abended with code ATCV.

YES
The transaction may not obey SNA protocols, and CICS queues incoming data on temporary storage until the data is specifically requested by the transaction. RAQ(YES) is provided only for compatibility with transactions that support both bisynchronous devices and logical units, and its use is not recommended.

RTIMOUT({NO|*mmss*})
Specifies the timeout value for the read timeout feature. For tasks that are making HTTP client requests using the EXEC CICS WEB API, when this interval has ended, CICS returns a TIMEDOUT response to the application. For other tasks, either the failed command returns a TERMERR or the task that is timed out receives an AKCT , AZCT, or AZIG abend.

NO
The read timeout feature is not required.

value
This is an interval (MMSS for minutes and seconds) after which the task is terminated if no input has been received from the terminal. The maximum value that can be specified is 70 minutes. If a value is specified and you want to let it default to NO, you must completely delete the value previously specified.

SCRNSIZE({DEFAULT|ALTERNATE})
specifies whether the DEFAULT or ALTERNATE buffer size for a 3270 display or printer is to be used. For further information on the choice of screen sizes and buffer sizes, refer to the ALTSCREEN and DEFSCREEN attributes on the TYPETERM definition.

The SCRNSIZE value is ignored if the TYPETERM definition has ALTSCREEN(0,0) and DEFSCREEN(0,0). That is, the screen size is assumed from the related TERMMODEL attribute in the TYPETERM definition; the page size is taken from PAGESIZE, and the ALTPAGE value is ignored. The 3270 erase write (EW) command is inserted for output requests with the ERASE option.

ALTERNATE
If the TYPETERM definition has nonzero ALTSCREEN, the alternate screen size mode is applied, using the erase write alternate (EWA) command. That is, whenever a terminal output request with the ERASE option is issued, the 3270 EWA command is inserted in the data stream. The ALTSCREEN value is assumed as the screen size, and BMS uses the value in ALTPAGE as the page size.

SCRNSIZE(ALTERNATE) may be used for all CICS service transactions (for example, CSMT).

DEFAULT
If the TYPETERM definition has nonzero ALTSCREEN or nonzero DEFSCREEN, the default screen size mode is applied, using the erase write (EW) command. That is, whenever the terminal issues a terminal output request with the ERASE option, the 3270 EW command is inserted in the data stream. The screen size specified in the DEFSCREEN attribute is assumed, and BMS uses the value specified in the PAGESIZE attribute as the page size.

Note: Both DEFAULT and ALTERNATE can be overridden by the DEFAULT and ALTERNATE options on the SEND MAP, SEND TEXT, and SEND CONTROL commands.

UCTRAN({NO|YES}) (z/OS Communications Server only)
specifies whether terminal input is to be translated to uppercase before passing to programs for the transaction using this profile.

You can also request translation to uppercase at the terminal level on the associated TYPETERM definition (see "TYPETERM attributes" on page 230) but be aware of the following points:

- A TYPETERM UCTRAN(YES) definition overrides a PROFILE UCTRAN(NO) definition. So, if you specify TYPETERM UCTRAN(YES), a PROFILE UCTRAN(NO) has no effect.
- A PROFILE UCTRAN(YES) definition overrides a TYPETERM UCTRAN(NO) definition.

- Specifying TYPETERM UCTRAN(TRANID) causes the tranid to be translated to uppercase so that CICS can locate the transaction definition. All other input received by the application is translated according to what is specified for PROFILE UCTRAN.
- UCTRAN(YES) on a profile definition does not cause translation of the input data until an EXEC CICS RECEIVE or CONVERSE is executed. This means that if the transaction is routed through a dynamic routing program, for example DFHDYP, the copy of the input data passed to the routing program is unaffected by the UCTRAN option of the PROFILE definition.

Note: In a transaction routing environment where your z/OS Communications Server terminals have a remote definition on the AOR, and the AOR has a different UCTRAN value from the TOR, the TOR value of UCTRANST (as specified in an EXEC CICS SET TERMINAL command) overrides that on the AOR.

Table 2 on page 120 shows which portion of the terminal input is translated (transaction id and/or data) according to the setting of the UCTRAN on the PROFILE and TYPETERM resource definitions.

Table 2. The effect of UCTRAN attributes on tranid and data translation

UCTRAN in PROFILE	UCTRAN in TYPETERM	TRANID translated?	Data Translated?
YES	YES	Yes	Yes
YES	NO	No	Yes
YES	TRANID	Yes	Yes
NO	YES	Yes	Yes
NO	NO	No	No
NO	TRANID	Yes	No

Some national-language characters are not automatically translated when UCTRAN(YES) is specified. If that is the case, you can use one of the methods described in Translating national characters to uppercase.

Profile resource definitions

Profile definitions control the interactions between transactions and terminals or logical units.

PROGRAM resources

PROGRAM resources control information for a program that is stored in the program library and used to process a transaction, or part of a transaction.

You can create PROGRAM definitions by using CEDA or DFHCSDUP, or by setting the appropriate system initialization parameters and allowing programs to be autoinstalled. See Autoinstalling programs, map sets, and partition sets in Configuring for information about autoinstall for programs.

PROGRAM resources in CICS bundles

You can use a CICS bundle to create, edit, and install a PROGRAM resource definition. If you create a PROGRAM resource in this way, you must use the CICS bundle to manage the lifecycle of that resource, and you cannot manage the resource independently. You can inquire on a PROGRAM resource that is dynamically generated by a CICS bundle, by using the **EXEC CICS INQUIRE PROGRAM** or **CEMT INQUIRE PROGRAM** command. However, you cannot issue SET or DISCARD commands against a PROGRAM resource that is dynamically generated by a CICS bundle. You must issue the commands against the BUNDLE resource, and CICS applies them to the PROGRAM resource.

For more information about defining resources in CICS bundles, see Defining CICS bundles.

PROGRAM resources as application entry points

The PROGRAM resource can be defined as an application entry point. For more information about how to define the entry point, see Defining application entry points in the CICS Explorer product documentation.

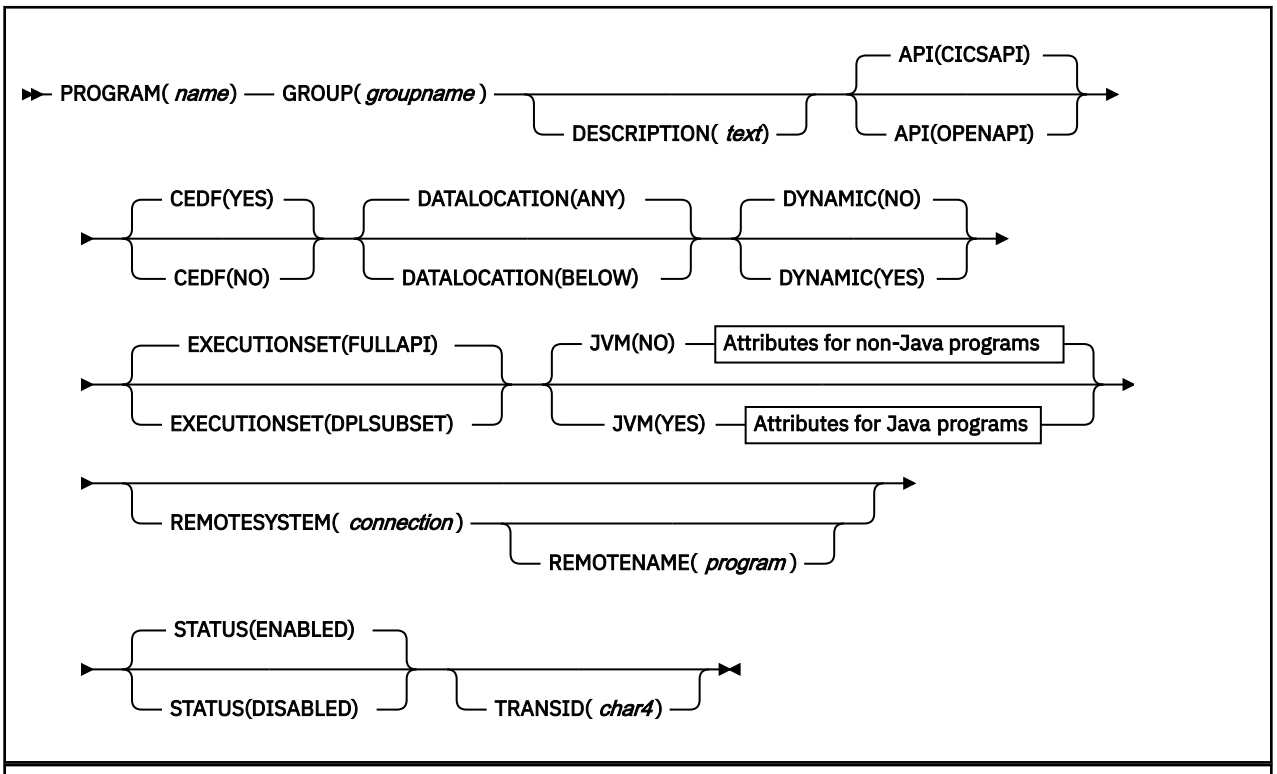
Private PROGRAM resources for applications

The PROGRAM resource is supported as a private resource for an application version. A program that is auto-installed by a task for an application that is deployed on a platform is also private to that version of the application.

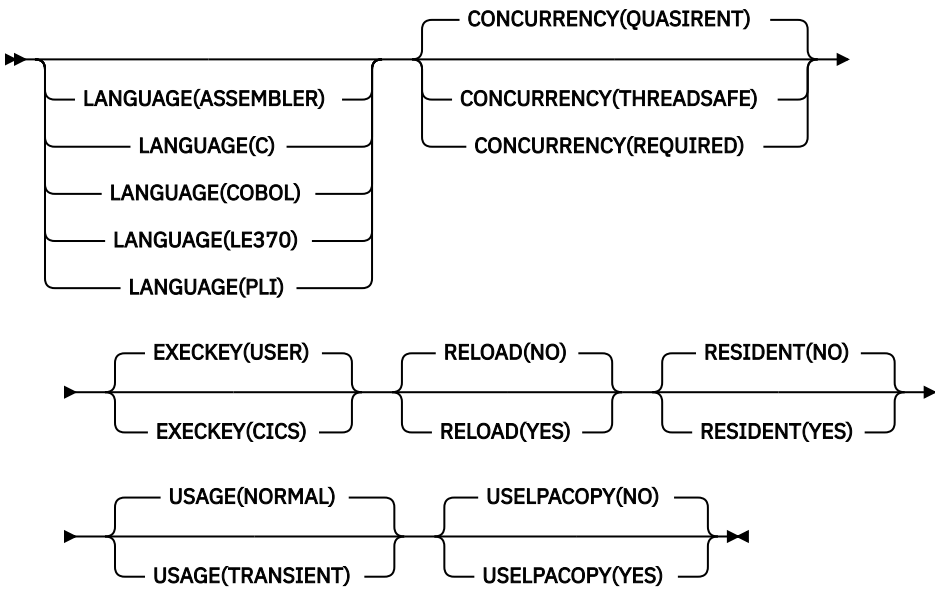
For more information about private resources, see Private resources for application versions.

PROGRAM attributes

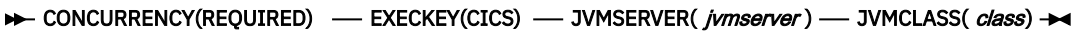
Describes the syntax and attributes of the PROGRAM resource.



Attributes for non-Java programs



Attributes for Java programs



API({CICSAPI|OPENAPI})

Specifies which API is to be used by the program. The API attribute applies to the following programs:

- User application programs
- PLT programs
- User-replaceable programs
- Task-related user exit programs. For more information, see Task-related user exit programs.

The API attribute does not apply to global user exits.

CICSAPI

The program is restricted to use of the CICS permitted application programming interfaces only.

If the program is defined with CONCURRENCY(QUASIRENT), it always runs on the quasi-reentrant (QR) TCB. If the program is defined with CONCURRENCY(THREADSAFE), it runs on whichever TCB is in use by CICS at the time that is determined as suitable, and if the program is defined with CONCURRENCY(REQUIRED), it always runs on an open TCB.

OPENAPI

The program is not restricted to the CICS application programming interfaces.

CICS runs the program on its own open TCB. The type of open TCB used depends on the value of the EXECKEY attribute and the language of the program.

If CICS requires a switch to the QR TCB when running a command, it returns to the open TCB before handing control back to the application program.

To specify the OPENAPI attribute, your program must be coded to threadsafe standards and defined with CONCURRENCY(REQUIRED). The primary use for OPENAPI programs is to move application workloads off the QR TCB and onto multiple open TCBs. You can benefit from better exploitation of server resources to achieve better throughput.

Note: The combination of CONCURRENCY(THREADSAFE) API(OPENAPI) supported in previous releases is deprecated but is kept for compatibility, and produces the same behavior as CONCURRENCY(REQUIRED) API(OPENAPI).

Use of other (non-CICS) APIs in OPENAPI programs is possible. If an open TCB is blocked by an operating system wait, only the single application is affected and not the whole of CICS. Such OPENAPI programs are not permitted to run on the QR TCB precisely because of this risk of blocking the TCB by an operating system wait and thus affecting the whole of CICS. Nevertheless, OPENAPI programs still have obligations to the CICS system as a whole. For more information, see Multithreading: Reentrant, quasi-reentrant, and threadsafe programs .

Important: Use of other (non-CICS) APIs within CICS is entirely at the discretion and risk of the user. No testing of other (non-CICS) APIs within CICS has been undertaken and use of such APIs is not supported by IBM Service.

CEDF({YES|NO})

Specifies the action of the execution diagnostic facility (EDF) when the program is running under EDF control.

NO

The EDF diagnostic screens are not displayed.

YES

The EDF diagnostic screens are displayed. If the program is translated with the NOEDF option, only the program initiation and termination EDF screens are displayed. See Table 3 on page 123.

Table 3. The effect on programs of CEDF(NO) and NOEDF

CEDF option on PROGRAM	EDF specified for translator	NOEDF specified for translator
YES	ALL EDF screens	Program initiation and termination screens only
NO	NO EDF screens	NO EDF screens

Note: The table shows how the CEDF option on the program resource definition interacts with the EDF option specified for the translator.

CONCURRENCY({QUASIRENT|THREADSAFE|REQUIRED})

Specifies whether the program is written to threadsafe standards, or is only quasi-reentrant. You can specify the CONCURRENCY attribute for all CICS executable program objects:

- User application programs.
- PLT programs.
- User-replaceable programs.
- Global user exit programs. For more information, see Global user exit programs.
- Task-related user exit programs. For more information, see Task-related user exit programs.

Java programs and any C and C++ programs compiled with the XPLink option must be defined as CONCURRENCY(REQUIRED).

QUASIRENT

The program is quasi-reentrant only, and relies on the serialization provided by CICS when accessing shared resources.

The program is restricted to the CICS permitted programming interfaces, and must comply with the CICS quasi-reentrant rules. For more information, see Multithreading: Reentrant, quasi-reentrant, and threadsafe programs .

This value is supported for all executable programs.

CICS ensures that the program always runs under the QR TCB, even when control is returned after it has started a JVM or an open API task-related user exit, or when it interacts with threadsafe programs.

THREADSAFE
The program is written to threadsafe standards. When it accesses shared resources it takes into account the possibility that other programs might be running concurrently and attempting to modify the same resources. The program, therefore, uses appropriate serialization techniques when accessing any shared resources.

For information about CICS DB2 application programs, see Preparing CICS DB2 programs for execution and production.

For information about writing threadsafe application programs, see Threadsafe programs.

This value is supported for all executable programs.

REQUIRED
The program is written to threadsafe standards. CICS starts the program on an open TCB and ensures that the program always runs on an open TCB. If CICS switches to the QR TCB to run a CICS command, it returns to the open TCB before handing control back to the application program. The type of open TCB used depends on the API setting and the language of the program.

- Java™ programs and OSGi bundles that run in a JVM server use a T8 TCB.
- C or C++ XPLink programs operate like OPENAPI programs and use an X8 TCB if CICS key is set, and an X9 TCB if user key is set.
- COBOL, PL/I, non-XPLink C or C++, and assembler language programs that also specify API(CICSAPI) use an L8 TCB because CICS commands can operate on this TCB irrespective of the execution key of the program.
- COBOL, PL/I, non-XPLink C or C++, and assembler language programs that also specify API(OPENAPI) use an L8 TCB if CICS key is set or use an L9 TCB if user key is set.

REQUIRED is applicable to user application programs, PLT programs, and user-replaceable programs.

DATALOCATION({ANY|BELOW})
Commands that use the SET option can return a data address to an application program; this operand specifies the location of the data. For example, in the command **EXEC CICS RECEIVE SET(ptr-ref)**, *ptr-ref* is less than 16 MB if DATALOCATION(BELOW) is specified, but might be greater than 16 MB if DATALOCATION(ANY) is specified. DATALOCATION does not affect the operation of the GETMAIN command. For more information about where CICS obtains storage in response to this command, see GETMAIN.

ANY
The program can handle 31-bit addresses. This is the default. The address of the data can be above or below the 16 MB line. The values specified for the DATALOCATION attribute are independent of the addressing mode of the link-edited program. Programs that are link-edited with addressing mode AMODE=24 cannot access data above 16 MB; ensure that the value you specify is compatible with the addressing mode of the link-edited application program:

- Specify ANY for all 31-bit or 64-bit programs, unless they pass CICS data addresses on to other 24-bit programs.
- Specify DATALOCATION(BELOW) for an AMODE=24 program, unless storage addresses are being passed to a program that can access storage above 16 MB, or the program explicitly switches addressing mode.

BELOW
The program can handle only 24-bit addresses and must therefore only be given data located below the 16 MB line. If required, data is copied below the 16 MB line before passing its address to the application program.

DATALOCATION does not affect the operation of the GETMAIN64 or GET64 CONTAINER commands. For more information about these commands, see GETMAIN64 and GET64 CONTAINER.

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

DYNAMIC({NO|YES})
Specifies whether, if the program is the subject of a program-link request, the request can be dynamically routed.

NO
If the program is the subject of a program-link request, the dynamic routing program is not started.

For a distributed program link (DPL) request, the server region on which the program is to run must be specified explicitly on the REMOTESYSTEM attribute or on the SYSID option of the **EXEC CICS LINK** command; otherwise it defaults to the local region.

YES
If the program is the subject of a program-link request, the CICS dynamic routing program is started. If a remote region is not named on the SYSID option of the **EXEC CICS LINK** command, the routing program can route the request to the region on which the program is to run.

The DYNAMIC attribute takes precedence over the REMOTESYSTEM attribute; see REMOTESYSTEM.

For guidance information about the dynamic routing of DPL requests, see Dynamically routing DPL requests.

EXECKEY({USER|CICS})
Specifies the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. Except for reentrant programs (that is, programs link-edited with the RENT attribute), EXECKEY also defines, with the residency mode, into which of the DSAs CICS loads the program.

CICS
CICS gives control to the program in CICS key when it is called. CICS loads the program into one of the CICS-key DSAs - either the CDSA or the ECDSA, depending on the residency mode specified for the program.

In a CICS region with storage protection active, a CICS-key program has read and write access to CICS-key and user-key storage of its own task and all other tasks, whether transaction isolation is active.

USER
CICS gives control to the program in user key when it is called. CICS loads the program into one of the user-key shared DSAs - either the SDSA or the ESDSA, depending on the residency mode specified for the program.

In a CICS region with storage protection only active, a user-key program has read and write access to all user-key storage, but read-only access to CICS-key storage.

In a storage protection and transaction isolation environment, a user-key program has read and write access to the user-key task-lifetime storage of its own task only, and to any shared DSA storage, if the transaction is defined with ISOLATE(YES).

If a transaction is defined with ISOLATE(NO) in a transaction isolation environment, its user-key programs also have read and write access to the user-key task-lifetime storage of other transactions that are defined with ISOLATE(NO).

User-key programs always have read-only access to CICS-key storage.

The EXECKEY attribute is ignored in the following cases:

- First-level global user exit programs, task-related user exit programs, user-replaceable programs, and PLT programs always run in CICS key.

- If the program is link-edited with the RENT attribute, CICS loads the program into one of the read-only DSAs - either the RDSA or the ERDSA, depending on the residency mode specified for the program. The read-only DSAs are allocated from read-only storage only if RENTPGM=PROTECT is specified as a system initialization parameter.
- Programs called by COBOL dynamic CALL always run in the same key as the caller.

EXECUTIONSET({FULLAPI|DPLSUBSET})
Specifies whether you want CICS to link to and run a program as if it were running in a remote CICS region.

DPLSUBSET
Specify DPLSUBSET if you want CICS to link to the program and run it with the API restrictions of a remote DPL program. See Exception conditions for LINK command for details of the API restrictions for a DPL program.

FULLAPI
Specify FULLAPI if you want CICS to link to the program and run it without the API restrictions of a DPL program. The program can use the full CICS API.

The EXECUTIONSET attribute applies only in these cases:

- To programs that are being linked to, and not to those programs that are the first to be given control by a transaction.
- When the REMOTESYSTEM name is the same name as the local CICS region. Its purpose is to test programs in a local CICS environment as if they were running as DPL programs.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

JVM({NO|YES})
Specifies whether the program is a Java program that must run in a Java Virtual Machine (JVM).

NO
The program is not to run in a JVM.

YES
The program is to run in a JVM. Specify a class name in the JVMCLASS attribute if you specify JVM(YES).

In addition to YES and NO, you can also specify DEBUG, but in compatibility mode only (see Sharing the CSD between different releases of CICS).

If you set JVM to YES, and a value is also set for JVMPROFILE (an obsolete attribute), program installation is prevented.

JVMCLASS(*class*)
Specifies the name (up to 255 characters in length) of the service for a Java program.

- For OSGi bundles that run in a JVM server, this value is the name of the OSGi service. The OSGi service is registered when you install the BUNDLE resource that contains the OSGi bundle. You can look up the name of the OSGi service in the Bundle Parts view in CICS Explorer.
- For Java programs that run in a JVM server, this value is the class name qualified by the package name.

- For applications in a Liberty JVM server, this value is `wlp:className#methodName`. For example: `wlp:com.ibm.cics.server.examples.tsq.ClassOne#myLinkMethod`. The method name is optional.

The names are case sensitive and must be entered with the correct combination of uppercase and lowercase letters. If you use a terminal, ensure that uppercase translation is suppressed.

The value for JVMCLASS can include the following characters:

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

This attribute applies only to Java applications running under the control of a JVM. If you specify JVM(NO), CICS ignores any value specified in the JVMCLASS.

JVMPROFILE(*jvmprofile*)

This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS. To run a Java program in a JVM server, use the JVMSERVER attribute instead.

JVMSERVER(*jvmserver*)

Specifies the name (up to 8 characters in length) of the JVMSERVER resource that contains the OSGi service. A JVMSERVER resource represents the JVM server runtime environment in CICS. The JVM server runs all programs in the CICS key.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . - _ % ? ! : | = , ;
```

LANGUAGE({COBOL|ASSEMBLER|LE370|C|PLI})

Specifies the program language.

ASSEMBLER
An assembler language program that was not translated using the LEASM translator option. LEASM is used to translate those assembler programs that are Language Environment-conforming MAIN programs.

C
A C or C++ program that was not compiled by a Language Environment-conforming compiler.

COBOL
A COBOL program.

LE370
A program that uses multi-language support, or has been compiled by a Language Environment-conforming compiler, or is an assembler MAIN program that was translated using the LEASM option to produce a Language Environment-conforming program.

PLI
A PL/I program.

In most cases, you do not have to specify the LANGUAGE attribute, because the CICS program manager deduces the correct language and ignores the value you specify. However, if the program is written in assembler language and does not have a DFHEAI or DFHEAG stub, CICS cannot deduce the language and you must specify the appropriate value: If the language is not specified and CICS cannot deduce it, transactions that attempt to use the program abend with code ALIG.

Although, you usually do not have to specify a value for this attribute, be aware that the value specified is returned in the LANGDEDUCED and LANGUAGE options of the INQUIRE PROGRAM command. Programs that use this command might be affected if you change the value of this attribute.

This attribute does not apply to JVM programs. CICS deduces that the program is a Java program to run under the control of a JVM when JVM(YES) is specified.

PROGRAM(*name*)
Specifies the name of this PROGRAM definition. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

It is preferable not to use names that start with DFH because these characters are reserved for use by CICS.

CICS can delete or discard them and replace them with CICS resources when you upgrade or provide new function or fixes.

To use the program in an active CICS region, it must be link-edited into one of the libraries specified as part of the DFHRPL or dynamic LIBRARY concatenation. If the program is reentrant, you can place it in the link pack area (LPA). For more information about installing application programs, see Installing application programs.

RELOAD({NO|YES})
Specifies whether a program control link, load, or XCTL request is to bring in a fresh copy of a program. This attribute does not apply to JVM programs.

NO
Any valid copy of the program currently in storage is reused for the request.

YES
A fresh copy of the program is brought into storage for every request. Furthermore, each of these program copies must be removed from storage explicitly, using a storage control FREEMAIN request, when it is no longer required and before the transaction terminates. If the relevant FREEMAIN requests are not issued, areas of the DSA/EDSA become tied up with inaccessible program copies, potentially causing storage shortage or fragmentation.

Note: If a new version of the program has been placed in the LIBRARY concatenation, a NEWCOPY or PHASEIN must be issued for the program before the new version is loaded.

You can use RELOAD(YES) to load tables or control blocks that are modified by the execution of any associated programs. Do not specify this value for the first program loaded for a task, because the task does not have a way to issue a FREEMAIN request for the program.

You must specify RELOAD(YES) for nonreentrant programs.

For more information about the RELOAD attribute, see Controlling the use of modules from the MVS link pack area.

REMOTENAME(*program*)
Specifies the name of the program on the remote CICS region.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

If you specify REMOTESYSTEM and omit REMOTENAME, the REMOTENAME attribute defaults to the same name as the local name (that is, the program name on this resource definition).

REMOTESYSTEM(*connection*)
Specifies the name of an IPCONN or a CONNECTION resource that defines a link to the remote CICS region on which the program resides. Specify this attribute if you want CICS to ship a distributed program link (DPL) request to another CICS region.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

Besides the REMOTESYSTEM attribute of the program definition, the DPL server region can also be specified by the following:

- The application program, using the SYSID option of the **EXEC CICS LINK PROGRAM** command
- The dynamic routing program.

The rules of precedence are as follows:

1. If an application program issues a DPL request, and the SYSID option on the **LINK** command specifies a remote CICS region, CICS ships the request to the remote region.

   If the installed program definition specifies DYNAMIC(YES), or there is no installed program definition, the dynamic routing program is called for notification only; it cannot reroute the request.
2. If an application program issues a DPL request, but the SYSID is the same name as the local CICS region or the SYSID option is not specified:
   a. If the installed program definition specifies DYNAMIC(YES), or there is no installed program definition, the dynamic routing program is called and can route the request.

      The REMOTESYSTEM attribute of the program definition, if specified, names the default server region passed to the dynamic routing program.
   b. If the installed program definition specifies DYNAMIC(NO), CICS ships the request to the remote system named on the REMOTESYSTEM attribute. If REMOTESYSTEM is not specified, CICS runs the program locally.

The rules for specifying the remote system name are the same as for the CONNECTION attribute of the CONNECTION resource definition.

Note: You must not specify remote attributes for any user-written CICS programs, such as the dynamic transaction routing or autoinstall user programs.

RESIDENT({NO|YES})
Specifies the residence status of the program. This attribute does not apply to JVM programs.

NO
The program is not to be permanently resident. This value must be specified if RELOAD(YES) is specified.

YES
The program is to be loaded on first reference and is then to be permanently resident in virtual storage, but is to be pageable by the operating system. When you specify RESIDENT(YES), CICS assumes a specification of USAGE(NORMAL).

For more information about the effects of the RESIDENT attribute, see Defining programs as resident, nonresident, or transient.

RSL
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS. For more information, see Obsolete attributes.

STATUS({ENABLED|DISABLED})
Specifies the program status.

DISABLED
The program cannot be used.

ENABLED
The program can be used.

This attribute is ignored for PROGRAM resources that are dynamically generated by a CICS bundle. The initial status of a PROGRAM resource is derived from the initial status of the bundle that defines the resource.

TRANSID(*name*)

If the program is dynamic, this value is the default TRANSID used for the distributed program link (DPL) request. If the program is not dynamic, this value specifies the name of the transaction you want the remote CICS to attach, and under which it is to run the remote program.

If you do not specify a transaction name on the TRANSID attribute, the remote region runs the DPL program under one of the following CICS-supplied default mirror transactions. If you have defined a user transaction in the remote region to call the mirror program, the DPL program runs under that transaction ID.

CPMI
The mirror transaction that is used for LU6.2 connections that require data conversion and for TCP/IP and IPIC requests from the CICS Transaction Gateway for Multiplatforms.

CSMI
The CICS mirror transaction for MRO and LU6.2 connections with sync level 2 and for EXCI and IPIC requests from the CICS Transaction Gateway for z/OS.

USAGE({NORMAL|TRANSIENT})
Specifies when the storage for this program is released. This attribute does not apply to JVM programs.

NORMAL
When the resident use count (RESCOUNT) for this program reaches zero, it becomes eligible for removal from storage as part of the normal dynamic program storage compression process.

This value must be specified if RELOAD(YES) is specified.

TRANSIENT
When the resident use count (RESCOUNT) for this program becomes zero, the storage for this program is released. Specify this value for programs that are referenced infrequently.

USELPACOPY({NO|YES})
Specifies whether the program is to be used from the link pack area (LPA). This attribute does not apply to JVM programs.

NO
The program is not to be used from the LPA. It is loaded into the CICS address space.

YES
The program can be used from the LPA if LPA=YES is specified as a system initialization parameter. The use of the program from the LPA requires that it has been installed there and that the program is not named by the PRVMOD system initialization parameter. For more information, see Controlling the use of modules from the MVS link pack area.

PROGRAM resource definitions

PROGRAM resource definitions describe the control information for a program that is stored in the program library and used to process a transaction.

SESSIONS resources

A SESSIONS resource defines the logical link between two CICS systems that communicate using intersystem communication (ISC) or multiregion operation (MRO).

Before two systems can communicate using ISC or MRO, they must be logically linked through one or more sessions. The nature of the logical link determines how they can communicate. CICS does **not** use the SESSIONS name when the definition has been installed in the active system. This name is used only to identify the definition in the CSD file.

You use the CONNECTION attribute of the SESSIONS resource to name the CONNECTION with which these SESSIONS are associated when they are installed in the active systems.

Special considerations for different session types are:

MRO links and sessions
When you install a SESSIONS definition for MRO, you are telling CICS about a set of parallel sessions between this CICS and another CICS. The number of sessions is determined by the SENDCOUNT and RECEIVECOUNT attributes. The SEND sessions are identified by names created from the SENDPFX and SENDCOUNT attributes. The RECEIVE sessions are identified by names created from the RECEIVEPFX and RECEIVECOUNT attributes.

APPC (LUTYPE6.2) links and parallel sessions
When you install the SESSIONS definition, the sessions are grouped (for the benefit of z/OS Communications Server) into a modeset, which is identified by the MODENAME. The individual sessions are named by a counter; the first session created is named -999, the second -998, and so on. The value of this counter is retained over a warm or emergency start. The number of sessions created is controlled by the MAXIMUM attribute on the SESSIONS definition.

LUTYPE6.1 CICS-CICS ISC links and sessions
The way in which the sessions are identified by CICS depends on the way you defined them, using SENDPFX, SENDCOUNT, RECEIVEPFX, and RECEIVECOUNT like MRO sessions, or using SESSNAME as for CICS-IMS sessions.

Note: Use APPC for all new CICS-CICS ISC links.

LUTYPE6.1 CICS-IMS links and sessions
When you install the SESSIONS definitions in the active CICS system, CICS identifies each session by the SESSNAME attribute.

INDIRECT connections
Because the association between an INDIRECT link and the intermediate systems used for communicating with it is made at installation time, install the definition for the intermediate system before the definition for the INDIRECT link. If you install the INDIRECT link first, it remains dormant until the intermediate definition is installed, and until any other already installed connections that make reference to it are resolved. For example, System A is indirectly connected with system C through system B. In system A, install the following definitions in this order:

1. The intermediate system:

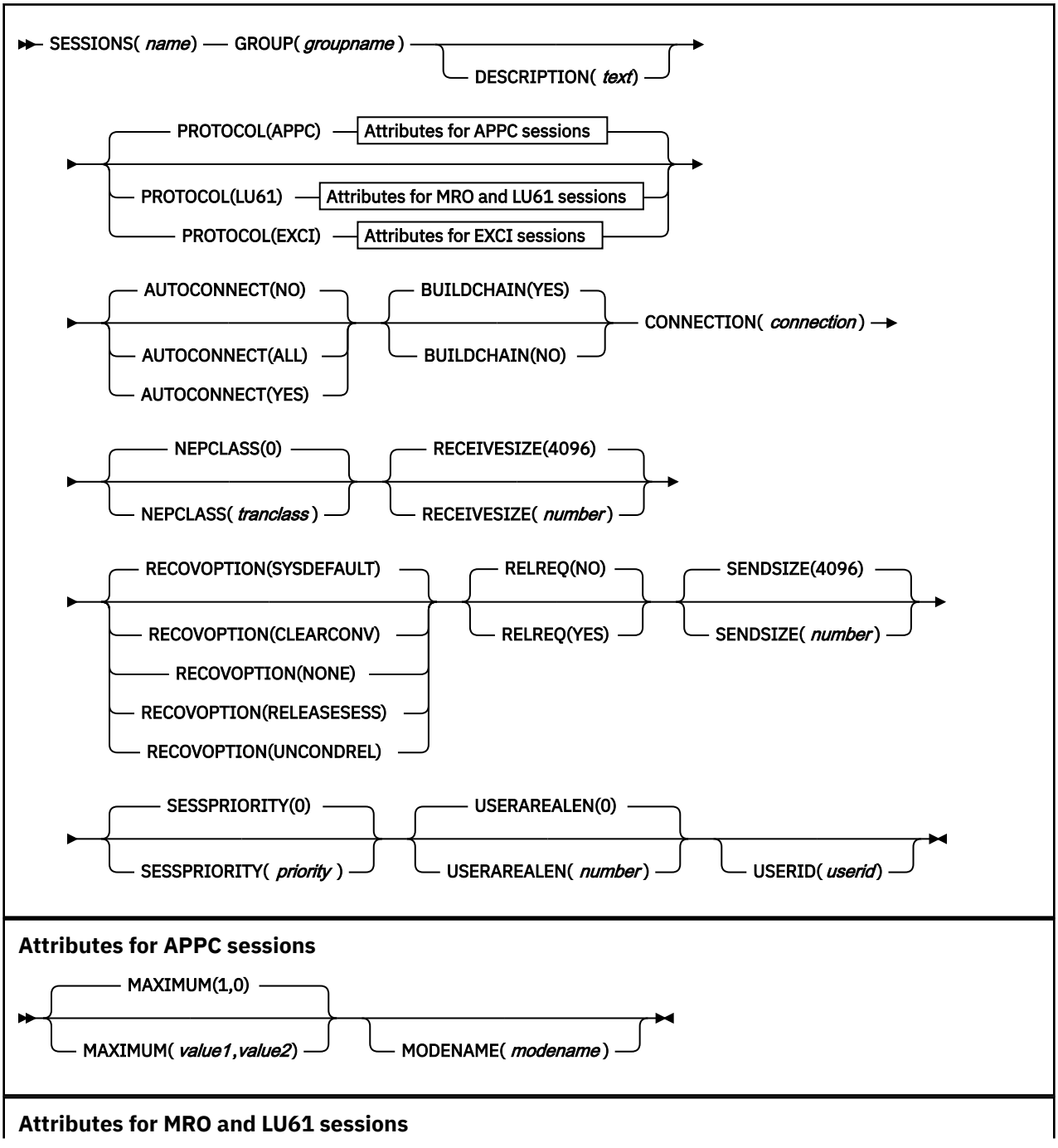
```
CONNECTION(B) NETNAME(B) ACCESSMETHOD(IRC) ...
```

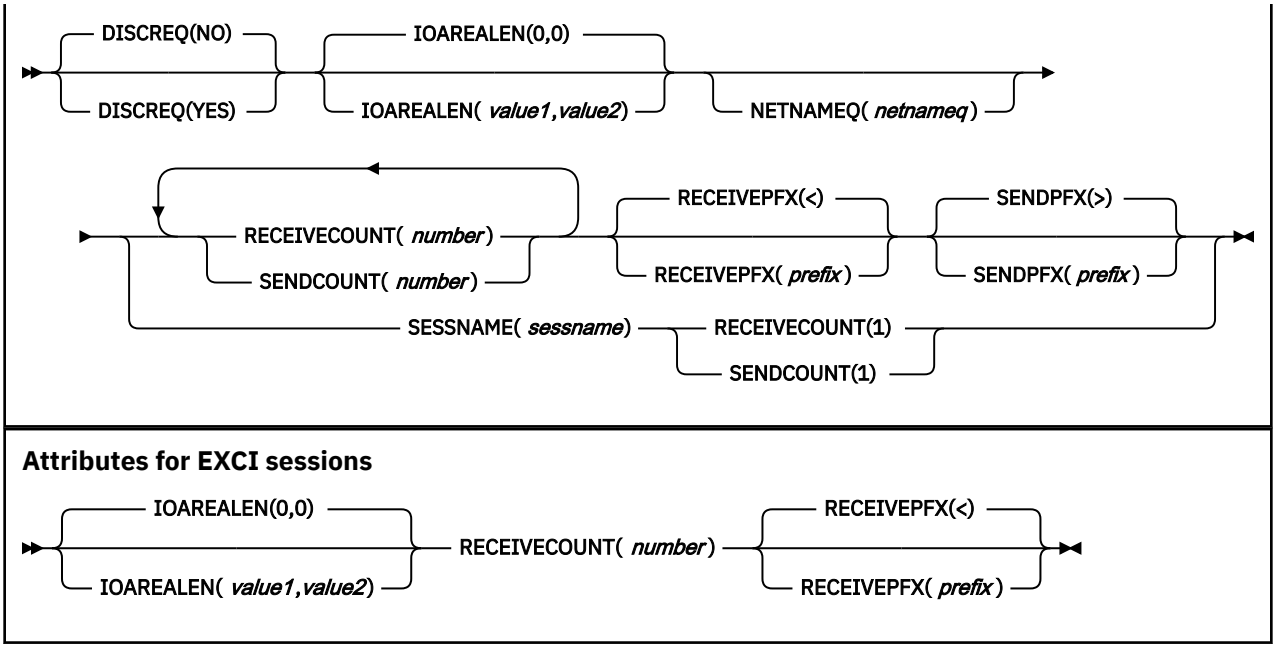
2. The INDIRECT link

```
CONNECTION(C) NETNAME(C) ACCESSMETHOD(INDIRECT)
   INDSYS(B) ...
```

SESSIONS attributes

Describes the syntax and attributes of the SESSIONS resource.





AUTOCONNECT({NO|YES|ALL})
specifies how connections are to be established. What you have to specify for LU6.1 and APPC sessions is discussed below:

APPC sessions
For a z/OS Communications Server-connected system that has AUTOCONNECT(YES) or (ALL) on the connection definition:

NO
CICS does not attempt to bind any sessions when the connection is established. However, one or more user sessions may be allocated as part of any ACQUIRE CONNECTION processing which takes place.

YES or ALL
A contention-winner session is established (that is, BIND is performed) during CICS initialization, or when communication with z/OS Communications Server is started using the CEMT SET VTAM OPEN command. If the connection cannot be made at this time because the remote system is unavailable, the link must be subsequently acquired using the CEMT SET CONNECTION(sysid) INSERVICE ACQUIRED command, unless the remote system becomes available in the meantime and itself initiates communications.

AUTOCONNECT(ALL) should not be specified for sessions to other CICS systems, because this can cause a bind race.

For a z/OS Communications Server-connected system that has AUTOCONNECT(NO) on the CONNECTION definition:

ALL
All sessions, not just contention winners, are established when the connection is acquired by issuing CEMT SET CONNECTION(name) ACQUIRED, or when the remote system itself initiates communication.

NO
CICS does not attempt to bind any sessions when the connection is established. However, one or more user sessions may be allocated as part of any ACQUIRE CONNECTION processing that takes place.

YES
Contention-winner sessions are established when the connection is acquired by issuing CEMT SET CONNECTION(sysid) ACQUIRED, or when the remote system itself initiates communication.

LU6.1 sessions
Specify AUTOCONNECT(YES) on the SESSIONS if you want the connection to be established at initialization or CEDA install.

Specify AUTOCONNECT(NO) on the SESSIONS if you do not want the connection to be established at initialization or CEDA installation.

BUILDCHAIN({YES|NO})
specifies whether CICS is to perform chain assembly before passing the input data to the application program.

NO
Any TIOA received by an application program from this logical unit contains one request unit (RU).

YES
Any terminal input/output area (TIOA) received by an application program from this logical unit contains a complete chain.

CONNECTION(*connection*)
specifies the name of the connection definition that you want to use with this session definition. The name can be up to four characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

Note that the CONNECTION definition must be in the same GROUP as the SESSIONS definition.

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

DISCREQ({NO|YES})
specifies whether disconnect requests are to be honored. DISCREQ applies to LUTYPE6.1 ISC sessions, but not to MRO sessions where CICS is not dealing with z/OS Communications Server devices.

DISCREQ does not apply to APPC (LUTYPE6.2) sessions. When APPC is used, individual sessions are acquired as transactions need them, then are subsequently freed. Because it is possible to have multiple sessions between APPC logical units, there should never be a problem of one request holding up another. It is not possible to disconnect an individual APPC session; instead, you can issue a CEMT SET CONNECTION RELEASED command.

NO
CICS is not to honor a disconnect request for a z/OS Communications Server device.

YES
CICS is to honor a disconnect request for a z/OS Communications Server device, and issue a z/OS Communications Server CLSDST macro instruction to terminate the z/OS Communications Server session with that logical unit.

CESF LOGOFF or GOODNIGHT commands issued from the terminal also cause disconnection if you specify DISCREQ(YES).

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

INSERVICE
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS.

IOAREALEN({0|*value1*},{0|*value2*})
specifies the length, in bytes, of the terminal input/output area to be used for processing messages transmitted on the MRO link.

value1
value1 specifies the minimum size of a terminal input/output area to be passed to an application program when a RECEIVE command is issued.

value2
If *value2* is not specified, or is less than *value1*, it defaults to the value of *value1*.

You can specify *value2* as greater than or equal to *value1*. In this case, when the size of an input message exceeds value1, CICS uses a terminal input/output area (TIOA) *value2* bytes long. When a transaction is attached on an MRO link, CICS uses a TIOA that is long enough to contain the initial input message. Otherwise, if the input message size also exceeds *value2*, the node abnormal condition program sends an exception response to the terminal.

You can waste both real and virtual storage by specifying an IOAREALEN value that is too large for most messages transmitted on your MRO link. On the other hand, if you specify an IOAREALEN value that is either zero or smaller than most of your messages, excessive FREEMAIN and GETMAIN activity may occur. This results in additional processor requirements.

MAXIMUM({1|*value1*},{0|*value2*}) (APPC only)
specifies the maximum number of sessions that are to be supported for the modeset. Value1 must be greater than or equal to value2.

1|*value1*
The maximum number of sessions in the group. This value can be in the range 1 through 999. The default is 1.

0|*value2*
The maximum number of sessions that are to be supported as contention winners. This value can be in the range 0 to 999. The default is 0. Note that this operand has no meaning for a single session connection.

SNA allows some resources (for example, switched lines) to be defined in the network as **limited resources**. At bind time, z/OS Communications Server indicates to CICS whether the bind is over a limited resource. When a CICS task frees a session across a limited resource, CICS unbinds the session if no other task wants to use it.

If the sessions are to use limited resources, specify MAXIMUM(*value1*,0). This causes any unbound session to be reset so that either side can then bind it as a winner when it is next required.

For further information on the effects of the MAXIMUM option, and the use of limited resources, see Effects of the MAXIMUM option of the SESSIONS resource

MODENAME(*modename*) (APPC only)
specifies the name that identifies a group of sessions for use on an APPC connection. The name can be up to eight characters in length, and must be the name of a z/OS Communications Server LOGMODE entry defined to z/OS Communications Server. It must not be the reserved name SNASVCMG. If you omit the modename it defaults to blanks. See Defining groups of APPC sessions for more information about z/OS Communications Server modenames.

The MODENAME must be unique for each group of sessions defined for any one intersystem link. That is, the MODENAME must be unique among the SESSIONS definitions related to one CONNECTION definition. It is passed to z/OS Communications Server as the LOGMODE name.

NEPCLASS({0|*tranclass*})
specifies the transaction class for the node error program. This value acts as the default.

0
This results in a link to the default node error program module.

tranclass
The transaction class for the (nondefault) node error program module. The value can be in the range 1 through 255. For programming information about the node error program, see Writing a node error program.

The NEPCLASS attribute is ignored for SNASVCMGR sessions.

NETNAMEQ(*netnameq*)
specifies the name by which the remote IMS system knows this particular session. This is used for CICS-IMS sessions. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 $ @ # . / - _ % & ¢ ? ! : | " = ¬ , ; < and >. Lowercase characters are converted to uppercase except when using the CREATE command.

OPERID
OPERPRIORITY
OPERRSL
OPERSECURITY
These attributes ares obsolete, but are supported to provide compatibility with earlier releases of CICS.

PROTOCOL({APPC|LU61|EXCI})
specifies the type of protocol that is to be used for an intercommunication link (ISC or MRO).

APPC (LUTYPE6.2)
Advanced program-to-program communication (APPC) protocol. Specify this for CICS-CICS ISC.

EXCI
The external CICS interface. Specify this to indicate that the sessions are for use by a non-CICS client program using the external CICS interface.

LU61
LUTYPE6.1 protocol. Specify this for CICS-CICS ISC, for CICS-IMS, or for MRO.

RECEIVECOUNT(*number*)
For MRO, and z/OS Communications Server LU6.1 sessions, and for sessions with EXCI clients, specifies the number of *receive sessions*; that is, sessions that normally receive before sending:

- MRO receive sessions (including sessions with EXCI clients) always receive before sending.
- z/OS Communications Server LU6.1 receive sessions normally receive before sending, but may send before receiving when there is a shortage of suitable send sessions.

If you do not specify the RECEIVECOUNT attribute, there are no receive sessions.

The number of receive sessions you can specify depends upon the length of the prefix specified in the RECEIVEPFX attribute:

- If you use the default receive prefix (<), or your own 1-character prefix, you can specify 1 through 999 receive sessions.
- If you use a 2-character prefix, you can specify 1 through 99 receive sessions.

You should also ensure that the value specified matches the number of send sessions in the partner system:

- If the partner is another CICS system, the value should match the SENDCOUNT specified in the partner system.

- If the partner is an EXCI client, you cannot specify the number of send sessions in the partner. However, there is an upper limit of send sessions in an EXCI address space. When this limit is reached, IRP rejects further requests for a session with SYSTEM_ERROR reason code 608. For information about how the limit is specified for the CICS system, see EXCI pipe allocation in Installing.

RECEIVEPFX({<|*prefix*})
specifies a 1-or 2-character prefix that CICS is to use as the first one or two characters of the receive session names (the names of the terminal control table terminal entries (TCTTEs) for the sessions).

Prefixes must not cause a conflict with an existing connection or terminal name.

< (MRO and EXCI sessions)
For MRO and EXCI sessions, if you do not specify your own receive prefix, CICS enforces the default prefix—the less-than symbol (<), which is used in conjunction with the receive count to generate receive session names.

CICS creates the last three characters of the session names. The acceptable characters are A-Z 1-9. These 3-character identifiers begin with the letters AAA, and continue in ascending sequence until the number of session entries reaches the limit set by the RECEIVECOUNT value. Note that receive session names are generated **after** the send sessions, and they follow in the same sequence.

For example, if the last session name generated for the send sessions is >AAJ, using the default sendprefix (>) CICS generates the receive session names as <AAK, <AAL, <AAM, and so on. (This method of generation of session identifiers is the same as for APPC sessions, except for the initial prefix symbol.)

If you use more than 46656 sessions (<AAA to <999), CICS allocates the next range of AAA< to 999<, again in a similar manner to APPC sessions.

A region with more than 46656 sessions might not perform well. You should consider the alternative of increasing the number of CICS regions.

Although you can define up to 93312 MRO sessions there is a current restriction that prevents you from attempting to acquire more than 65535 sessions in one attempt. This might occur when CICS starts or for a CEDA install for more than 65536 sessions if ALL the partner regions are up and running. Further sessions can be acquired later.

Note: If you specify your own prefix, CICS generates the session names in the same way as it does for LUTYPE6.1 sessions.

***prefix* (LUTYPE6.1 sessions)**
If the sessions are on LUTYPE6.1 ISC connections, you must specify a 1-or 2-character prefix. Do not use the default < symbol for LUTYPE6.1 sessions.

For LUTYPE6.1 sessions (and MRO if you specify your own 1-or 2-character prefix), CICS generates session names by appending a number to the prefix, either in the range 1 through 99, or 1 through 999. The number begins with 1 and is incremented by 1 until the specified RECEIVECOUNT is reached.

RECEIVESIZE({4096|*number*})
specifies the maximum z/OS Communications Server request unit (RU) size that these sessions are capable of receiving. The value must be between 1 and 30720 for LU61 sessions, or 256 and 30720 for APPC sessions. The default is 4096.

The value specified is transmitted to the connected logical unit. This value may be rounded down by CICS, depending on what value you specified, because the value must be transmitted in an architected form. The value may be negotiated down still further at BIND time.

If CICS is the secondary LU session, this indicates the maximum z/OS Communications Server request unit (RU) size that these sessions are capable of sending.

RECOVNOTIFY
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS.

RECOVOPTION({SYSDEFAULT|CLEARCONV| RELEASESESS|UNCONDREL|NONE})

This option applies to the recovery of sessions in a CICS region running with z/OS Communications Server persistent sessions, or with XRF.

z/OS Communications Server persistent sessions: In a CICS region running with persistent session support, this option specifies how you want CICS to recover the session, and return the terminal to service on system restart within the persistent session delay interval.

XRF: In a CICS region running with XRF support, this option specifies how you want CICS to recover the session, and return the terminal to service after an XRF takeover.

For all recovery options other than NONE, if the action taken is a z/OS Communications Server UNBIND, the UNBIND is followed by a z/OS Communications Server SIMLOGON.

CLEARCONV

z/OS Communications Server persistent sessions: CLEARCONV is not supported for APPC sessions. It defaults to SYSDEFAULT.

XRF: If AUTOCONNECT(YES) is specified, the session is restarted. If AUTOCONNECT(NO) is specified, the session is unbound.

NONE

z/OS Communications Server persistent sessions: In a CICS region running with persistent sessions support, this specifies that the session is not to be recovered at system restart within the persistent session delay interval: in effect, the sessions on the modegroup have no persistent sessions support. LU6.2 sessions are unbound and the modegroup CNOS value is reset to zero. After system restart, the session is reconnected automatically if you specify AUTOCONNECT(YES).

XRF: In a CICS region running with XRF support, this specifies that the logon state is not tracked by the alternate system, and the terminal session is not automatically recovered after a takeover; in effect, the terminal has no XRF support. After takeover, the terminal is reconnected automatically by the alternate system, if you specify AUTOCONNECT(YES).

RELEASESESS

z/OS Communications Server persistent sessions: RELEASESESS is not supported for APPC sessions. It defaults to SYSDEFAULT.

XRF: If AUTOCONNECT(YES) is specified, the session is restarted. If AUTOCONNECT(NO) is specified, the session is unbound.

SYSDEFAULT

z/OS Communications Server persistent sessions: In a CICS region running with persistent sessions support, this specifies that CICS is to select the optimum procedure to recover a session on system restart within the persistent session delay interval, depending on the session activity and on the characteristics of the terminal.

Although sessions are recovered, any transactions in-flight at the time of the failure are abended and not recovered. Transactions are also abended if the recovered session is being used by another CICS region over an APPC connection.

CICS recovers the session with the least possible impact, in one of the following ways:

- If the session was not busy at the time that CICS failed, no action is required.
- If the session was busy at the time that CICS failed, CICS issues a DEALLOCATE(ABEND) (equivalent to an EXEC CICS ISSUE ABEND) for the APPC conversation in progress at the time of the failure.
- If neither of these events apply, the session is unbound.

XRF: If AUTOCONNECT(YES) is specified, the session is restarted. If AUTOCONNECT(NO) is specified, the session is unbound.

UNCONDREL
Requires CICS to send an UNBIND request to release the active session. The UNBIND is sent whether or not the session was busy at the time of system restart (in the case of persistent sessions support) or takeover (in the case of XRF).

RELREQ({NO|YES})
specifies whether CICS is to release the logical unit upon request by another z/OS Communications Server application program.

SENDCOUNT(*number*)
For MRO, and z/OS Communications Server LU6.1 sessions only, specifies the number of *send sessions*; that is, sessions that normally send before receiving:

- MRO send sessions always send before receiving
- z/OS Communications Server LU6.1 send sessions normally send before receiving, but may receive before sending when there is a shortage of suitable receive sessions

The number of send sessions you can specify depends upon the length of the prefix specified in the SENDPFX attribute:

- If you use the default send prefix (>), or your own 1-character prefix, you can specify 1 through 999 send sessions
- If you use a 2-character prefix, you can specify 1 through 99 send sessions.

You should also ensure that the value specified matches the number of receive sessions in the partner system:

- If the partner is another CICS system, the value should match the RECEIVECOUNT specified in the partner system

If you do not specify the SENDCOUNT attribute, there are no send sessions. Do not specify the SENDCOUNT attribute when the partner is an EXCI client

SENDPFX({>|*prefix*})
specifies a 1-or 2-character prefix that CICS is to use as the first one or two characters of the send session names (the names of the terminal control table terminal entries (TCTTEs) for the sessions).

Prefixes must not cause a conflict with an existing connection or terminal name.

> (MRO sessions)
For MRO sessions, if you do not specify your own send prefix, CICS enforces the default prefix—the greater-than symbol (>), which is used in conjunction with the send count to generate send session names.

CICS creates the last three characters of the session names from the alphanumeric characters A through Z, and 1 through 9. These 3-character identifiers begin with the letters AAA, and continue in ascending sequence until the number of session entries reaches the limit set by the SENDCOUNT value.

For example, using the default prefix (>), CICS generates session names as >AAA, >AAB, >AAC, and so on. If you use more than 46656 sessions (>AAA to >999), CICS allocates the next range of AAA> to 999>. (This method of generation of session identifiers is the same as for APPC sessions, except for the initial symbol.)

A region with more than 46656 sessions might not perform well. You should consider the alternative of increasing the number of CICS regions.

Although you can define up to 93312 MRO sessions there is a current restriction that prevents you from attempting to acquire more than 65535 sessions in one attempt. This might occur during CICS start up or for a CEDA install for more than 65536 sessions if ALL the partner regions are up and running. Further sessions can be acquired later.

Note: If you specify your own prefix, CICS generates the session names in the same way as it does for LUTYPE6.1 sessions.

***prefix* (for LUTYPE6.1 sessions)**
If the sessions are on LUTYPE6.1 ISC connections, you must specify a 1-or 2-character prefix. Do not use the default > symbol for LUTYPE6.1 sessions.

For LUTYPE6.1 sessions (and MRO if you specify your own 1-or 2-character prefix), CICS generates session names by appending a number to the prefix, either in the range 1 through 99, or 1 through 999. The number begins with 1 and are incremented by 1 until the specified SENDCOUNT is reached.

SENDSIZE({4096|number})
specifies the maximum z/OS Communications Server request unit (RU) size that these sessions are capable of sending. The value must be between 1 and 30720 for LU61 sessions, or between 256 and 30720 for APPC sessions. The default is 4096. The value may be negotiated down at bind time. Increasing the value of SENDSIZE causes more storage to be allocated for the session but may decrease the number of physical messages sent between the two nodes.

If CICS is the secondary LU session, this attribute indicates the maximum z/OS Communications Server request unit (RU) size that these sessions are capable of receiving. The value must be between 256 and 30720.

SESSIONS(*name*)
specifies the name of this SESSIONS definition. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

This name is used to identify the SESSIONS definition on the CSD file. It is not used within the active CICS system.

SESSNAME(*sessname*)
specifies the symbolic identification to be used as the local half of a session qualifier pair in a CICS intercommunication parallel session. The name can be up to four characters in length.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

SESSPRIORITY({0|*priority*})
specifies the terminal priority. This decimal value (0 through 255) is used in establishing the overall transaction processing priority. (Transaction processing priority is equal to the sum of the terminal priority, transaction priority, and operator priority; this must not exceed 255.)

TRANSACTION
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS.

USERAREALEN({0|*number*})
Specify the length, in bytes, of the user area for this session, in the range 0 through 255. It should be made as small as possible. The terminal user area is initialized to zeros when the session is installed.

The terminal user area might be located in 31-bit storage or in 24-bit storage, depending on the value of the **TCTUALOC** system initialization parameter. The default is that the terminal user area can be in either 31-bit storage or 24-bit storage, but CICS uses 31-bit storage wherever possible. If you require the terminal user area to be in 24-bit storage, because you have application programs that are not capable of 31-bit addressing, specify the system initialization parameter **TCTUALOC=BELOW** for the CICS region.

USERID(*userid*)
specifies a user identifier used for sign-on (SEC=YES) and referred to in security error messages, security violation messages, and the audit trail. It must be a valid userid defined to the security manager, or operators will be unable to sign on. All access to protected resources depends on USERID.

This USERID overrides a SECURITYNAME specified on the CONNECTION definition.

The name can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

Installing sessions definitions

If you use the INSTALL command to install a new SESSIONS definition for MRO when a definition is already installed, you must close down all interregion communication (IRC) and open it again before you can use the definition.

About this task

Procedure

1. Close IRC down.
   Use the following command:

   ```
   CEMT SET IRC CLOSED
   ```
2. Install the resource definitions.
   Use the following command:

   ```
   CEDA INSTALL GROUP(groupname)
   ```
3. When you have successfully installed the group containing the definitions, open IRC again.
   Use the following command:

   ```
   CEMT SET IRC OPEN
   ```

Session resource definitions

Session definitions describe the nature of logical links between regions or systems that communicate using intersystem communication (ISC) or multiple region operation (MRO).

TCPIPSERVICE resources

A TCPIPSERVICE resource defines which TCP/IP services are to use CICS internal sockets support.

With TCPIPSERVICE resources, the CICS services that can be defined are ECI over TCP/IP (for CICS Clients), CICS web support (HTTP), IPIC, or a user-defined protocol. The TCPIPSERVICE definition allows you to manage these internal CICS interfaces, with CICS listening on multiple ports, with different flavors of ECI, CICS web support or the user-defined protocol on different ports.

TCPIPSERVICE definitions are for use only with the CICS-provided TCP/IP services, and have nothing to do with the z/OS Communications Server IP CICS Sockets interface. The TCP/IP Socket Interface for CICS is supplied with z/OS Communications Server, which is an integral part of z/OS and does not use the CICS SO domain.

TCPIPSERVICE resources in CICS bundles

You can use a CICS bundle to create, edit, and install a TCPIPSERVICE resource definition. If you create a TCPIPSERVICE resource in this way, you must use the CICS bundle to manage the lifecycle of that resource, and you cannot manage the resource independently.

You can inquire on a TCPIPSERVICE resource that is dynamically generated by a CICS bundle, by using the **EXEC CICS INQUIRE TCPIPSERVICE** or **CEMT INQUIRE TCPIPSERVICE** command.

You cannot issue the DISCARD command against a TCPIPSERVICE resource that is dynamically generated by a CICS bundle. You must discard the CICS bundle, and CICS applies the operation to the TCPIPSERVICE resource.

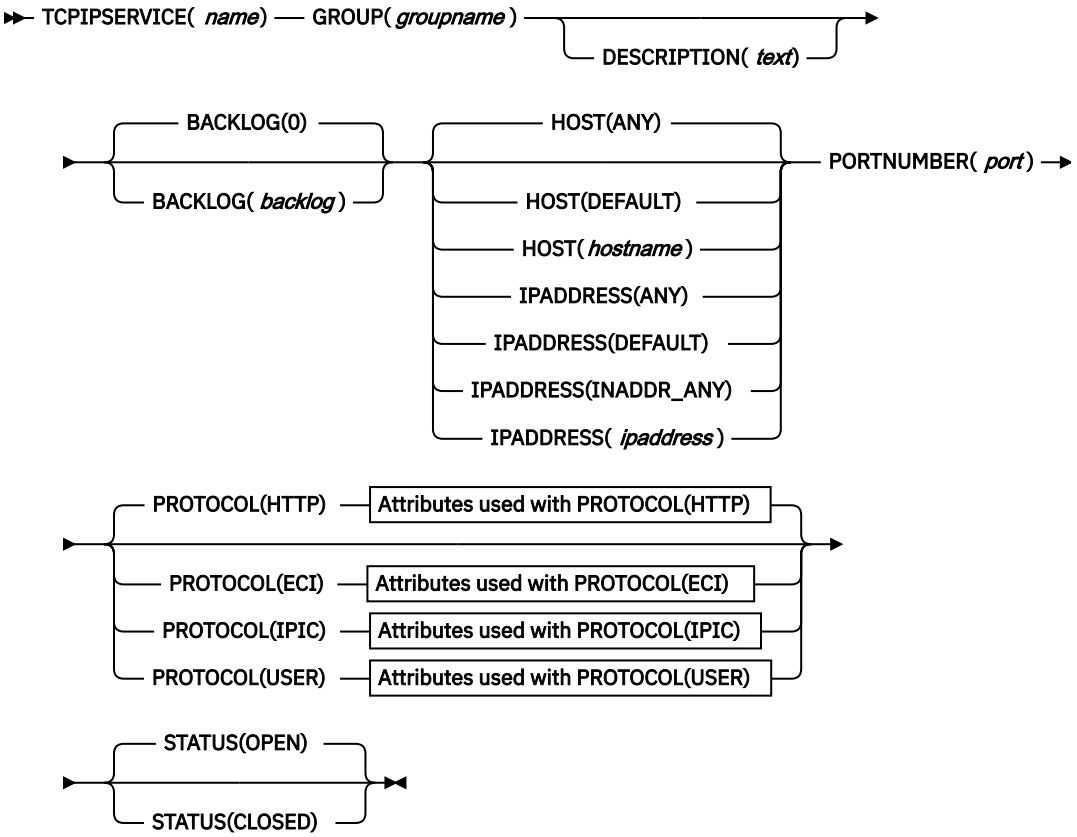
To modify the attributes of a TCPIPSERVICE resource that was defined and installed in a CICS bundle, use the resource editor in the CICS Explorer to modify the definition in the CICS bundle, and install a new version of the CICS bundle or of the application with which it was deployed. You can use the SET TCPIPSERVICE command to change the attributes of the dynamically generated resource, but these changes are not cataloged and are not recovered across a warm restart of CICS.

To change the status of a TCPIPSERVICE resource that was defined and installed in a CICS bundle, enable or disable the CICS bundle. If you have disabled the CICS bundle, but the service has not yet closed, you can issue the SET TCPIPSERVICE IMMCLOSE command against the dynamically generated resource to close the service immediately.

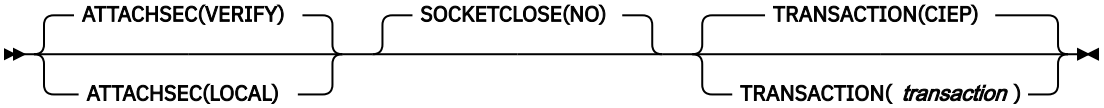
For more information about defining resources in CICS bundles, see Defining CICS bundles.

TCPIPSERVICE attributes

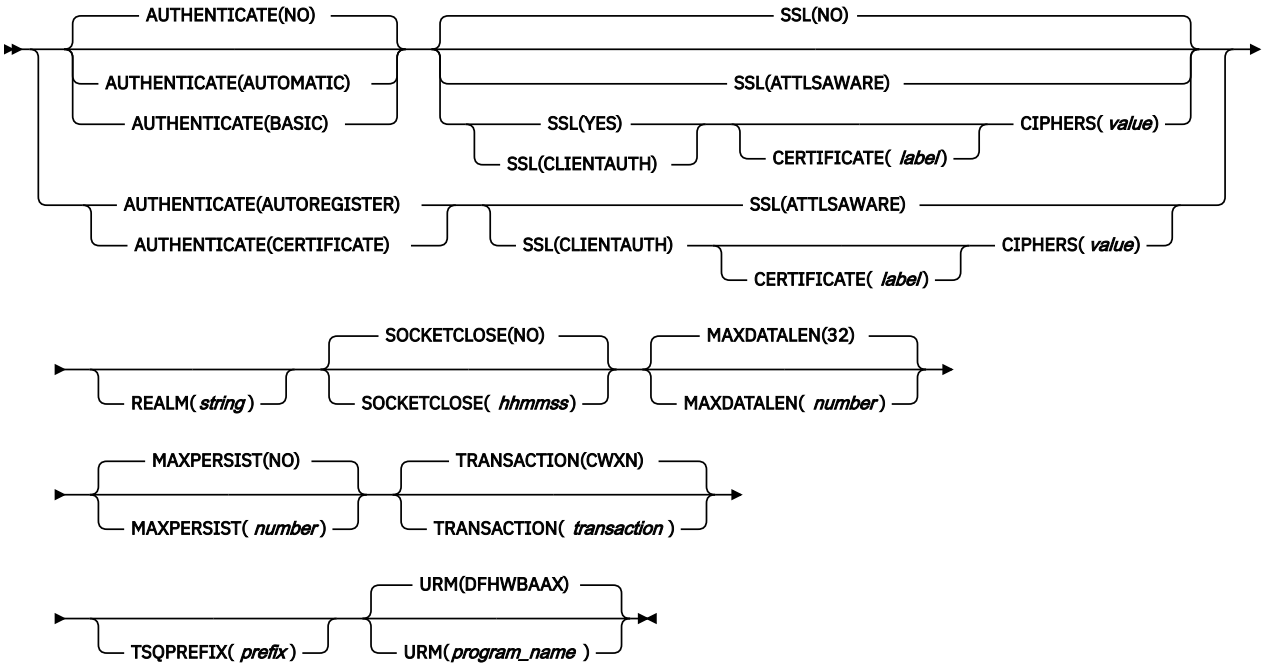
Describes the syntax and attributes of the TCPIPSERVICE resource.



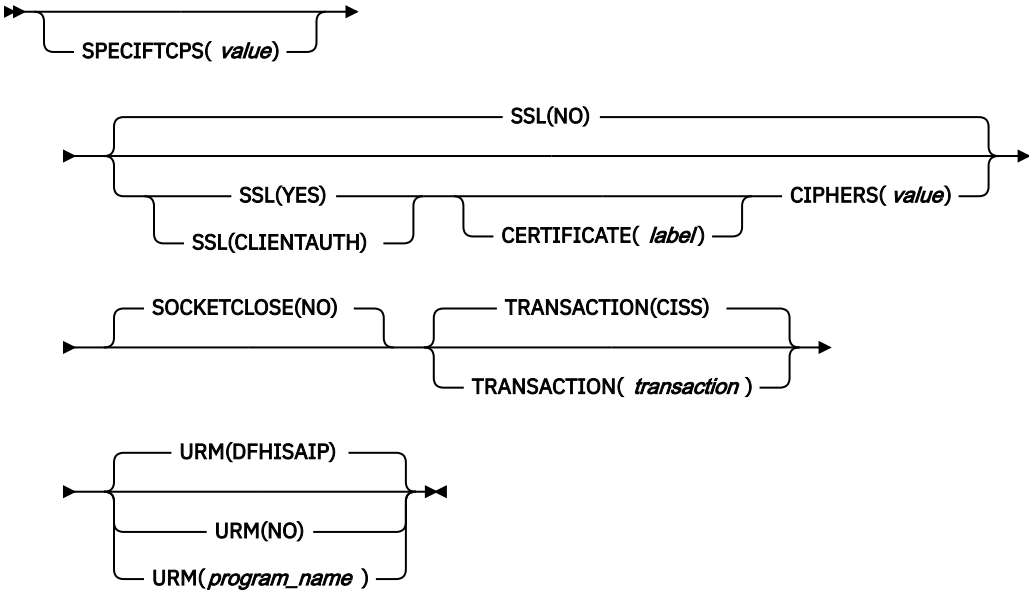
Attributes used with PROTOCOL(ECI)



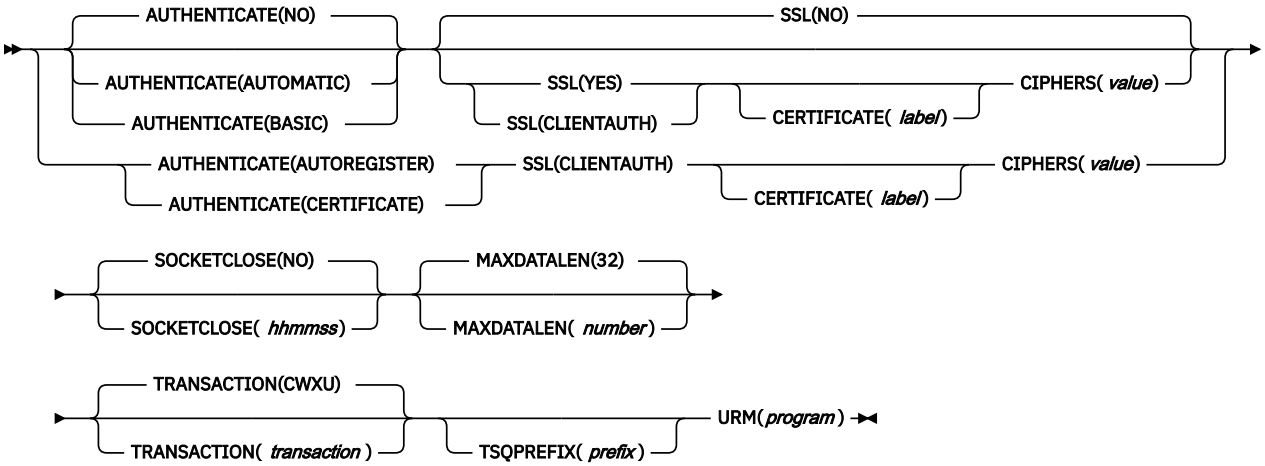
Attributes used with PROTOCOL(HTTP)



Attributes used with PROTOCOL(IPIC)



Attributes used with PROTOCOL(USER)



ATTACHSEC ({LOCAL | VERIFY})
Specifies the level of attach-time user security that is required for this connection.

This option is valid only for PROTOCOL(ECI).

LOCAL
Specifies that CICS does not require a user ID or password (or password phrase) from clients.

VERIFY
Specifies that incoming attach requests must specify a user ID, and a user password or password phrase. Specify VERIFY when connecting systems are unidentified and cannot be trusted.

AUTHENTICATE ({NO | ASSERTED | AUTOMATIC | AUTOREGISTER | BASIC | CERTIFICATE})
Specifies the authentication and identification scheme to be used for inbound TCP/IP connections for the HTTP and USER protocols. For the IPIC protocol, this attribute is not applicable. For the ECI protocol, this attribute is invalid. For more information about authentication, see Identification and authentication.

ASSERTED
The ASSERTED attribute in TCPIPSERVICE definitions is obsolete but is retained for compatibility reasons.

When PROTOCOL(HTTP) or PROTOCOL(USER) is specified:

NO
The client is not required to send authentication or identification information. However, if the client sends a valid certificate that is already registered to the security manager and associated with a user ID, then that user ID identifies the client.

BASIC
HTTP Basic authentication is used to obtain a user ID, and password or password phrase from the client.

If the client has sent an Authorization header, its contents are decoded as a user ID, and password or password phrase. If these are not valid, an HTTP 401 response is returned, together with a WWW-Authenticate header, which causes the client program to prompt the user for a new user ID, and password or password phrase. This process continues until the client either supplies a valid user ID, and password or password phrase, or cancels the connection.

When the user has been successfully authenticated, the user ID supplied identifies the client.

AUTOREGISTER
SSL client certificate authentication is used to authenticate the client.

- If the client sends a valid certificate that is already registered to the security manager and associated with a user ID, then that user ID identifies the client.
- If the client sends a valid certificate that is not registered to the security manager, then HTTP Basic authentication is used to obtain a user ID, and password or password phrase from the

client. If the password or password phrase is valid, CICS registers the certificate with the security manager, and associates it with the user ID. The user ID identifies the client.

- For a certificate to be registered, the client needs the appropriate access definition in RACF. The access required is READ access to the IRR.DIGTCERT.ADD profile in the FACILITY class.

Note: If you specify AUTHENTICATE(AUTOREGISTER), you must also specify SSL(CLIENTAUTH | ATTLSAWARE).

AUTOMATIC
This function combines the AUTOREGISTER and BASIC functions.

- If the client sends a certificate, then processing will continue as described in AUTOREGISTER.
- If the client does not send a certificate, then processing will proceed as described in BASIC.

CERTIFICATE
SSL client certificate authentication is used to authenticate and identify the client. The client must send a valid certificate that is already registered to the security manager, and associated with a user ID. If a valid certificate is not received, or the certificate is not associated with a user ID, the connection is rejected.

When the user has been successfully authenticated, the user ID associated with the certificate identifies the client.

Note: If you specify AUTHENTICATE(CERTIFICATE), you must also specify SSL(CLIENTAUTH | ATTLSAWARE).

Note: For the HTTP or USER protocol, the analyzer program (named by the user replaceable module (URM) attribute) can change the user ID supplied by the authentication process. If the authentication process does not supply a user ID, the analyzer program or URIMAP definition can supply one. Otherwise, the CICS default user ID is used.

BACKLOG(0|*backlog*)
Specifies, in the range 0 - 32767, the maximum number of inbound TCP/IP connection open requests that can be queued in TCP/IP waiting for CICS processing. When the maximum is reached, TCP/IP rejects additional requests.

If the value of BACKLOG is zero, CICS does not limit the backlog; instead, the value of TCP/IP attribute SOMAXCONN is used. If the value of BACKLOG is greater than SOMAXCONN, the value of SOMAXCONN is used.

For IPIC connections, it is recommended to set this value to 0 to allow this value to default to the TCP/IP default value for SOMAXCONN. Only increase this if the number of potential IPIC connections is greater than this value.

If performance tuning for HTTP connections is enabled, when CICS is at maximum capacity, all inbound HTTP connection open requests queue outside of CICS in the backlog queue of the TCPIPSERVICE's listening connection. Ensure that the BACKLOG value is large enough, because connection requests will be refused when this queue is full. To view the backlog values in use for a listening connection, use CICS TCP/IP services: Resource statistics, or use the NETSTAT ALL command to obtain information about the status of the local host, which includes information about the listening connection's backlog.

When connection balancing is in use, the backlog queue's depth is included in the processing that determines the most suitable listener, so consider the BACKLOG attribute across all TCPIPSERVICEs on CICS regions that share a port.

CERTIFICATE(*label*)
Specifies the label of an X.509 certificate that is used as a server certificate during the SSL handshake when the connection is acquired. If this attribute is omitted, the default certificate that is defined in the key ring for the CICS region user ID is used.

Certificate labels can be up to 32 bytes long.

The certificate must be stored in a key ring in the database of the external security manager. For more information, see Building a key ring manually.

This attribute cannot be specified unless SSL(YES) or SSL(CLIENTAUTH) is also specified.

CIPHERS(*value*)

The CIPHERS attribute can be specified in either of two ways:

- **Recommended:** The name of the SSL cipher suite specification file, which is a z/OS UNIX file in the `security/ciphers` subdirectory of the directory that is specified by the **USSCONFIG** system initialization parameter. For example, if **USSCONFIG** is set to `/var/cicsts/dfhconfig` and **CIPHERS** is set to `strongciphers.xml`, the fully qualified file name is `/var/cicsts/dfhconfig/security/ciphers/strongciphers.xml`. For more information, see Creating an SSL cipher suite specification file.
- A string of up to 56 hexadecimal digits that is interpreted as a list of up to 28 2-digit cipher suite codes.

When you use the CEDA transaction to define the resource, CICS automatically initializes the attribute with a default list of acceptable codes. For CICS to initialize the attribute, the KEYRING system initialization parameter must be specified in the CICS region where you are running CEDA. If KEYRING is not set, CICS does not initialize the attribute. The default list of 2-digit ciphers is 3538392F3233.

You can reorder the cipher codes or remove them from the initial list. However, you cannot add cipher codes that are not in the default list for the specified encryption level. To reset the value to the default list of codes, delete all of the cipher suite codes. The field is automatically repopulated with the default list.

When you use DFHCSDUP to define the resource, the CIPHERS attribute is not populated with a default value. If the resource is used to establish a secure connection and the CIPHERS attribute is empty, CICS uses the default list of 2-digit ciphers. An INQUIRE on the resource that is created in this manner, returns an empty string for the CIPHERS value.

Any unsupported ciphers are removed at run time. A list of the removed ciphers is reported in messages DFHSO0145 and DFHSO0146.

For more information, see Cipher suites and cipher suite specification files.

DESCRIPTION(*text*)

You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

DNSGROUP(*dnsgroup*)

This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS.

GROUP(*groupname*)

Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

GRPCRITICAL({NO|YES})

This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS.

HOST({ANY|DEFAULT|*hostname*})
Specifies the 116-character IPv4 or IPv6 address or host name on which CICS listens for incoming connections. Use HOST instead of IPADDRESS when you define new resources. Do not specify both HOST and IPADDRESS, because HOST always takes precedence over IPADDRESS. IPADDRESS is supported for existing programs that specify IPv4 function.

Possible values are as follows:

ANY
The ANY option has the same function as the ANY and INADDR_ANY options of IPADDRESS. The ANY option specifies that CICS listens on any of the addresses known to TCP/IP for the host system. You can have multiple IP addresses defined for a host. By specifying ANY, you also allow for the TCPIPSERVICE definition to be shared among CICS servers. If you specify ANY, CICS attempts to bind to the port on every stack where it is defined. If, in addition, you want more than one CICS region to bind to the port, you must specify the SHAREPORT option in every stack where the port is defined. If you do not do so, only one CICS region can bind to the port number in those stacks that do not have the SHAREPORT option. Subsequent attempts by other regions to bind to every stack fail, and CICS issues a message indicating that the port is in use.

If you specify the ANY option in a dual-mode (IPv4 and IPv6) environment, CICS attempts to reuse the most recent IPv4 or IPv6 address. If this is the first connection, and CICS cannot retrieve an address, `0.0.0.0` is returned, and no affinity is assigned.

DEFAULT
The DEFAULT option assigns affinity to the TCP/IP stack that is defined as the default in a multistack CINET environment.

If the DEFAULT option is used in a dual-mode (IPv4 and IPv6) environment, affinity is assigned to the IPv4 environment because the DEFAULT option is applied to the IPv4 environment.

If DEFAULT is used in a non-CINET environment or no default TCP/IP stack exists, an exception trace is written, `0.0.0.0` is returned, and no affinity is assigned.

Important: If you are operating in a dual-mode (IPv4 and IPv6) environment, specifying HOST(DEFAULT) forces all traffic to pass across the IPv4 network connection.

hostname
hostname can be a character host name, an IPv4 address, or an IPv6 address.

You can specify an address as a character name that can be looked up on the domain name server. The host name can be entered in uppercase, lowercase, or mixed case, but if a host name is specified instead of an IP address, the host name is converted to lowercase in the TCPIPSERVICE definition.

Do not use a character host name if you have a list of addresses at the domain name server, because *hostname* resolves against the first IP address only in the list (that is, the server does not listen on any of the IP addresses in the list for this host name). If you require a particular IP address in a list at the domain name server, define the IP address explicitly in *hostname*.

If you specify an IPv6 address (or a host name that resolves to an IPv6 address), ensure that you are operating in a dual-mode (IPv4 and IPv6) environment and that the client or server that you are communicating with is also operating in a dual-mode (IPv4 and IPv6) environment. For more information about IPv6, see Understanding IPv6 and CICS.

You can specify IPv4 and IPv6 addresses in a number of acceptable formats. See IP addresses for more information about address formats.

IPADDRESS({ANY|INADDR_ANY|DEFAULT|*ipaddress*})
Specifies the dotted decimal IPv4 address on which this TCPIPSERVICE listens for incoming connections. It must be of the form nnn.nnn.nnn.nnn where nnn is 0 through 255. You can use the HOST attribute to specify the same information as IPADDRESS, but HOST also supports an IPv6 format address and character host name. If you are using IPv6 connections, you must use the HOST attribute for your definitions instead of IPADDRESS. HOST always takes precedence over IPADDRESS. The IPADDRESS attribute interacts with HOST in a number of ways:

- If you specify HOST, IPADDRESS is always overwritten with a value that depends on the contents of HOST:
  - If you specify an IPv4 address, ANY, or DEFAULT in HOST, IPADDRESS is overwritten with the contents of HOST.
  - If you specify an IPv6 address or a character host name in HOST, IPADDRESS is overwritten with blanks.
- If you specify both HOST and IPADDRESS, the HOST value is always used:
  - If HOST contains an IPv4 address, ANY, or DEFAULT, IPADDRESS is populated with the contents of HOST.
  - If HOST contains an IPv6 address, IPADDRESS is overwritten with blanks.
- If you specify IPADDRESS (but not HOST), HOST is populated with the contents of IPADDRESS.

If you specify an IP address of `0.0.0.0` and the HOST option is blank, a warning is issued and the value ANY is assumed.

IPADDRESS is supported for existing IPv4 function only. Use the HOST option for new resources.

Possible values are:

ANY or INADDR_ANY
The TCPIPSERVICE listens on any of the addresses known to TCP/IP for the host system. It is possible to have multiple IP addresses defined for a host. Specifying ANY or INADDR_ANY also allows for the TCPIPSERVICE definition to be shared among CICS servers.

If you specify ANY or INADDR_ANY, CICS attempts to bind to the port on every stack where it is defined. If, in addition, you want more than one CICS region to bind to the port you must specify the SHAREPORT option in every stack where the port is defined. If you do not do so, only one CICS region is able to bind to the port number in those stacks that do not have the SHAREPORT option. Subsequent attempts by other regions to bind to every stack fails: CICS issues a message indicating that the port is in use. For information about the SHAREPORT option, see z/OS Communications Server: IP Configuration Reference.

DEFAULT
Assigns affinity to the TCP/IP stack that is defined as the default in a multi stack CINET environment. If DEFAULT is used in a non-CINET environment or there is no default TCP/IP stack, then an exception trace is written and no affinity is assigned.

ipaddress
The TCPIPSERVICE accepts connections on this particular address. If the address specified is not known to TCP/IP on the host system, the TCPIPSERVICE does not open. If you enter a specific address here, this definition might not be valid for CICS servers running on other regions, and you might not be able to share the definition with those servers.

MAXDATALEN({32|*number*})
Specifies, in kilobytes, the maximum length of data that can be received by CICS as an HTTP server, on the HTTP protocol or the USER protocol. The default value is 32 KB. The minimum is 3 KB, and the maximum is 524,288 KB. Use MAXDATALEN to guard against denial of service attacks involving the transmission of large amounts of data.

MAXPERSIST({NO|*number*})
Specifies the maximum number of persistent connections from web clients that the CICS region allows for this port at any one time. This setting applies only for the HTTP protocol.

- The default value NO means that there is no limit on the number of persistent connections.
- In a CICS region that is at risk of being overloaded with persistent connections, you can specify a suitable value (up to a theoretical maximum of 65535) based on the number of persistent connections that the CICS region can handle simultaneously. When this limit is reached and further web clients connect on the port, CICS requires the new clients to close the connection after they receive each response. When the new clients reconnect, if they connect to another CICS region that shares the port and has not reached its limit, they can maintain a persistent connection there

instead. An HTTP/1.1 server should normally allow persistent connections, so only set this option in a CICS region that has experienced performance problems due to persistent connections from long-lived web clients.

- If you specify a value of zero for this option, the CICS region does not allow persistent connections and requires every web client to close the connection after they receive each response. A zero setting for MAXPERSIST is not compliant with the HTTP/1.1 specification, so only use that setting if you have a special requirement for it in a CICS region that is not currently handling external requests, for example, in a test environment.

PORTNUMBER(*port*)
Specifies, in the range 1 through 65535, the decimal number of the port on which CICS is to listen for incoming client requests.

The well-known ports are those from 1 through 1023. It is advisable to use well-known port numbers only for those services to which they are normally assigned. The well-known ports for services supported by CICS are:

80
HTTP (non-SSL)

443
HTTP with SSL

1435
ECI (Registered port number)

You should take care to resolve conflicts with any other servers on the same MVS image that might use the well-known ports.

Port sharing must be enabled for any port that you want to share across CICS systems within an MVS image. For more information, see z/OS Communications Server: IP Configuration Reference.

PRIVACY
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS. For more information, see Obsolete attributes.

PROTOCOL({HTTP|IIOP|ECI|IPIC|USER})
Specifies the application level protocol used on the TCP/IP port.

HTTP
HTTP protocol is used. HTTP protocol is handled by CICS Web support. CICS performs basic acceptance checks for messages sent and received using this protocol. This protocol is required for the well-known ports 80 (used for HTTP without SSL) and 443 (used for HTTP with SSL). HTTP requests might qualify for being processed by directly attached user transactions, and bypassing the web attach task, defined in the TRANSACTION attribute of TCPIPSERVICE. For more information, see HTTP requests are processed by directly attached user transactions.

IIOP
The IIOP attribute in TCPIPSERVICE definitions is obsolete but is retained for compatibility reasons.

ECI
The CICS ECI protocol is used.

IPIC
IPIC protocol is used. Specify IPIC for TCPIPSERVICE resources that are to be used for IP interconnectivity (IPIC) connections.

USER
The user-defined protocol is used. Messages are processed as non-HTTP messages. They are flagged as non-HTTP and passed unchanged to the analyzer program for the TCPIPSERVICE resource. CICS Web support facilities are used for handling the request, but no acceptance checks are carried out for messages sent and received by using this protocol. Processing for all non-HTTP requests must be carried out under the USER protocol, so that they are protected from the basic acceptance checks that CICS carries out for requests by using the HTTP protocol. If an HTTP message is handled by the USER protocol, you are responsible for checking its validity.

REALM(*string*)
Specifies the realm that is used for HTTP basic authentication. You can only specify this attribute for the HTTP protocol.

The realm is provided by CICS in the WWW-Authenticate header, and is seen by the user during the process of basic authentication. It identifies the set of resources to which the authentication information requested (that is, the user ID, and password or password phrase) applies.

If you do not specify a realm, the default used by CICS is `CICS application aaaaaaaa`, where *aaaaaaaa* is the APPLID of the CICS region.

The realm can be up to 56 characters, and can include embedded blanks. It is specified in mixed case, and the case is preserved. Do not specify opening and closing double quotation marks, as CICS provides these when assembling the WWW-Authenticate header.

Acceptable characters:

```
A-Z a-z 0-9 ¢ @ ♯ . - _ % & ? ! : | ' = ¬ + * , ; < > ( )
```

Space characters are also permitted. If parentheses are used, you must use them as pairs of opening and closing parentheses.

SOCKETCLOSE({NO|*hhmmss*})
Specifies whether, and for how long, CICS waits before it closes the socket. The SOCKETCLOSE attribute does not apply to the first receive request that is issued after a connection is made. On the first receive request for the ECI and USER protocols, CICS waits for data for 30 seconds before it closes the socket. On the first receive request for the HTTP protocol, CICS waits for the DTIMEOUT value associated with the transaction specified on the TCPIPSERVICE. If this DTIMEOUT value is zero, CICS waits for 30 seconds.

The interval is measured from the time of the initial receive request for incoming data on that socket.

NO
The socket remains open until it is closed by the client, or by a user application program in CICS.

hhmmss
The interval (in HHMMSS format) from the time of the initial receive request for incoming data, after which CICS is to timeout the socket. Choose a value that is appropriate to the responsiveness of the client, and the reliability of your network. Specifying 000000 closes the socket immediately if no data is available for any receive requests other than the first one.

If you are using this TCPIPSERVICE resource for CICS web support with the HTTP protocol, a zero setting for SOCKETCLOSE means that CICS closes the connection immediately after receiving data from the web client, unless further data is waiting. This setting means that persistent connections cannot be maintained, and it is not compliant with the HTTP/1.1 specification. Use a zero setting for SOCKETCLOSE with the HTTP protocol only if you have a special requirement for it in a CICS region that is not currently handling external requests, for example, in a test environment.

If you are using this TCPIPSERVICE resource for CICS web support with the HTTP protocol, the SOCKETCLOSE value of the TCPIPSERVICE affects how soon CICS closes HTTP persistent connections when a **SET TCPIPSERVICE OPENSTATUS(CLOSED)** command is issued. CICS attempts to close HTTP persistent connections as follows:

- When there are requests for HTTP persistent connections, CICS allows their next request to process. When the processing is complete, CICS sends out a close header to the client, and then closes the HTTP persistent connections.
- When there are no requests for HTTP persistent connections, CICS closes the HTTP persistent connections within 30 seconds or within the time as specified in the SOCKETCLOSE attribute of the TCPIPSERVICE if this value is less than 30 seconds.

If you specify PROTOCOL(ECI) or PROTOCOL(IPIC) you must specify SOCKETCLOSE(NO).

If you specify PROTOCOL(USER), persistent sessions are not supported, and you should specify SOCKETCLOSE(000000).

After the TCPIPSERVICE resource is installed, you cannot change this value using CEMT; you must set the TCPIPSERVICE resource out of service, and then re-install the TCPIPSERVICE resource with the modified definition.

SPECIFTCPS
This parameter specifies the 8-character name of the specific TCPIPSERVICE when defining a generic TCPIPSERVICE as part of a high-availability cluster. This parameter is only valid when the protocol is IPIC and the IP end point defined by the PORT and HOSTNAME or IPADDRESS attributes is the generic end point of an HA cluster of CICS regions.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters are converted to uppercase.

Note: If both the specific and the generic resources are being installed from a single resource group, you are advised to choose a name for the generic resource that alphabetically follows the name of the corresponding specific TCPIPSERVICE. This is because CICS installs resources of the same type in their alphabetical order when they are stored in a common resource group. If, as a result of the alphabetical order of its name, the generic resource is installed before the specific one that it references, then the generic resource is left in a closed state and you have to take action to resolve this every time that you install the group.

SSL({NO|YES|CLIENTAUTH|ATTLSAWARE})
Specifies whether the TCP/IP service is to use the secure sockets layer (SSL) for encryption and authentication. You can specify this attribute for the HTTP, USER, and IPIC protocols, but not for the ECI protocol. Note: When they use SSL, MAXSSLTCBs should be set to a value that is at least twice the number of IPIC connections that use SSL by the CICS region.

NO
SSL is not to be used. No security checks are applied when the connection is being acquired. No encryption is applied to outbound messages. SSL(NO) and PROTOCOL(HTTP) HTTP requests might qualify for being processed by directly attached user transactions, and bypassing the web attach task. For more information, see HTTP requests are processed by directly attached user transactions.

YES
An SSL session is to be used; CICS sends a server certificate to the client. SSL decryption processing is applied to all messages arriving at this port. The level of encryption that is applied to inbound messages is found from the value of the CIPHERS attribute.

CLIENTAUTH
An SSL session is to be used; CICS sends a server certificate to the client. CICS expects to receive a client certificate from the partner system during the SSL handshake, when the connection is being acquired.

ATTLSAWARE
CICS queries the client connection to determine whether AT-TLS is active. CICS retrieves a client certificate from TCP/IP if one was provided by the partner. ATTLSAWARE should be used with PROTOCOL(HTTP) TCPIPSERVICES. SSL(ATTLSAWARE) and PROTOCOL(HTTP) HTTP requests might qualify for being processed by directly attached user transactions, and bypassing the web attach task. For more information, see HTTP requests are processed by directly attached user transactions.

STATUS({OPEN|CLOSED})
Indicates the initial status of the service after installation. Set it to OPEN if CICS is to begin listening for this service after installation. Set to CLOSE if CICS is not to listen on behalf of this service after installation.

This attribute is ignored for TCPIPSERVICE resources that are dynamically generated by a CICS bundle. The initial status of a TCPIPSERVICE resource is derived from the initial status of the bundle that defines the resource.

TCPIPSERVICE(*name*)
Specifies the 8-character name of this service.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

TRANSACTION(*transaction*)
Specifies the 4-character ID of the CICS transaction attached to process new requests received for this service.

- For an ECI over TCP/IP TCPIPSERVICE resource, specify CIEP (or another transaction that executes program DFHIEP).
- For an HTTP TCPIPSERVICE resource, specify CWXN (or another transaction that executes program DFHWBXN). HTTP requests might qualify for being processed by directly attached user transactions, and bypassing the web attach task. For more information, see HTTP requests are processed by directly attached user transactions.
- For an IPIC TCPIPSERVICE resource, specify CISS (or another transaction that executes program DFHISCOP).
- For a USER TCPIPSERVICE resource, specify CWXU (or another transaction that executes program DFHWBXN).

TSQPREFIX(*prefix*)
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS. For more information, see Obsolete attributes.

URM({NO|*program_name*})
Specifies the name of a user-replaceable program to be started by this service.

NO
Autoinstall is not permitted with this TCPIPSERVICE resource. This is only applicable for PROTOCOL(IPIC).

program_name
For those protocols for which URM is a required attribute, the default program name depends upon the value of the PROTOCOL attribute:

- For the HTTP protocol, specify the name of an analyzer program to be associated with this TCPIPSERVICE resource. The CICS-supplied analyzer program DFHWBAAX is the default. DFHWBAAX provides basic error handling when all requests on the port should be handled by URIMAP definitions (for example, web service requests). If you must provide support for requests that are not handled by URIMAP definitions, the analyzer program specified for your TCPIPSERVICE resource should be the CICS-supplied sample analyzer program DFHWBADX or your own customized analyzer program. See Analyzer programs in Developing system programs for more information about analyzer programs.
- For the IPIC protocol, specify the name of the autoinstall user program for IPIC connections, if required. For PROTOCOL(IPIC), if you do not specify this attribute CICS uses the CICS-supplied default autoinstall user program, DFHISAIP.
- For the USER protocol, specify the name of an analyzer program to be associated with this TCPIPSERVICE definition. The analyzer program must be present, and it handles all requests on this protocol. The CICS-supplied sample analyzer program, DFHWBADX, is suitable. See the Analyzer programs in Developing system programs for more information about analyzer programs.

TCPIPSERVICE: interrelated attributes

The attributes and the values you can specify depend in some cases on which other attributes you have specified.

For TCPIPSERVICE resources:

- The values you can specify for the AUTHENTICATE attribute depend upon the value of the PROTOCOL attribute:

AUTHENTICATE	PROTOCOL(ECI) or PROTOCOL(IPIC)	PROTOCOL(HTTP) or PROTOCOL(USER)
NO	invalid	valid
BASIC	invalid	valid
CERTIFICATE	invalid	valid
AUTOREGISTER	invalid	valid
AUTOMATIC	invalid	valid

- If you specify PROTOCOL(HTTP) or PROTOCOL(USER), ATTACHSEC must be blank. You can specify the ATTACHSEC attribute only when you specify PROTOCOL(ECI).
- If you specify PROTOCOL(HTTP), the default for URM is the CICS-supplied default analyzer program DFHWBAAX.
- If you specify PROTOCOL(IPIC), the default for URM is the CICS-supplied default autoinstall program for IP connections, DFHISAIP.
- If you specify AUTHENTICATE(CERTIFICATE) or AUTHENTICATE(AUTOREGISTER), you must specify SSL(CLIENTAUTH) or SSL(ATTLSAWARE).
- If you specify a well known port number in the PORTNUMBER attribute, CICS sets the values of other attributes:

PORT	PROTOCOL	TRANSACTION	SSL
80	HTTP	CWXN	NO
443	HTTP	CWXN	YES
1435	ECI	CIEP	NO

- If you specify the PROTOCOL attribute, but not the TRANSACTION attribute, CICS sets the TRANSACTION attribute. Similarly, if you supply one of the following values in the TRANSACTION attribute, but not the PROTOCOL attribute, CICS sets the PROTOCOL attribute:

PROTOCOL	TRANSACTION
ECI	CIEP
HTTP	CWXN
IPIC	CISS
USER	CWXU

- You can change the value that CICS supplies for the TRANSACTION attribute. Depending on the value that you specify for the PROTOCOL attribute, some values are not permitted:

PROTOCOL	TRANSACTION (CWXN)	TRANSACTION (CIEP)	TRANSACTION (CWXU)	TRANSACTION (CISS)
HTTP	Default	Invalid	Invalid	Invalid
USER	Invalid	Invalid	Default	Invalid

PROTOCOL	TRANSACTION (CWXN)	TRANSACTION (CIEP)	TRANSACTION (CWXU)	TRANSACTION (CISS)
ECI	Invalid	Default	Invalid	Invalid
IPIC	Invalid	Invalid	Invalid	Default

- If you specify PROTOCOL(ECI) or PROTOCOL(IPIC) you must specify SOCKETCLOSE(NO).

TCPIPSERVICE resource definitions

TCPIPSERVICE resource definitions define which TCP/IP services are to use internal sockets support.

TDQUEUE resources

A TDQUEUE definition defines the attributes of a transient data queue.

The following transient data resources can be managed using RDO:

- Intrapartition
- Extrapartition
- Indirect
- Remote

Intrapartition definitions contain attributes that provide information about recovery characteristics, trigger levels, associated transactions, facilities, and userids.

Extrapartition definitions contain information about the associated QSAM data set, and the number of buffers that are to be used.

Indirect definitions identify the underlying queue name.

Remote definitions contain the name of the remote system and the name by which the queue is known on that remote system.

Before a transient data queue can be used by an active CICS system, you must install its definition in the running system. CICS uses the definition to access the data set associated with the queue, and records the number of read and write operations on the queue.

Remote transient data queues can be defined using the CEDA transaction in one of two ways:

- If the queue TYPE is omitted and data is entered into only the REMOTE ATTRIBUTES section of the definition, a remote definition will be created.
- If a TYPE of INTRA or EXTRA is specified and the REMOTE ATTRIBUTES section is completed, both a local and a remote resource will be established at the same time.

  See Figure 3 on page 155 for an example of defining a dual-purpose transient data resource definition.

Dual-purpose resource definition for transient data

You cannot specify TYPE=REMOTE for transient data queues.

Instead, you may want to consider dual-purpose resource definition (see Shared resources for intercommunication). Dual-purpose resource definition can be used with transient data definitions. Figure 3 on page 155 gives an example of dual-purpose resource definition for transient data resources.

If the definition shown in Figure 3 on page 155 is installed in a system called CICQ, that definition becomes a local intrapartition queue (the value of REMOTESYSTEM is CICQ).

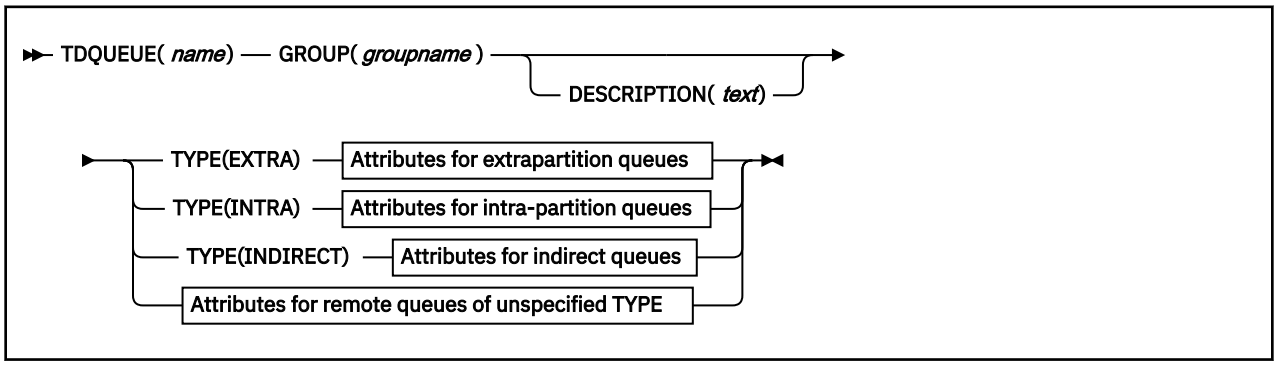
If the definition shown in Figure 3 on page 155 is installed in a system other than CICQ, the definition becomes a REMOTE queue.

```
 TDqueue        : TDQ1
 Group          : Example
 DEscription  ==>
 TYPE         ==> Intra                          Extra | INTra | INDirect
EXTRA PARTITION PARAMETERS
 DAtabuffers    :                                1-255
 DDname         :
 DSname         :
 Sysoutclass    :
 Erroroption    :                                Ignore | Skip
 Opentime       :                                Initial | Deferred
 REWind         :                                Leave | Reread
 TYPEFile       :                                Input | Output | Rdback
 RECORDSize     :                                0-32767
 BLOCKSize      :                                0-32767
 RECORDFormat   :                                Fixed | Variable
 BLOCKFormat    :                                Blocked | Unblocked
 Printcontrol   :                                A | M
 DIsposition  :                                  Shr | Old |Mod
INTRA PARTITION PARAMETERS
 Atifacility  ==> Terminal          Terminal | File | System
 RECOVstatus  ==> Logical           No | Physical | Logical
 Facilityid   ==> FR1
 TRAnsid      ==>
 TRIggerlevel ==> 00001             0-32767
 Userid       ==>

 INDOUBT ATTRIBUTES
 WAIT         ==> Yes                            Yes|No
 WAITAction   ==> Reject                         Queue|Reject

 INDIRECT PARAMETERS
 Indirectname :

 REMOTE PARAMETERS
 REMOTEName   ==> FR1
 REMOTESystem ==> CICQ
 REMOTELength ==>                                      0-32767
```

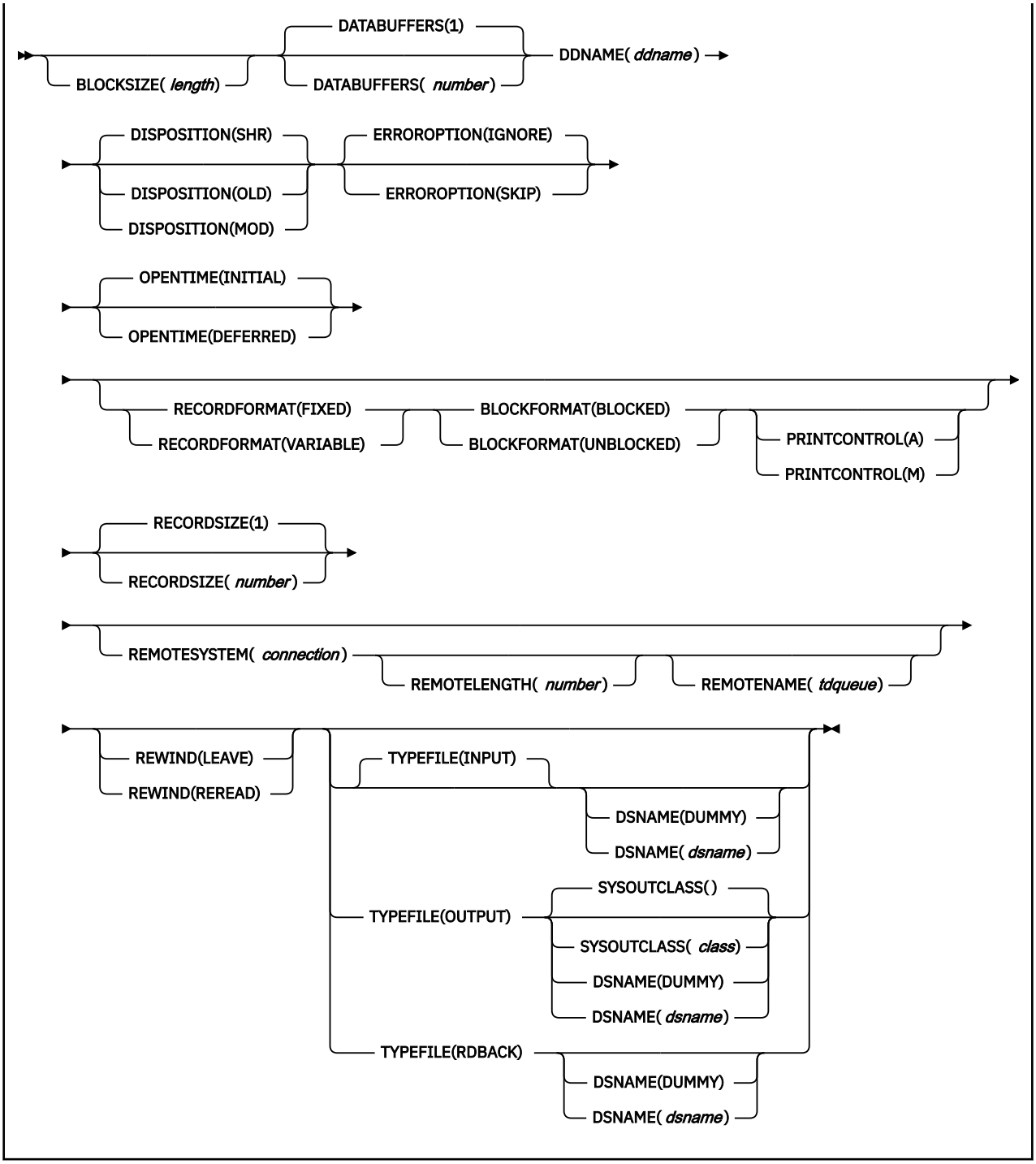
Figure 3. Dual-purpose resource definition for transient data

TDQUEUE attributes

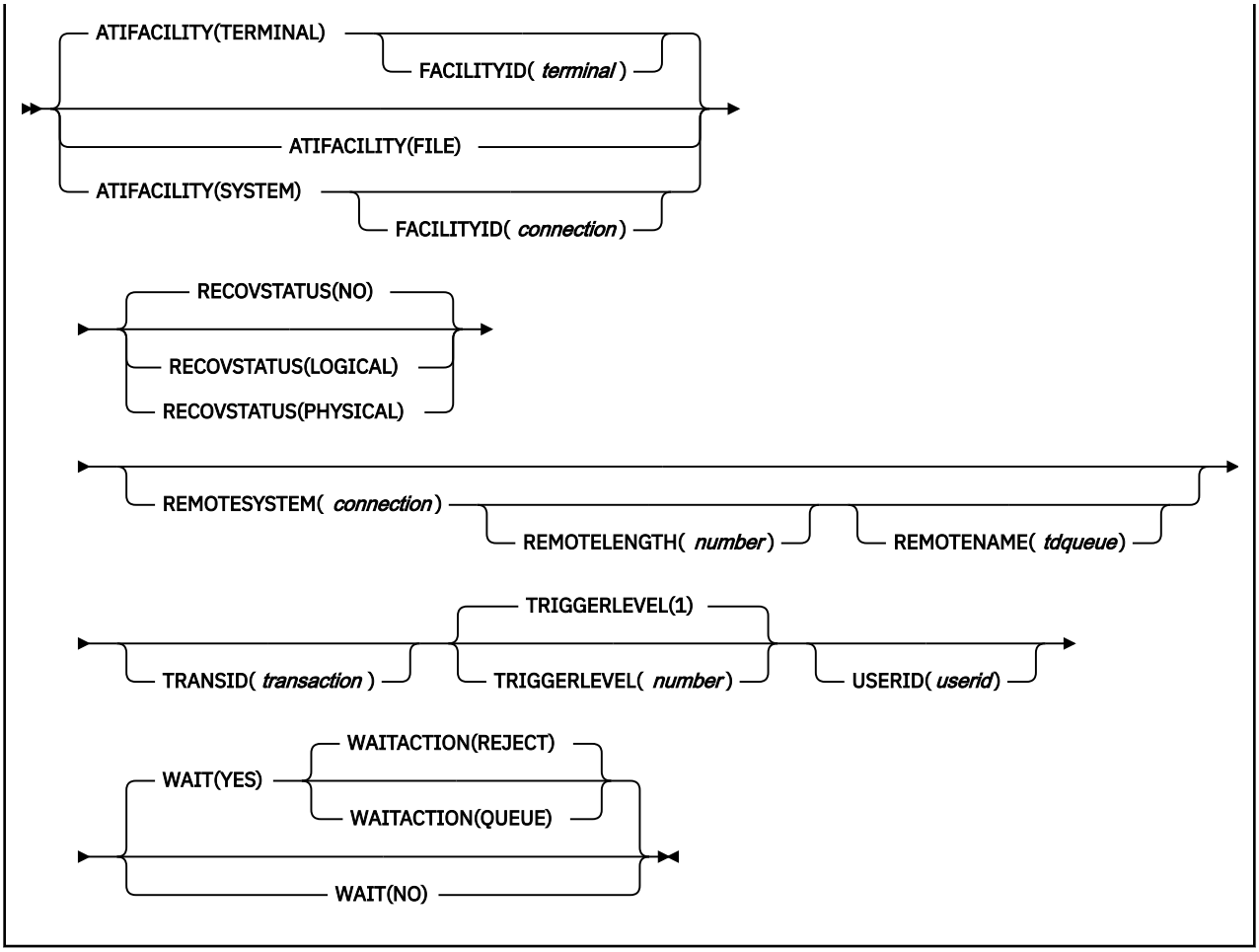
Describes the syntax and attributes of the TDQUEUE resource.



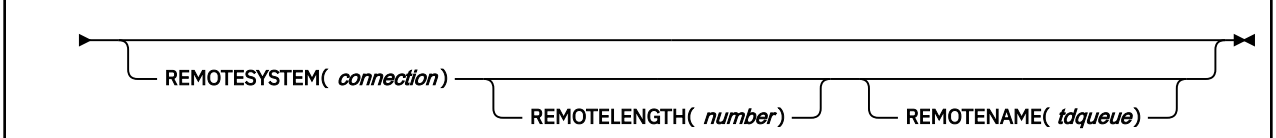
Attributes for extrapartition queues



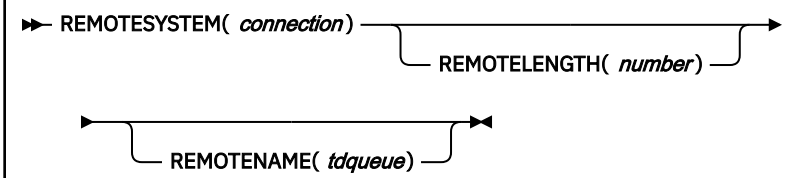
Attributes for intrapartition queues



TDQUEUE attributes for indirect queues



TDQUEUE attributes for remote queues of unspecified TYPE



ATIFACILITY({TERMINAL|FILE|SYSTEM}) (intrapartition queues only)
Specifies the type of destination the queue represents.

FILE
The transient data queue is to be used as a file of data records that are not associated with a particular terminal or system. ATI does not require a terminal to be available.

SYSTEM
The transient data queue is to be associated with the specified system identifier. The system must be defined to the local CICS system using an RDO CONNECTION definition.

Specifying ATIFACILITY(SYSTEM) initiates a distributed transaction processing (DTP) session. For more information about DTP considerations in application programming, see Distributed transaction processing (DTP).

TERMINAL
The transient data queue is to be associated with the terminal. The terminal must be defined to CICS. If you do not specify TERMINAL, it defaults to the value of FACILITYID. If ATI is used, as specified in the TRANSID and TRIGGERLEVEL attributes, the transaction that is initiated is associated with the specified terminal, which must be available before the transaction can be initiated.

BLOCKFORMAT({BLOCKED|UNBLOCKED|*blank*}) (extrapartition queues only)
Specifies the block format of the data set. There is no default. If you specify the record format (RECORDFORMAT attribute) as undefined (or allow it to default), you cannot specify anything for the BLOCKFORMAT attribute.

blank
Indicates that no block format is defined for this data set. Leave this field blank if you leave RECORDFORMAT blank.

BLOCKED
Blocked record format.

UNBLOCKED
Unblocked record format.

You are strongly advised to specify an unblocked record format for extrapartition queues that are used as an interface to the JES internal reader. If you use a blocked record format, your job is held in the SYSOUT data set, and not sent directly to JES until one of the following actions occurs:

- You follow the JES /*EOF control statement with a second /*EOF statement
- Your application writes another job to the same queue
- You explicitly close the queue after the job is written
- You shut down CICS normally

BLOCKSIZE({length}) (extrapartition queues only)
Specifies the length of the block in bytes, in the range 0 through 32767.

The maximum value you can specify depends on whether SYSOUTCLASS is specified, either explicitly or by default, and on whether RECORDFORMAT is FIXED or VARIABLE.

- If you specify SYSOUTCLASS, the maximum value of RECORDSIZE is 8968.
- Each block in a variable format data set consists of a block descriptor word followed by one or more logical records. Therefore, if you specify RECORDFORMAT(VARIABLE), the value you specify for BLOCKSIZE must include 4 bytes for the block descriptor word, and space for the largest possible logical record.

These limits are summarized in the following table:

Is SYSOUTCLASS specified?	Yes	Yes	No	No
RECORDFORMAT	VARIABLE	FIXED	VARIABLE	FIXED
Maximum value of RECORDSIZE	8968	8968	32763	32767
Maximum value of BLOCKSIZE	8972	8968	32767	32767

DATABUFFERS({1|*number*}) (extrapartition queues only)
Specifies the number of buffers to be provided, up to a maximum of 255.

DDNAME(*ddname*)
Specifies a 1- to 8-character value that may refer to a data set defined in the startup JCL. The name must not start with the characters "DFH", which are reserved for use by CICS, unless the name describes one of the standard destinations.

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

DISPOSITION({SHR|OLD|MOD}) (extrapartition queues only)
Specifies the disposition of the data set.

MOD
CICS first assumes that the data set exists. For an existing sequential data set, MOD causes the read/write mechanism to be positioned after the last record in the data set. The read/write mechanism is positioned after the last record each time the data set is opened for output.

If CICS cannot find volume information for the data set:

- On the DD statement
- In the catalog
- Passed with the data set from a previous step

it assumes that the data set is being created in this job step. A data set allocated dynamically in this way is deleted when the queue is closed, and all records are lost.

For a new data set, MOD causes the read/write mechanism to be positioned at the beginning of the data set.

OLD
The data set existed before this job step.

SHR
The data set existed before this job step and can be read by other concurrent jobs.

DSNAME({*dsname*|DUMMY}) (extrapartition queues only)
Specifies the name of the QSAM data set that is to be used to store records written to this extrapartition queue.

When CICS receives a request to open an extrapartition transient data queue, the startup JCL is referenced to check if a data set definition has been created. If one is not found, the 44-character name specified on the DSNAME attribute is used to dynamically allocate the required data set.

If you have JCL that preallocates for this queue the DSCNAME value to a DSNAME value, the DSNAME value in the resource definition is overridden by the DSNAME value from the JCL. JCL allocation always takes priority.

Partitioned data sets (PDS) are not supported on the DSNAME attribute. If you want to use a PDS member for an extrapartition queue data set, code it explicitly in your JCL. Bear in mind that if you inquire upon this queue, the DSNAME value that is returned will not give you any indication of the member name.

DUMMY
A dummy data set name.

name
The 44-character name of a physical data set.

Attention: If you use log streams for extrapartition queue data sets, unpredictable results may occur.

ERROROPTION({IGNORE|SKIP}) (extrapartition queues only)
Specifies the action to be taken if an I/O error occurs. This can be one of the following:

IGNORE
The block that caused the error is accepted.

SKIP
The block that caused the error is skipped.

FACILITYID(*terminal|connection*) (intrapartition queues only)
Specifies a 4-character field that contains either:

- The system identifier for an intrapartition queue that specifies ATIFACILITY(SYSTEM)
- The terminal identifier where ATIFACILITY(TERMINAL) is specified.

If you do not specify anything in the FACILITYID field, it defaults to the name of the queue in each case.

If ATIFACILITY(FILE) is specified, the FACILITYID field must remain blank.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

INDIRECTNAME(*tdqueue*) (indirect queues only)
Specifies the name of the transient data queue to which this indirect queue routes data. The transient data queue must be defined to CICS, and it can be intrapartition, extrapartition, remote, or indirect.

OPENTIME({INITIAL|DEFERRED}) (extrapartition queues only)
Specifies the initial status of the data set. The initial status can be one of the following values:

DEFERRED
The data set remains closed until you indicate that you want to open it by using the CEMT INQUIRE|SET TDQUEUE command.

INITIAL
The data set is to be opened at install time. However, if the DSNAME attribute is not specified, and the data set name is not specified in the DD statement in the startup JCL, the transient data queue is allocated to JES during CICS startup.

PRINTCONTROL({ASA|MACHINE|*blank*}) (extrapartition queues only)
Specifies the control characters to be used. There is no default.

If you allow RECORDFORMAT to default to blank, you cannot specify anything in the PRINTCONTROL field. You can use the following characters:

ASA
ASA control characters.

blank
No control characters are to be used.

MACHINE
Machine control characters.

RECORDFORMAT({FIXED|VARIABLE|blank) (extrapartition queues only)
Specifies the record format of the data set.

blank
If RECORDFORMAT is not specified (that is, remains blank), the BLOCKFORMAT and PRINTCONTROL fields must also remain blank. If the RECORDFORMAT is not specified in the

resource definition, TD attempts to derive this attribute from the CICS startup JCL or from the QSAM data set definition at the time it attempts to open the queue. The open request will fail if this information cannot be derived from either of these sources.

FIXED
Fixed records. If you specify RECORDFormat(Fixed), you must also specify a block format.

VARIABLE
Variable records. If you specify RECORDFormat(Variable), you must also specify a block format.

RECORDSIZE({1|*number*}) (extrapartition and remote queues)
Specifies the record length in bytes, in the range 0 through 32767.

1
The default record length is 1 byte.

number
The record length, in bytes, up to 32767.

The maximum value you can specify depends on whether SYSOUTCLASS is specified, either explicitly or by default, and on whether RECORDFORMAT is FIXED or VARIABLE.

- If you specify SYSOUTCLASS, the maximum value of RECORDSIZE is 8968.
- For V format data sets, each logical record consists of a record descriptor followed by a data record. The value entered in the RECORDSIZE field must include 4 bytes for the record descriptor word (or LLBB), and should allow for the largest possible data record.

These limits are summarized in the description of the BLOCKSIZE attribute.

If the TDQUEUE is defined with RECORDSIZE(0), the record length is taken from the dataset DCB LRECL parameter for either input or output queues operations.

If the TDQUEUE is defined with a non-zero RECORDSIZE, the LENGTH parameter for the **EXEC CICS READQ** or the **EXEC CICS WRITEQ** command must equal the RECORDSIZE value, and the underlying DCB LRECL value is overridden.

RECOVSTATUS({NO|PHYSICAL|LOGICAL}) (intrapartition queues only)
Specifies the recoverability attributes of the queue in the event of an abnormal termination of either CICS or the transaction that is processing the queue. The recoverability attributes are:

LOGICAL
This queue is logically recoverable. Automatic logging is to be performed to keep track of accesses by application programs. If a transaction that accessed this queue was in-flight at the time of abnormal termination, or in the subsequent emergency restart or dynamic transaction backout, the queue is restored to the status it was in before the in-flight UOW modified it.

When this queue is accessed, the task that issued the DELETEQ TD, WRITEQ TD, or READQ TD command is enqueued on the input, the output, or both ends of the transient data queue. The enqueue is maintained until the task terminates (or issues a syncpoint request to signal the end of a UOW) to ensure the integrity of the data being accessed. This means that enqueues can be maintained for a longer time, and can result in a queue lockout if an application program accessing the queue performs more than one UOW against the queue without defining each separate UOW to CICS by issuing a syncpoint request.

Furthermore, when a DELETEQ request is issued for a logically recoverable queue, both the input and output ends of the queue are enqueued upon. This can increase the possibility of an enqueue lockout.

Note: CICS provides an enqueuing protection facility for logically recoverable (as distinct from physically recoverable) TD queues similar to that for recoverable files. However, CICS regards each logically recoverable destination as two separate recoverable resources—one for writing and one for reading.

In the case of a file record, a record is treated as a single resource and requires only one lock. The TD queue, on the other hand, has two 'ends'—the read end and the write end, and these can be enqueued on (locked) independently. This is because, to control both reading and writing from

the TD queue (at the same time), CICS has to maintain two pointers (cursors)—one for reading and one for writing, and these need to be protected from conflicting transactions.

Queue records are held on one or more control intervals (CIs). Each CI is marked for release as soon as the last record on it has been read. However, the release does not occur until the end of task, or until after the next user syncpoint.

NO
This queue is not recoverable. Automatic logging is not performed to keep track of accesses to this queue. Queue records are held on one or more control intervals (CIs). Each CI is released as soon as the last record on it has been read.

PHYSICAL
This queue is physically recoverable. Automatic logging is to be performed to keep track of accesses by application programs. If emergency restart occurs, this queue is to be recovered to its status at the time CICS terminated.

The queue is **not** recovered to its status at the time CICS terminated if the last action on the queue was a READQ request, and if the associated unit of work (UOW) did not commit the changes. On emergency restart, the last read operation is backed out, and appears never to have taken place.

Queue records are held on one or more control intervals (CIs). Each CI is released as soon as the last record on it has been read.

REMOTENAME(*tdqueue*) (remote queues only)
Specifies, if the transient data queue resides on a remote system, the 4-character name by which the queue is known in the system or region on which the queue resides.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

REMOTELENGTH({1|*number*}) (remote queues only)
Specifies the length in bytes, in the range 1 through 32767.

For SYSOUT data sets, the value entered in the REMOTELENGTH field must not be greater than 8968 bytes (when the SYSOUTCLASS attribute has been specified).

1
The length is 1 byte.

number
The length in bytes, up to 32767.

If the queue is defined with TYPE=EXTRA, and no value is specified for REMOTELENGTH, the value on the RECORDSIZE attribute is used at installation time.

REMOTESYSTEM(*connection*)
Specifies the 4-character alphanumeric name of the system or region in which the remote transient data queue resides. The name entered must be the same as the name specified on the RDO CONNECTION or the first 4 characters of the name specified on an RDO IPCONN definition. For more information about the connection definitions, see “CONNECTION attributes” on page 9 or “IPCONN attributes” on page 66.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

When the transient data queue definition is installed, the name entered in the REMOTESYSTEM attribute is compared with the system identifier. If the names are different, the system or region is

remote. If the names are the same, the value specified in the TYPE attribute is used. If the TYPE attribute is blank, the installation fails.

REWIND({LEAVE|REREAD}) (extrapartition queues only)
Specifies the disposition of a tape data set. The disposition can be one of the following:

LEAVE
The current tape is positioned at the logical end of the data set.

REREAD
The current tape is positioned at the logical start of the data set.

SYSOUTCLASS({blank|A..Z|0..9|*}) (extrapartition queues only)
Instead of allocating an extrapartition queue to a physical data set, you can allocate it to a system output data set (referred to as SYSOUT). Use the SYSOUTCLASS attribute to specify the class of the SYSOUT data set.

blank
This is the default class.

A..Z|0..9
A single alphabetic or numeric character that represents an output class that has been set up on the z/OS system on which the CICS job is to run.

Acceptable characters:

```
A-Z 0-9
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

*****
The class of the SYSOUT data set defaults to the class that is specified on the MSGLCASS parameter of the JOB statement for the CICS job.

You can use SYSOUTCLASS as an alternative to DSNAME. As with DSNAME, the queue may already be preallocated to SYSOUT using a JCL DD statement. A JCL DD statement overrides any specification made using the TDQUEUE resource definition.

When CICS receives a request to open an extrapartition transient data queue, the startup JCL is referenced to check if a data set definition has been created. If one is not found, the 44-character name specified on the DSNAME attribute is used to dynamically allocate the required data set.

When SYSOUT is specified for a queue in the JCL, attributes other than class can also be specified (for example, form types).

Note: Specifying SYSOUT data sets using RDO supports the class parameter only. If you require other parameters, you specify SYSOUT data sets in the JCL.

For more information about SYSOUT and its associated classes, see the z/OS MVS JCL User's Guide.

TDQUEUE(*name*)
Specifies the 1- to 4-character name of a transient data queue.

Acceptable characters:

```
A-Z a-z 0-9 $ @ ♯ . / - _ % & ? ! : | " = ¬ , ; < >
```

If the name supplied is fewer than four characters, it is left-justified and padded with blanks up to four characters.

Note:

1. If you use a comma (,) in a name, you will be unable to use those commands such as

```
CEMT INQUIRE TDQUEUE(value1,value2)
CEMT SET     TDQUEUE(value1,value2)
```

where the comma serves as a list delimiter. See for information about using lists of resource identifiers.

2. If you protect your transient data queues using RACF, avoid using % and & in the name. RACF commands assign a special meaning to these characters when they are used in a profile name. See .

TRANSID(*transaction*) (intrapartition queues only)
Specifies the name of the transaction that is to be automatically initiated when the trigger level is reached. Transactions are initiated in this way to read records from the queue. If the TRANSID attribute is not specified (or if TRIGGERLEVEL(0) is specified), you must use another method to schedule transactions to read records from transient data queues.

The transaction specified must not reside in a remote CICS system. If it does, transaction initiation fails and a warning message is issued to the console.

TRIGGERLEVEL({1|*number*}) (intrapartition queues only)
Specifies the number of records to be accumulated before a task is automatically initiated to process them. (This number is known as the trigger level.)

If you specify the TRANSID attribute, TRIGGERLEVEL defaults to 1. Specify a trigger level of 0 if you want to disable ATI processing. If you do not specify a transaction id, the trigger level is ignored.

If you have specified ATIFACILITY(TERMINAL), the task is not initiated until the specified terminal is available. If you have specified ATIFACILITY(FILE), a terminal is not necessary for the task to be initiated.

If, at maximum task, a short-on-storage or a no-space condition exists for a nonterminal destination, the task is not initiated. This is also true during stages 1 and 2 of initialization, and during the final stage of shutdown. The task is initiated when the stress condition no longer exists, and a subsequent TD WRITE occurs.

For logically recoverable transient data queues, the ATI task is not attached until the task commits forward. This may mean that the trigger level is far exceeded before ATI occurs.

If a z/OS Communications Server terminal is defined as the destination for the CSTL transaction on two ISC CICS systems with a trigger level of 1, a performance problem may arise when both systems repeatedly acquire and release the terminal to write the session-started and session-ended messages.

You can change the trigger level when CICS is running using the CEMT transaction. If you reduce the trigger level to a number that is equal to (or less than) the number of records accumulated so far, the task is initiated when the next record is successfully put on the queue.

1
Only one record can accumulate.

number
The number of records that can accumulate (up to a maximum of 32767) before ATI occurs.

TYPE({EXTRA|INTRA|INDIRECT})
Specifies the following types of transient data queue:

EXTRA
A queue that is outside the CICS region is allocated to CICS.

Extrapartition queues are used for:

- Sending data outside the CICS region: for example, data created by a transaction for processing by a batch program.
- Retrieving data from outside the region: for example, data received from terminals as input to a transaction.

Extrapartition data is sequential and is managed by QSAM.

INDIRECT
An indirect queue is a queue that does not point to an actual data set, but to another queue. An indirect queue can be extrapartition, intrapartition, remote, or even another indirect queue.

For example, you can give a different symbolic name, INDIRECTDEST, to each of several different message types. You can then send all these message types to the same physical queue (INDIRECTDEST), or to different physical queues.

The DFH$TDWT sample program demonstrates how you can use indirect queues to send different categories of message to the same terminal. For programming information about DFH$TDWT, see Using the transient data write-to-terminal program (DFH$TDWT). DFH$TDWT sample definitions are given in the *CICS/ESA 4.1 Sample Applications Guide*.

If the QUEUE operand of an EXEC CICS WRITEQ TD, EXEC CICS READQ, or EXEC CICS DELETEQ command specifies an indirect queue, access is determined by the security setting of the final target queue.

INTRA
A queue for data that is to be stored temporarily.

An intrapartition destination can be a terminal, a file, or another system. A single data set, managed by VSAM, is used to hold the data for all intrapartition queues.

You can specify a transaction to process the records and a trigger level for each intrapartition queue. The trigger level represents a number of records that are allowed to accumulate before the specified transaction is initiated. See the description of the TRIGGERLEVEL attribute for more information about trigger levels.

The intrapartition queue can be defined as logically recoverable, physically recoverable, or not recoverable.

A logically recoverable queue is restored (after an individual transaction failure or a total system failure) to the status it had at the end of the last completed unit of work (UOW). (A UOW begins at start of task or at a syncpoint, and ends at end of task or at a syncpoint).

Physically recoverable queues are restored (after a total system failure) to the statuses they had when the system failure occurred.

TYPE=REMOTE cannot be specified on the TDQUEUE resource definition. If you want to define a remote transient data queue, leave the TYPE attribute blank and specify values for the remote attributes, REMOTELENGTH, REMOTENAME, and REMOTESYSTEM. Alternatively, you can include the remote attributes as part of the resource definitions for the other transient data queue types.

TYPEFILE({INPUT|OUTPUT|RDBACK})
Specifies the type of data set the queue is to be associated with.

INPUT
An input data set.

OUTPUT
An output data set.

RDBACK
An input data set that is to be read backward.

Note: This is appropriate only for data sets that have been defined on magnetic tape.

An extrapartition queue can be input or output, but not both.

For more information about the DCB macro fields, see z/OS DFSMS Macro Instructions for Data Sets.

USERID(*userid*) (intrapartition queues only)
Specifies the userid you want CICS to use for security checking when verifying the trigger-level transaction specified in the TRANSID field. The userid can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

The value entered in the USERID field is valid only when ATIFACILITY(FILE) is also specified.

When security is active, the trigger-level transaction runs under the authority of the specified userid. This userid must be authorized to all the resources used by the trigger-level transaction.

If you omit the userid from a transient data queue definition, CICS uses the CICS default userid, specified on the DFLTUSER system initialization parameter. Security checking takes place when you are installing an intrapartition definition containing a userid. If the security check fails, the resource definition for that intrapartition queue is not installed.

For further information about surrogate user security, see Transient data trigger-level transactions.

WAIT({YES|NO}) (intrapartition queues only)
Specifies whether an indoubt unit of work (UOW) that has modified a logically recoverable queue should wait for resynchronization with its coordinator to determine whether to commit or back out the changes.

NO
The UOW is not to wait. Any changes made to recoverable resources are to be backed out or committed, as specified by the ACTION attribute on the TRANSACTION resource definition.

YES
The UOW is to wait, and any action required while waiting is determined by the WAITACTION attribute.

This attribute overrides the WAIT attribute defined on the UOW's transaction definition. See Table 4 on page 210 for an explanation of the interactions of indoubt attributes on the TDQUEUE and TRANSACTION definitions.

WAITACTION({REJECT|QUEUE}) (intrapartition queues only)
Specifies the action CICS is to take for an indoubt unit of work (UOW) if the definition for this queue specifies WAIT(YES). The possible actions are:

QUEUE
The UOW is indoubt and waiting; any locks held by the UOW for this queue remain active until the final state of the UOW is known. This means that tasks are suspended rather than receiving the LOCKED response. When the final state of the UOW is known, any changes that it has made are committed or backed out. Until then, any further requests of the following types that need one of the active locks must wait:

- READQ, if the indoubt UOW had issued READQ or DELETEQ requests.
- WRITEQ, if the indoubt UOW had issued WRITEQ or DELETEQ requests.
- DELETEQ, if the indoubt UOW had issued READQ, WRITEQ or DELETEQ requests.

REJECT
The UOW is indoubt and is waiting. Any lock held by the UOW for this queue is retained until the final state of the UOW is known. When the final state is known, any changes the UOW has made are committed or backed out. Until then, any further request that needs one of the retained locks is rejected, and a LOCKED response is returned. WAITACTION=REJECT causes LOCKED to be raised in exactly the same circumstances as those in which QUEUE causes a transaction to wait.

Required TDQUEUE definitions

CICS services use certain transient data queues, and the associated TDQUEUE definitions must be installed as soon as possible during a cold start.

To ensure that the required TDQUEUE definitions are installed as soon as possible, specify the **GRPLIST** system initialization parameter in one of the following ways:

- Specify DFHLIST as the first list in **GRPLIST**. The DFHLIST list contains the group DFHDCTG, which contains the relevant TDQUEUE definitions.
- Specify that group DFHDCTG is the first group in the first list in **GRPLIST**.

The transient data queues in group DFHDCTG are as follows:

CADL (needed to log z/OS Communications Server resource definitions)
For z/OS Communications Server resources, this destination keeps a log of each RDO definition installed in the active CICS system. The log records both the installation of entries in the TCT, and the deletion of autoinstalled entries from the TCT. It records definitions installed:

- By autoinstall
- Using CEDA INSTALL
- At system initialization

For details about defining data sets for transient data queues, see Defining data sets.

CADO (needed for CICS Development Deployment Tool messages)
CADO is the transient data queue that logs the messages from the CICS Development Deployment Tool.

CADS (needed for messages produced by auditing system programming interface commands)
CADS is the transient data queue that logs the messages produced when auditing the system programming interface commands.

CAIL (needed to log autoinstall terminal model definitions)
The autoinstall terminal model manager (AITM) uses this destination to log all autoinstall terminal model entries installed in, and deleted from, the TCT.

CCPI (needed for CPI Communications messages)
The common programming interface for communications (CPI Communications) writes messages to this destination.

CCSE (needed for C language support)
CICS directs the C standard streams to transient data queues. (Queue names are fixed in CICS: thus, the C standard streams cannot be redirected to other queues.) C programs write to the CCSE queue by writing to stderr. You can code this destination as extrapartition, intrapartition, or indirect. If you do not provide a TDQUEUE definition for the CCSE queue, writing to stderr in C programs fails.

CCSI (optional, for C language support)
The CCSI queue is reserved for stdin, the C standard stream for input data. Although the CCSI queue name is reserved for stdin, any attempt to read from stdin in CICS results in EOF being returned. For this reason, this destination is optional. You can code it as extrapartition, intrapartition, or indirect.

CCSO (needed for C language support)
CCSO is associated with stdout, the C standard stream for output data. You can code this destination as extrapartition, intrapartition, or indirect. If you do not define the CCSO queue, writing to stdout in C programs fails.

CCZM (needed for CICS class messages)
CCZM is the transient data queue that logs the messages from the CICS foundation classes.

CDBC (needed for DBCTL DFHDB81xx messages)
CDBC is defined as an indirect queue, which points to the CSML extrapartition queue. Only DBCTL DFHDB81xx messages use this data log; other messages use either the terminal or the console.

CDB2 (needed if DB2 is used)
CDB2 is the transient data queue for messages and statistics from the CICS DB2 attachment facility.

CDEP (needed for deprecation messages)
CDEP is the transient data queue for deprecation messages.

CDUL (needed for transaction dump messages)
CDUL is the destination for transaction dump messages. If a transaction dump is requested, for example after a transaction abend, a message is written to this destination to show that a dump has been taken or to give a reason why the dump was suppressed.

CECO (needed for event capture and emission messages)
CECO is the transient data queue for messages from the event capture and emission (EC) component.

CEJL (needed for Enterprise Java messages)
CEJL is the transient data queue for messages from the Enterprise Java domain.

CEPO (needed for event processing messages)
CEPO is the transient data queue for messages from the event processing (EP) domain.

CESE (needed for runtime output from Language Environment)
For Language Environment, all runtime output is written to this transient data queue. For further information, see the z/OS Language Environment Programming Guide.

CESO (needed for runtime output from Language Environment)
For Language Environment, all runtime output is written to this transient data queue. For further information, see the z/OS Language Environment Programming Guide.

CIEO (needed for IP ECI messages)
CIEO is the transient data queue for Internet Protocol (IP) external call interface (ECI) messages from the IP ECI (IE) domain.

CISL (needed to log IPCONN resource definitions)
This destination provides a log of installed IPCONN resource definitions. During early CICS initialization, if CISL is not yet available messages intended for this destination can be routed to the console.

CISO (needed for IPIC messages)
CISO is the transient data queue for IP interconnectivity (IPIC) connection messages from the inter-system (IS) domain.

CJRM (needed for JRas logging and tracing facility messages)
CJRM is the transient data queue for messages from the JRas logging and tracing facility. These are messages that are issued by a Java class running in a CICS JVM that are not CICS messages.

CKQQ (needed for CICS-MQ Connection messages)
CKQQ is the transient data queue that logs the CICS-MQ messages that are issued during CICS-MQ adapter startup and shutdown. It is defined as an intrapartion queue.

CMIG (needed for migration log)
CMIG is a **migration log**, which receives messages reporting the use of functions that are no longer supported in CICS (for example, the EXEC CICS ADDRESS CSA command). You can define CMIG as an intrapartition, extrapartition, or indirect destination.

CMLO (needed for markup language messages)
CMLO is the transient data queue for messages from the markup language domain.

CMPO (needed for managed platform messages)
CMPO is the transient data queue for messages from the managed platform domain.

CMQM (needed for CICS-MQ messages)
CMQM is the queue that logs all CICS-MQ messages that are issued by the CICS-MQ adapter, CICS -MQ trigger monitor,and CICS-MQ bridge.

CPIO (needed for CICS SOAP messages)
CPIO is the transient data queue that logs the CICS SOAP messages.

CRDI (needed to log program resource definitions)
This destination provides a log of installed resource definitions, such as programs, transactions, maps, and mapsets.

CRLO (needed for resource lifecycle messages)
CRLO is the transient data queue for messages from the resource lifecycle manager.

CRPO (needed for Open Network Computing Remote Procedure Call (ONC RPC) messages)
CRPO is defined as an extrapartition queue, but you can make the destination intrapartition or indirect.

CSBA (needed if using the business application manager (BAM))
CSBA is the transient data queue for messages from the business application manager domain.

CSBR (needed if using the bridge facility)
CSBR is the transient data queue that logs the bridge facility messages.

CSCC (needed for CICS client error messages)
CSCC is the transient data queue that logs the CICS client error messages.

CSCS (needed for the sign-on transaction)
CSCS receives a message giving details of each sign-on and sign-off. It also receives a message about each rejected attempt at sign on and each resource authorization failure. This destination can be of any type.

CSDH (needed if using the document handler template)
CSDH is the transient data queue that receives explanatory messages from a document handler template exit program.

CSDL (needed to log RDO commands)
The resource definition online (RDO) transactions write to this destination all commands that result in changes to the CICS system definition (CSD) file or active CICS system.

You need CSDL only if you use RDO and want to keep a log of commands.

The maximum length of data records written to CSDL is 128 bytes. If you define CSDL as extrapartition, the associated SDSCI or DD statement should specify V format records with a minimum blocksize of 136 bytes.

CSFL (needed to log file resource definitions)
CSFL is a log of all file resource definitions installed in the active CICS system. Deletions of file resource entries are also logged here.

CSJE and CSJO (needed for redirected output from Java programs that execute in a JVM)
CSJE and CSJO receive output from JVMs that has been intercepted by the CICS-supplied sample class com.ibm.cics.samples.SJMergedStream. This sample class can optionally be specified by the USEROUTPUTCLASS option in a JVM profile. CSJE is for stderr output, internal messages, and unresettable event logging, and CSJO is for stdout output. The com.ibm.cics.samples.SJMergedStream class handles both types of output, and directs them to the correct transient data queue. CSJE and CSJO are defined as indirect queues that point to the CSSL queue.

The length of messages issued by the JVM can vary, and the maximum record length for the CSSL queue (133 bytes) might not be sufficient to contain some of the messages you receive. If this happens, the sample output redirection class issues an error message, and the text of the message might be affected. If you find that you are receiving messages longer than 133 bytes from the JVM, you should redefine CSJO and CSJE as separate transient data queues. Make them extrapartition destinations, and increase the record length for the queue. You can allocate the queue to a physical data set or to a system output data set. You might find a system output data set more convenient in this case, because you do not then need to close the queue in order to view the output. If you redefine CSJO and CSJE, ensure that they are installed as soon as possible during a cold start, in the same way as for transient data queues that are defined in group DFHDCTG.

CSKL (needed to log transaction and profile resource definitions)
CSKL is a log of all transaction and profile resource definitions installed in the active CICS system. Deletions are also logged here.

CSLB (needed to log LIBRARY resource definitions)
CSLB is a log of all changes to the dynamic LIBRARY resources installed in the active CICS region. That is, install and discard of LIBRARY resources, and any changes to the configuration of LIBRARY resources, including changes to the enablement, ranking or critical status of a LIBRARY. If no dynamic LIBRARY resources are installed, no audit logging to CSLB takes place.

CSML (needed for the sign-off transaction)
CICS sign-off writes data to this destination.

CSMT (needed for terminal error and abend messages)
The terminal abnormal condition program (DFHTACP) and abnormal condition program (DFHACP) write terminal error and ABEND messages, respectively, to this destination. You can code this destination as extrapartition, intrapartition, or indirect.

CSNE (needed for node error messages)
The node abnormal condition program (DFHZNAC) and the node error program (DFHZNEP) write terminal error messages and data to this destination. You can code this destination as extrapartition, intrapartition, or indirect.

CSOO (needed for sockets domain messages)
CSOO is the transient data queue for messages from the sockets domain.

CSPL (needed to log program resource definitions)
CSPL is a log of all program resource definitions installed in the active CICS system. Deletions are also logged here.

CSQL (needed for transient data queue messages)
CSQL is the transient data queue that receives audit log messages from the transient data queue manager itself.

CSRL (needed to log partner resource definitions)
CSRL is a log of all PARTNER resources installed in the active CICS system. Deletions are also recorded here.

CSSH (needed for scheduler services)
CSSH is the transient data queue that receives messages from the scheduler services.

CSSL (needed for recovery utility statistics)
The recovery utility program (DFHRUP) writes statistics to this destination. This destination needs a minimum logical record length of 136 bytes and a minimum blocksize of 140 bytes.

CSTL (needed for terminal I/O error messages)
The terminal abnormal condition program (DFHTACP) writes terminal I/O error messages to this destination. You can code this destination as extrapartition, intrapartition, or indirect.

CSZL (needed for the Front End Programming Interface)
CSZL is the transient data queue that logs FEPI messages.

CSZX (needed for the Front End Programming Interface)
CSZX is the transient data queue that logs unexpected events for FEPI.

CWBO (needed for CICS web support messages)
CWBO is the transient data queue that logs the CICS web support messages that are issued from the CICS Web Interface.

CWBW (needed for CICS web warning messages)
CWBW is the transient data queue that logs the HTTP warning headers on messages that are received by CICS web support.

Installing transient data queue definitions

After CICS has been initialized, you can install additional resources using the CEDA transaction, or the EXEC CICS CREATE commands.

About this task

A transient data queue is **always** installed in an enabled state. Any queues that were disabled when a CICS system terminated, will still be enabled when the system is restored using a warm start or emergency restart.

Replacing existing transient data queue definitions

You can replace an existing transient data queue if certain conditions are satisfied.

About this task

All the following conditions must be satisfied before you can replace an existing transient data queue:

- CICS initialization is complete
- The queue TYPE is the same as the existing definition
- The intrapartition queue is disabled
- The extrapartition queue is disabled and closed

Existing definitions **cannot** be replaced during a cold start of CICS. If you are using multiple groups, take great care to ensure that duplicate names do not exist. Duplicate names cause error messages to be issued. The duplicate definition is not used to replace the existing one.

You can use the following transactions and commands to inquire about, set, and discard transient data definitions after they have been installed:

- CEMT INQUIRE TDQUEUE (or EXEC CICS INQUIRE TDQUEUE)
- CEMT SET TDQUEUE (or EXEC CICS SET TDQUEUE)
- CEMT DISCARD TDQUEUE (or EXEC CICS DISCARD TDQUEUE)
- The CECI transaction

Disabling transient data queues

You cannot disable a transient data queue when it is in use or when other tasks are waiting to use it.

About this task

For intrapartition and extrapartition queues:

- If tasks are waiting to use an extrapartition queue, a physically recoverable queue, or a nonrecoverable intrapartition queue, the queue enters a “disable pending” state. The last task to use the queue fully disables it.
- If you try to disable a logically recoverable intrapartition transient data queue when units of work are enqueued on it, the queue enters a “disable pending” state. The last unit of work to obtain the enqueue fully disables the intrapartition queue.
- If a unit of work owns an enqueue on a queue that is in a “disable pending” state, it is allowed to continue making updates.
- When a queue is in a “disable pending” state, no new tasks can alter the queue's state, or its contents. CICS returns a disabled response when you issue a READQ TD, WRITEQ TD, or DELETEQ TD request against a queue that is in a “disable pending” state.

Using dummy transient data queues

You can direct output to a dummy transient data queue when the data does not need to be saved.

About this task

An indirect transient data queue (TDQ) cannot be enabled or disabled at run time. If the data is not required, it can be redirected to a dummy TDQ. You create a dummy TDQ by specifying **DSNAME(DUMMY)** when defining the TDQ. See “TDQUEUE attributes” on page 155. You can redirect the queue permanently or temporarily.

For example, you do not want the audit messages which are directed to the CADS queue.

To redirect the queue permanently, change the definition of the CADS queue in the DFHDCTG group to redirect it to the dummy queue. The change takes effect when CICS is restarted.

To control redirection without restarting:

1. Move the definition of the CADS queue from the DFHDCTG group to a user-defined group.
2. Change the definition of the CADS queue to redirect it to the dummy queue
3. Install the customized queues
4. When the queue is needed, use CEDA to install CADS in group DFHDCTG

Transient data queue resource definitions

Transient data queue definitions describe intrapartition, extrapartition, indirect, and remote transient data destinations.

TERMINAL resources

A TERMINAL resource defines the characteristics of a terminal device which communicates with CICS. Terminal devices include visual display units, printers, operating system consoles, and more specialized devices such as facsimile (FAX) machines.

The unique and possibly dynamic properties of terminals are defined in the TERMINAL definition in the CSD file.

However, many of your terminals have identical properties, and you do not need to define each of them separately and fully to CICS. There are two ways you can reduce the time and effort needed to define each terminal. They are:

1. **TYPETERM definitions**, with or without the QUERY function. Each TERMINAL definition must refer to a TYPETERM definition that defines the properties that are common, often more complex, and usually static. Together, information from the TERMINAL and TYPETERM definitions makes up a terminal entry in the TCT (a TCTTE).

   One TYPETERM can represent a lot of the properties of many terminals. Some of these properties can be undefined at the time of creating the TYPETERM definition. These properties can be determined at logon time for each terminal, from the QUERY structured field.

   There are, however, still more properties that many terminals have in common, to the extent that their TERMINAL definitions would all be identical. CICS provides a facility that avoids the need for each terminal to have its own resource definition installed in the TCT the whole time CICS is active.
2. **Autoinstall**, using one TERMINAL definition to represent many terminals. You can let CICS create and install the resource dynamically when the terminal is needed, at logon time. To do this, CICS uses a **model TERMINAL definition** from the CSD file. This process is known as automatic installation, or autoinstall.

   Autoinstall reduces the virtual storage required for the terminal control table (TCT) if some of your terminals are not logged on when CICS is active.

   If you are involved in planning for and managing CICS communications resources such as terminals, read Autoinstalling VTAM terminals for further information.

Terminals for printing

A TERMINAL definition for a display device can name TERMINAL definitions for printers, using the PRINTER and ALTPRINTER attributes. Such a reference is not resolved when the TERMINAL definitions are installed. Instead, the reference is resolved when the printer is needed by the display device.

There are several ways in which printed output can be created and sent to a printer.

- BMS page building
- Screen copying, using one of the following:

A hardware copy key
A local copy key
The ISSUE PRINT command

The TYPETERM and TERMINAL definitions are used for both printers and display devices. A number of attributes apply only to printers, or have special meanings for printers. There are also some attributes that you need to specify for a display device that is to be used for screen-copying.

Printers

Supply a TERMINAL definition for each printer.

Specify NO for AUTINSTMODEL, unless you are using autoinstall for printers. (For more information about this, see Autoinstall and output-only devices.)

ALTPAGE
For BMS, the PAGESIZE attribute determines the default page size, and also the size of the print buffer. You specify the number of lines in the page (the length) and the number of characters in each line (the width).

Another attribute, ALTPAGE, indicates the page size to be used when the alternate screen size (ALTSCREEN) is selected. The width you specify in ALTPAGE must be the same as the width specified in the ALTSCREEN attribute. However, the length of ALTPAGE and ALTSCREEN can be different. This could be useful if you are using the same BMS map to display and to print. For instance, you could make the screen one line longer than the page, to reserve the last line of the screen for error messages.

The ALTPAGE, DEFSCREEN, and ALTSCREEN attributes do not normally apply to printers.

AUTOPAGE
AUTOPAGE must be YES for printers, but you do not need to worry about it, because RDO fills it in for all printer DEVICE types. Autopaging means that BMS multiple page messages are printed continuously, without operator intervention. This is what is normally required for a printer. (Contrast the requirement for multiple page messages, displayed on a 3270-type display, when the operator wants to finish reading a page before requesting the next page.)

Only BMS SEND commands with the PAGING option use autopaging. BMS SEND with TERMINAL or SET does not use autopaging.

You need at least one TYPETERM definition for each type of printer you use. You may need more, if you want to allow printers to be used only for some functions and not for others.

DEVICE
The TERMINAL definition for each printer must refer to a TYPETERM with an appropriate DEVICE type. The DEVICE attribute and, in one case, the SESSIONTYPE attribute, determine whether a TYPETERM defines printers or display devices. The values that you can specify for printers are:

DEVICE	SESSIONTYPE	Printers
3270P	-	All printers that support the 3270 data stream (not SNA-connected).
LUTYPE3	-	All printers that support the 3270 data stream (SNA-connected).
SCSPRINT	-	All printers that support the SNA character set (SNA-connected).
3790	SCSPRINT	IBM 3793 keyboard-printers that support the SNA character set (SNA-connected).

FORMFEED
Define FORMFEED as YES for BMS page building.

PAGESIZE
specifies the default page size for this terminal. The product of lines and columns must not exceed 32767, where lines = the number of lines in the page, and columns = the number of characters in each line.

If PGESIZE is not coded, the following defaults are used:

TW33, TW35	(12,80)
3270 display model 1	(12,40)
3270 display model 2	(24,80)
3270 printer	(12,80)

TERMINAL
The name of the printer is the TERMINAL name on the resource definition for that printer.

Note: The PRINTER attribute is used on a display device definition to refer to a printer device to be used for output from the display. Do not specify PRINTER on the printer definition. See "Associating printers with display devices" on page 174.

Associating printers with display devices

If you want to copy the contents of a screen direct to a printer, associate a specific printer with the display device in the TERMINAL definition. You do this using the PRINTER and ALTPRINTER attributes.

ALTPRINTCOPY
Specify YES for ALTPRINTCOPY if you want to use the hardware COPY feature on the alternative printer.

ALTPRINTER
The alternative printer is used if the primary printer is unavailable. ALTPRINTER and ALTPRINTCOPY can be set dynamically by the autoinstall control program, if the display device definition is autoinstalled.

PRINTER
The primary printer is to be used. PRINTER and PRINTERCOPY can be set dynamically by the autoinstall control program, if the display device definition is autoinstalled.

PRINTERCOPY
Specify YES for PRINTERCOPY if you want to use the hardware COPY feature on the primary printer.

Pipeline terminal definitions

When you define a 3600 pipeline logical unit, you generate a TCTTE that is associated with a pool of TCTTEs.

As messages enter CICS from the 3600 pipeline logical unit, a task is attached to process this message, using as an anchor block one of the TCTTEs from the pool. In this way, consecutive messages sent via the pipeline logical unit can be processed concurrently, with the number of concurrent transactions being limited by the number of TCTTEs in the pool. The number of TCTTEs in the pool should represent the high-water mark of inquiry activity. In this way, the pipeline facility allows fewer TCTTEs to be defined to CICS than the total number of pipeline inquiry terminals.

All the TERMINAL definitions within a named POOL must be in the same group in the CSD file. TASKLIMIT must have been specified on at least one of the definitions in the group. If it is specified on more than one definition, the value used is the maximum value of TASKLIMIT over the definitions in the group.

TERMINAL and TYPETERM definitions are resolved as for ordinary terminals.

If you do not install PIPELINE terminals by GROUP, the results can be unpredictable.

Note: CICS does not support automatic transaction initiation (ATI) on pipeline terminals.

Defining pipeline terminals

When you define a pipeline terminal, you specify particular attributes.

Define pipeline terminals with the following attributes:

NETNAME
The z/OS Communications Server session that is used.

POOL
All the TERMINAL definitions having the same POOL name belong to the same pipeline pool. The presence of a value for the POOL attribute distinguishes these from ordinary TERMINAL definitions.

SESSIONTYPE
Use this attribute on the TYPETERM definition to identify the TYPETERM as representing pipeline terminals. Specify PIPELINE as the value.

TASKLIMIT
Specify the maximum number of concurrent tasks that can be active for the pool of terminals on at least one of the TERMINAL definitions.

One TYPETERM would normally suit all the definitions. The TYPETERM may be in another group.

Pipeline transactions are associated with a PROFILE definition that has the ONEWTE attribute. A program associated with these transactions is permitted only one write or EXEC CICS SEND operation, or else it is terminated with an ATCC abend code. This means that CICS tasks rapidly appear and disappear across the pool of sessions.

There is an example of definitions for a pool of pipeline terminals in the description of the POOL attribute in “TERMINAL attributes” on page 181.

Note:

1. If you install a pipeline terminal naming a pipeline that already exists in CICS, both the old pipeline and all its related terminals are deleted before the new definitions are installed.
2. If you discard the terminal that is defined as owning the existing pipeline, the existing pipeline and all its related terminals are deleted.
3. If you discard a terminal that is not the pipeline owner or change it to a different pipeline, or to a nonpipeline terminal, the rest of the pipeline definition is unchanged. (The owning terminal of a pipeline is the terminal with the first name in alphanumeric sequence that is related to the pipeline in the group from which the pipeline was installed.)
4. You cannot change a terminal in an existing pipeline to a nonpipeline terminal, or change it to a new pipeline, if the old pipeline is also being reinstalled in the same group. To do this, you divide the installation into two stages. If you are installing the group twice, remember to set the relevant terminals out of service in the meantime.

Devices with LDC lists

For 3600, 3770 batch, 3770, and 3790 batch data interchange, and LUTYPE 4 logical units, you can specify the name of an LDC (logical device code) list.

The list specifies which LDCs are valid for this logical unit and, optionally, which device characteristics are valid for each LDC. The first LDC in this list is the default when CICS must choose a default LDC for a logical unit.

All the TERMINAL definitions for devices that reference a particular LDC list must name the same TYPETERM, because the LDCLIST attribute is on the TYPETERM definition.

The LDC list itself must be defined using a DFHTCT TYPE=LDCLIST macro. It may be a local LDC list or an extended local LDC list. There is an example of the coding for each in “TYPETERM attributes” on page 230, and further guidance in “DFHTCT logical device codes: z/OS Communications Server non-3270” on page 398. You use the label coded on the macro instruction to identify the LDC list on the TYPETERM definition, specifying it in the LDCLIST attribute.

APPC (LUTYPE6.2) single session terminal

An APPC (LUTYPE6.2) single session terminal can be defined as a TERMINAL, with a reference to a TYPETERM with DEVICE(APPC). When these definitions are installed, the resources they define are known to CICS as a connection and a modeset, just as they are when defined in RDO as CONNECTION and SESSIONS. The name of the connection is the TERMINAL name and the name of the modeset is the MODENAME on the TERMINAL definition.

An APPC terminal may be a PS/2, an Application System/400 (AS/400), a System/38, or similar.
You can define an APPC terminal either by a TERMINAL-TYPETERM definition or by a CONNECTION-SESSIONS definition. Both kinds of definition can be autoinstalled (see Autoinstalling VTAM terminals and Autoinstalling APPC connections for information about autoinstall). If you decide to use the TERMINAL-TYPETERM method, the following attributes are important:

DEVICE
The TERMINAL definition references a TYPETERM with APPC (advanced program-to-program communications) specified as the DEVICE. One such TYPETERM definition suffices for many terminals.

MODENAME
This is the name that CICS uses to identify the session when the definition is installed in the active system.

Terminals for transaction routing

Transaction routing enables terminals in one CICS system to start transactions in another CICS system.

In these topics, the word “system” is used to indicate a CICS system communicating with other systems using IPIC, MRO, or ISC. You can use transaction routing between systems connected by IPIC, MRO, or by an APPC link. See CICS transaction routing for more information.

You can define a terminal for use in transaction routing as one of the following types:

- A local terminal
- A remote terminal
- A dual-purpose terminal

The type of terminal that you define depends on how the definition is made available to the application-owning CICS system (AOR). You can make the definition available using one of the following methods:

- Duplicating terminal definitions
- Sharing terminal definitions
- Shipping terminal definitions

Terminal definitions used in transaction routing

You can define your terminals for use in transaction routing in two different ways. You can install a *local terminal definition* in the terminal-owning region and a *remote terminal definition* in the application-owning region. Alternatively, you can install a *dual-purpose terminal definition* in both regions.

Local terminal definitions

A *local terminal definition* is a full definition of the terminal, installed in the terminal-owning system.

A local terminal definition is used in the **duplicating** and in the **shipping** methods.

- No REMOTESYSTEM, REMOTESYSNET, or REMOTENAME is needed on the TERMINAL or CONNECTION definition.

  If the REMOTESYSTEM named on the TERMINAL definition is the name of the local system (as specified by the SYSIDNT system initialization operand), the definition is treated as a local terminal when it is installed, rather than a remote one.
- SHIPPABLE on the TYPETERM is NO if duplicating definitions.

- SHIPPABLE on the TYPETERM is YES if shipping definitions.
- SHIPPABLE on the TYPETERM is obligatory for APPC devices.
- All other necessary attributes on the TERMINAL and TYPETERM definitions must be specified.

Remote terminal definitions

When a terminal belonging to (local to and fully defined in) one system invokes a transaction belonging to another system, it is known to the application-owning region as a *remote terminal*.

The application-owning system needs to have access to at least a partial definition of the remote terminal. This partial definition is often known as a *remote definition*. This is a partial definition of the terminal, installed in the application-owning region and intermediate regions. It contains the minimum information necessary for the terminal to access a transaction in that system. You create remote definitions only if you are using the **duplicating** method.

- The REMOTESYSTEM name must be the name of the CONNECTION definition for the next region in the transaction routing path to the terminal-owning region.
- REMOTESYSNET must be the netname (generic applid) of the terminal-owning region. If REMOTESYSTEM names a direct CONNECTION to the terminal-owning region, REMOTESYSNET is not required unless the terminal-owning region is a member of a z/OS Communications Server generic resource group, and the direct connection is an APPC link.
- The REMOTENAME is the name by which the terminal or APPC device is known by in the terminal-owning region. It may be the same as or different from the TERMINAL or CONNECTION name, which must be the name the terminal is known by in the application-owning system. REMOTENAME defaults to the TERMINAL name if not specified.
- SHIPPABLE on the TYPETERM is NO for non-APPC devices.
- SHIPPABLE on the TYPETERM is obligatory for APPC devices.
- Some of the attributes on the TERMINAL and TYPETERM definitions may be omitted. For further guidance about these definitions, see Defining intercommunication resources in Configuring.

The following terminals and logical units cannot use transaction routing and therefore cannot be defined as remote:

- Pooled 3600 or 3650 pipeline logical units
- IBM 2260 terminals
- The MVS console

Dual-purpose terminal definitions

A *dual-purpose terminal definition* is a full definition of the terminal, installed in the terminal-owning system and in the application-owning and intermediate regions.

It is used in the **sharing** method only.

- The REMOTESYSTEM name must be the SYSIDNT of the terminal-owning region. This must also be the name of the CONNECTION that you define from the application-owning region to the intermediate region (if any), and from the intermediate region to the terminal-owning region. All the CONNECTION definitions on the path from the application-owning region to the terminal-owning region must be given this same name.
- REMOTESYSNET must be the netname (generic applid) of the terminal-owning region.
- The REMOTENAME is the same as the TERMINAL or CONNECTION name.
- SHIPPABLE on the TYPETERM is NO for non-APPC devices.
- SHIPPABLE on the TYPETERM is obligatory for APPC devices.
- All other necessary attributes on the TERMINAL and TYPETERM definitions must be specified.

Making partial definitions available for transaction routing

You can make a partial definition of a terminal available to the application-owning CICS region in several ways.

Shipping terminal definitions

You can define in the CSD file available to the terminal-owning region, a local terminal definition that can be shipped to other regions when required.

1. You create a local definition for the terminal, in the CSD file of the terminal-owning region. (This can be a shared CSD file.)
2. You install the local definition in the terminal-owning region. (This definition can be installed at system initialization, or by using CEDA INSTALL, or by autoinstall.)
3. When the terminal invokes a transaction belonging to another region, the information necessary to create a remote definition is shipped to that region, and a temporary definition is installed there automatically.

If the local definition was autoinstalled, the shipped definition lasts until the terminal is logged off. Otherwise, the shipped definition lasts until the local definition is installed again, or until the link between the regions is broken.

Duplicating terminal definitions

You can create a separate terminal definition for each region involved, each on a CSD file that is accessible to that region.

One is created as a local definition and one or more are created as remote definitions. This is referred to as **duplicating terminal definitions**, because there is more than one resource definition for the same terminal (the definitions are not necessarily exact duplicates of each other).

To duplicate terminal definitions:

1. Create a local definition for the terminal, in the CSD file of the terminal-owning region, or on a shared CSD file.
2. Create a remote definition for the terminal, in the CSD file of the application-owning region, or in a shared CSD file. If you have more than one application-owning region, you may need more than one remote definition, but if the regions are sharing a CSD file, you may be able to use the same remote definition for them all.
3. Install the local definition in the terminal-owning region. This definition can be autoinstalled.
4. Install the remote definition in the application-owning region(s).

Note: If your regions share a CSD file, make sure the definitions are in different groups, because:

- You want to install them in different regions
- The TERMINAL names are probably the same, so the definitions cannot be in the same group.

Advantages

There are some advantages to duplicating terminal definitions.

- You can use this method whether or not your systems share a CSD file, so you can use it for transaction routing between different MVS images.
- This is the **only** method you can use if your terminals are known by different names in different systems.
- You can use automatic transaction initiation (ATI) in the application-owning system for a remote terminal without having to set up XALTENF and XICTENF exits. This is especially useful for printers, because they must be acquired **before** any ATI can be successful.

Disadvantages

There are some disadvantages to duplicating your terminal definitions.

- The CSD file uses at least twice the necessary amount of disk storage for your definitions.
- The TCT uses more than the necessary amount of virtual storage for your definitions.

- Unnecessary work is involved in maintaining the definitions.
- You can autoinstall the terminal in the terminal-owning system, but you cannot autoinstall it in the application-owning systems.

Example
In this example, an APPC device (netname APPC1) is connected to a terminal-owning region (applid CICA), which is connected in turn to an application-owning region (applid CICB).

Resources which are installed in the terminal-owning region are maintained in CSD1; those installed in the application-owning region are maintained in CDS2. The configuration is illustrated in Figure 4 on page 180.

This table shows the resource definitions required in each region:

Purpose of definitions	**Definitions in the terminal-owning region**	**Definitions in the application-owning region**
Defines the connection between the systems	`CONNECTION(AOR)` `NETNAME(CICB)`	`CONNECTION(TOR)` `NETNAME(CICA)`

Defines the connection to the APPC device, using a TERMINAL and TYPETERM definition	`TERMINAL(APC1)`
	`TYPETERM(APPC0001)`
	`NETNAME(APPC1)`
	
	`TYPETERM(APPC0001)`
	`DEVICE(APPC)`
		`TERMINAL(BPC1)`
		`TYPETERM(APPC0001)`
		`NETNAME(APPC1)`
		`REMOTESYSTEM(TOR)`
		`REMOTENAME(APC1)`
		
		`TYPETERM(APPC0001)`
		`DEVICE(APPC)`

Defines the connection to the APPC device, using a CONNECTION and SESSIONS definition	`CONNECTION(APC1)`
	`NETNAME(APPC1)`
	
	`SESSIONS(APPC0001)`
	`CONNECTION(APC1)`
		`CONNECTION(BPC1)`
		`NETNAME(APPC1)`
		`REMOTESYSTEM(TOR)`
		`REMOTENAME(APC1)`

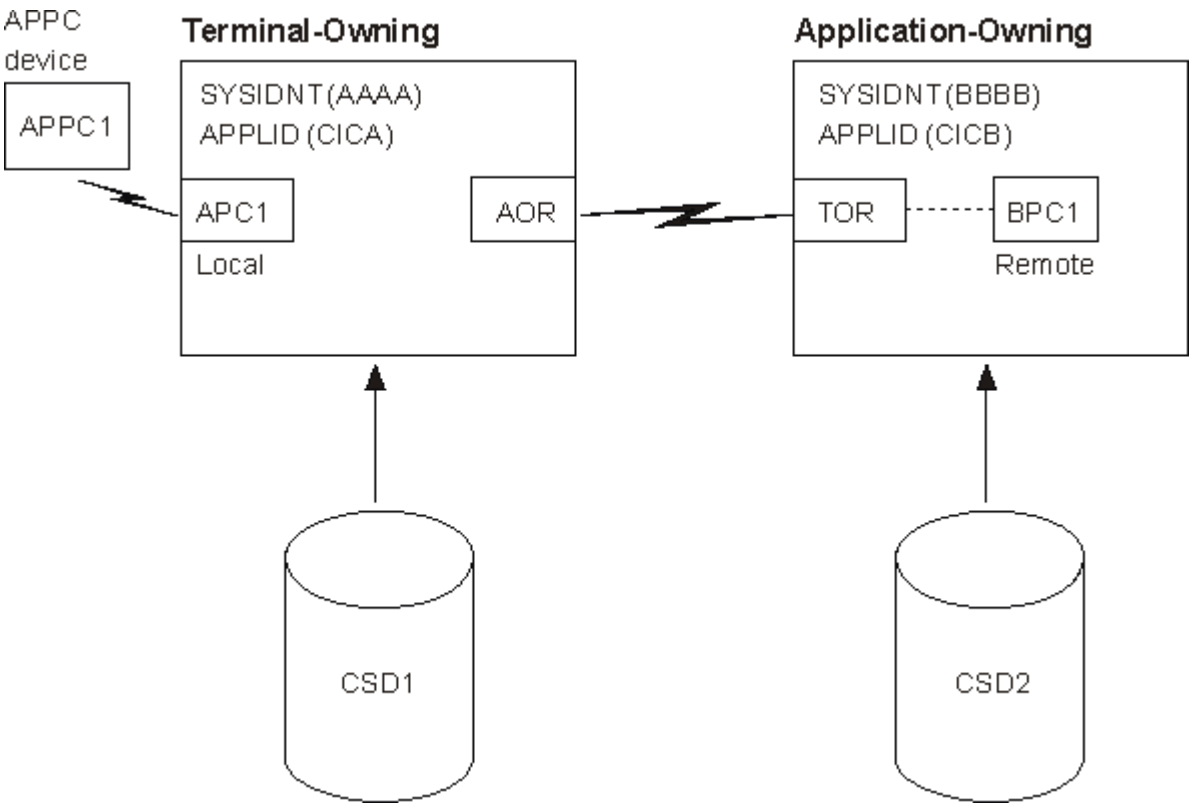


Figure 4. Maintaining local and remote definitions separately

APPC devices for transaction routing

APPC transaction routing allows an APPC device belonging to one CICS system to invoke transactions in another CICS system.

The three methods of defining a terminal for transaction routing, described in "Terminals for transaction routing" on page 176, are also applicable to APPC devices. These methods can involve the use of either TERMINAL-TYPETERM or CONNECTION-SESSION definitions. For APPC single session terminals, the TERMINAL-TYPETERM definition method is recommended (see "APPC (LUTYPE6.2) single session terminal " on page 176.)

Transaction routing summary

Before you define terminals for transaction routing, you must consider which of the various methods of defining terminals is most suitable for your CICS configuration.

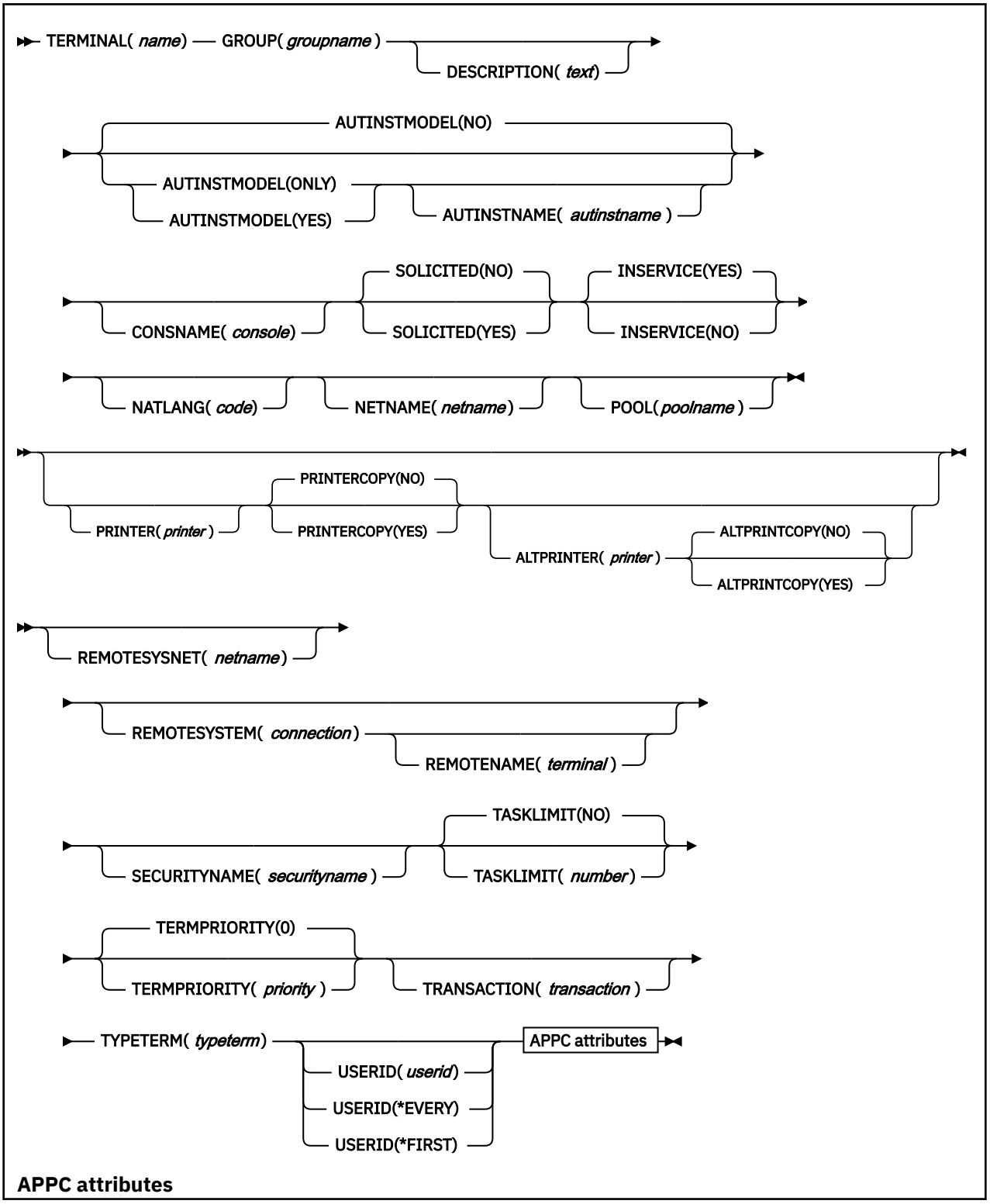
- Use the **shipping method**, unless you use terminals that are known by different names in different systems. For ATI to work with the shipping method in a transaction owning system, you might need to use the XALTENF and XICTENF global exits. See Terminal not known condition exits XALTENF and XICTENF for programming information on these exits.
- Use the **sharing method** for systems with a shared CSD file, if you use the same names in different systems and you do not want to use global exits to ensure that ATI works.
- Use the **duplicating method** if you use terminals that are known by different names in different systems, or if you use ATI to acquire terminals but do not have a shared CSD file, and you do not want to use the XALTENF and XICTENF global user exits.

You could use a mixture of methods: perhaps shipping for display terminals, and duplicating for printers that need ATI to acquire them but without the use of the XALTENF and XICTENF global user exits.

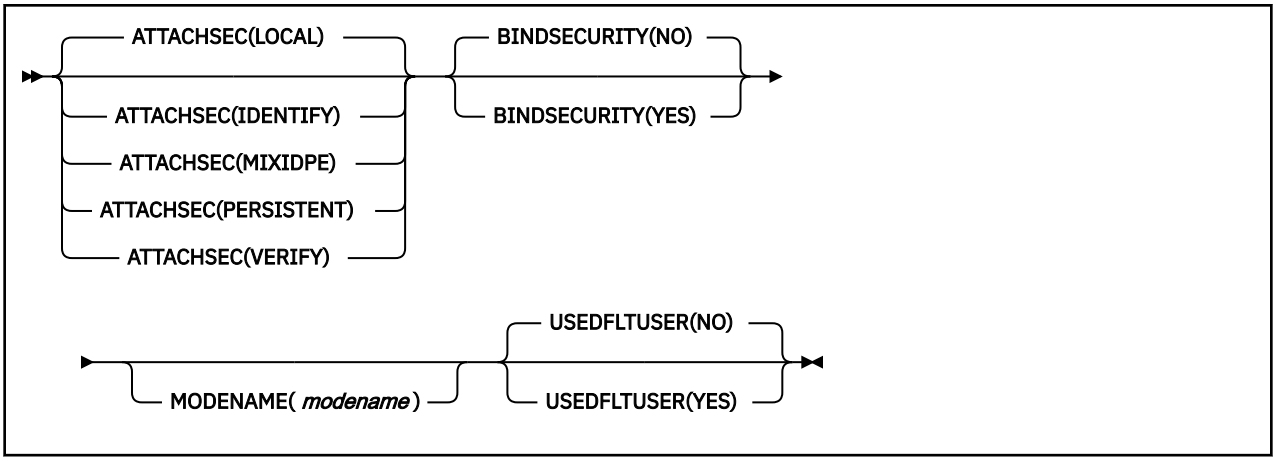
Before you start creating definitions for intercommunication resources, see Defining connections to remote systems for further information. There, you can find useful examples of the attributes you must specify for different types of links and sessions.

TERMINAL attributes

Describes the syntax and attributes of the TERMINAL resource.



APPC attributes



ALTPRINTCOPY({NO|YES})
Specifies whether the hardware COPY feature is to be used to satisfy a print request on the printer named in the ALTPRINTER attribute. For more details, see the PRINTERCOPY attribute.

NO
CICS will use the hardware COPY feature.

YES
CICS will not use the hardware COPY feature.

ALTPRINTER(*printer*)
Specifies the name of a 3270 printer to be used, if the printer named in the PRINTER attribute of this terminal definition is not available. The name can be up to 4 characters in length. For more details, see the PRINTER attribute. If you specify an ALTPRINTER without specifying a PRINTER, ALTPRINTER is ignored.

The printer that you name must be owned by the CICS system that owns this terminal definition.

To specify the hardware COPY feature for the alternative printer, specify YES for ALTPRINTCOPY on this terminal definition.

ATTACHSEC({LOCAL|IDENTIFY|VERIFY| PERSISTENT|MIXIDPE}) (APPC only)
Specifies the level of attach time user security required for the connection. PERSISTENT and MIXIDPE are valid only with z/OS Communications Server as the access method and when APPC is used.

LOCAL
The authority of the user is taken to be the authority of the link itself, and link security protects only the resource.

IDENTIFY
Incoming attach requests must specify a user identifier. Specify IDENTIFY when the connecting terminal has a security manager.

MIXIDPE
A connection can support attaches using either or both of the IDENTIFY and PERSISTENT security types. The security type used depends on the incoming attach.

As in previous releases, IDENTIFY implies a degree of trust between the two systems that allows this system to accept the sign-on logic of the other system. In effect, you have a distributed security manager, with one system performing the sign-on function and the other system performing the security check.

PERSISTENT
This option involves a user sign-on to a remote system that persists over multiple conversations until the user signs off from the remote system. In this way, the user ID and password are passed only on the first (sign-on) attach. Subsequent attach requests require only the user ID.

VERIFY
Incoming attach requests must specify a user identifier and a user password. Specify VERIFY when the connecting terminal has no security manager and therefore requires verification.

AUTINSTMODEL({NO|YES|ONLY})
Specifies whether this terminal definition can be used as a model terminal definition for autoinstall. For more information on autoinstall and model terminal definitions, see Autoinstalling VTAM terminals.

NO
This definition is not used as a model for autoinstall. It is used only as a definition for a specific device that is not autoinstalled.

ONLY
This definition is used only as a model for autoinstall. It is not used as a definition for a specific device.

YES
This definition is used for a specific device that is not autoinstalled. The definition is also used as a model for automatic installation.

AUTINSTNAME(*autinstname*)
Specifies the name by which this model definition is known in the autoinstall control program. The name can be up to 8 characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

You specify this name only if AUTINSTMODEL is YES or ONLY. You can default to the terminal name followed by four blanks, if this name is acceptable to the autoinstall control program. For more information about autoinstall models, autoinstall names, and the autoinstall control program, see Autoinstall control program.

BINDPASSWORD
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS.

BINDSECURITY({NO|YES}) (APPC only)
Specifies whether an external security manager (ESM) is being used for bind-time security.

NO
No external bind-time security is required.

YES
If security is active and the **XAPPC** system initialization parameter is set to YES, CICS attempts to extract the session key from RACF to carry out bind-time security. If no RACF profile is available, the bind fails.

CONSOLE
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS.

CONSNAME(*console*)
Using CONSNAME, you can install a CICS console definition without having an existing console, and without the console being previously defined in the CONSOL*nn* member of the MVS SYS1.PARMLIB. However, before you can use the console, you must define the name to MVS, either in the CONSOL*nn* member of SYS1.PARMLIB or by dynamic allocation. The length of CONSNAME must be 2 - 8 characters and must begin with an alphabetic character or one of #, @, or $. It uniquely identifies the console device in a CICS region, regardless of the MVS image to which it is connected; that is, you cannot install two console definitions with the same CONSNAME. The CONSNAME corresponds to the name defined for the console in the MVS SYS1.PARMLIB member, CONSOL*nn*.

To define a TSO user as a console device, specify CONSNAME(*name*), where *name* is the TSO user ID. This definition enables a TSO user authorized to use the TSO CONSOLE command to initiate CICS transactions. The TSO user ID does not have to be defined in the CONSOL*nn* member of SYS1.PARMLIB member.

The equivalent of CONSOLE(00) is CONSNAME(INTERNAL) or CONSNAME(INSTREAM), depending on the service level of CICS and the release of MVS being used; specify this option if you want to initiate a CICS transaction and issue a command to it in a JCL statement. For guidance about using JCL to issue CICS commands, see Using JCL to initiate CICS commands.

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

INSERVICE({YES|NO})
Specifies the status of the terminal that is being defined.

YES
Transactions can be initiated and messages are automatically sent to the terminal.

NO
The terminal cannot receive messages or transmit input.

MODENAME(*modename*) (APPC single session terminals only)
Specifies the name that is passed to z/OS Communications Server as the LOGMODE name. The name can be up to 8 characters in length, but cannot have the reserved name SNASVCMG. The name follows assembler language rules and must start with an alphabetic character.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

For further guidance on the LOGMODE name, see Coding entries in the z/OS Communications Server LOGON mode table.

NATLANG(*code*)
Specifies the language in which all globalization-enabled messages are displayed for this terminal. Specify one of the following NATLANG codes. The default is the value specified in NATLANG system initialization parameter.

E
English, which is the *system* default (that is, is provided even if you do not specifically code E).

C
Simplified Chinese, a Double-Byte Character Set language. Translation is performed by IBM.

K
Japanese, a Double-Byte Character Set language. Translation is performed by IBM.

NETNAME(*netname*)
Specifies the network name that identifies the terminal to ACF/Communications Server. The name can be up to 8 characters in length. The name follows assembler language rules and must start with an alphabetic character.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

If you do not specify a name, the NETNAME defaults to the TERMINAL name.

The NETNAME must be unique except in the case of a remote terminal. That is, you cannot install two local terminals with the same NETNAME, or a local terminal and any connection with the same NETNAME. However, the NETNAME for a remote terminal can be the same as the NETNAME for any other terminal or the NETNAME for any connection.

If the CICS region supports z/OS Communications Server dynamic LU alias; that is, LUAPFX=*xx* is specified on the CICS region APPL statement, the terminal with this NETNAME is assumed to be in the same network as the CICS region. If the terminal is in another network, it must be defined to z/OS Communications Server on a CDRSC definition with a predefined LUALIAS (LUALIAS=*netname*) to override z/OS Communications Server dynamic allocation. In this case, *netname* on the LUALIAS parameter must match the NETNAME defined on this terminal resource definition.

OPERID
OPERPRIORITY
OPERRSL
OPERSECURITY
These attributes are obsolete, but are supported to provide compatibility with earlier releases of CICS.

POOL(*poolname*)
Specifies the pool name for a 3600 or 3650 pipeline terminal pooled with other pipeline terminals.

When you define a 3600 pipeline logical unit, you generate a TCTTE that is associated with a pool of TCTTEs. A pool of pipeline TCTTEs can be used by one pipeline logical unit, or it can be shared by a number of pipeline logical units.

The pool name is used only as a method of identifying the related TERMINAL definitions on the CSD file. It is not used in the active CICS system. The name can be up to 8 characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

For a pipeline terminal, you must specify a TYPETERM with SESSIONTYPE(PIPELINE) specified. You must specify a TASKLIMIT on at least one of the pool of pipeline terminals. You must name the same group for each of the pipeline terminals in a pool.

Here is an example of the definitions for a pool of pipeline terminals:

```
DEFINE TERMINAL(ttt1) GROUP(g) POOL(poolid)
       TYPETERM(xxxxxxxx) NETNAME(nnnnnnn1)...
DEFINE TERMINAL(ttt2) GROUP(g) POOL(poolid)
       TYPETERM(xxxxxxxx) NETNAME(nnnnnnn2)...
DEFINE TERMINAL(ttt3) GROUP(g) POOL(poolid)
       TYPETERM(xxxxxxxx) NETNAME(nnnnnnn3)...
DEFINE TERMINAL(ttt4) GROUP(g) POOL(poolid)
       TASKLIMIT(nn) TYPETERM(xxxxxxxx)
       NETNAME(nnnnnnn4)...
```

```
DEFINE TYPETERM(xxxxxxxx) GROUP(g)
       DEVICE(3650) SESSIONTYPE(PIPELINE)
```

PRINTER(*printer*)
Specifies the name of the primary 3270 printer to be used to respond to an ISSUE PRINT command or a PRINT request from an operator pressing a program access (PA) key. The name can be up to 4 characters in length. The name is the TERMINAL name on the definition for the printer. If you name a PRINTER here, the TYPETERM referenced by this TERMINAL definition must have PRINTADAPTER(NO).

The printer that you name must be owned by the same CICS system that owns this TERMINAL definition.

You can name a PRINTER if this TERMINAL definition is for one of the following:

- A 3270 display without the printer-adapter feature
- A 3270 display attached to a 3274 control unit
- A 3276 control unit display station
- A 3790 in 3270 compatibility mode

If you want to specify the hardware COPY feature, specify PRINTERCOPY(YES) on this TERMINAL definition.

Note that SNA character string (SCS) printers accept only 3790 data streams; they do not accept 3270 data streams. They therefore cannot be used to print the buffer contents of a display unit.

If you use a program attention key, for example, PA1, to print the contents of the screen on an associated z/OS Communications Server printer, the screen size of the printer is chosen according to the SCRNSIZE operand as defined in the CICS-supplied default profile DFHCICST. This profile is defined with SCRNSIZE(DEFAULT) and, if you want to use the alternate screen size of the printer, you have to change the profile entry for transaction CSPP.

PRINTERCOPY({NO|YES})
Specifies whether the hardware COPY feature is to be used to satisfy a print request on the printer named in the PRINTER attribute of this terminal definition.

CICS uses the hardware COPY feature of the 3270 to perform the print, unless a task is currently attached to the display.

You do not need to specify COPY(YES) on the TYPETERM definition, because it is implied by PRINTERCOPY(YES) on the terminal definition.

If you have named an ALTPRINTER as well as a PRINTER, you can specify ALTPRINTCOPY(YES).

To use the COPY feature, both the printer and the display terminal must be on the same 3270 control unit. Otherwise, either the COPY might fail, raising an error condition, or, if the display device address is valid for the printer control unit, copying might be performed from a different display.

Do not specify PRINTERCOPY(YES) if, in a networking environment, the 3270 control unit is connected to a TCAM system in one domain, and a CICS system in another domain has access to the control unit through z/OS Communications Server. The hardware COPY address is not available to CICS and cannot therefore be used by terminals attached to such a control unit.

The COPY command is invalid for a 3270 compatibility mode display.

REMOTENAME(*terminal*)
Specifies the name by which the terminal is known in the system or region that owns the terminal. The name can be up to 4 characters in length.

Acceptable characters:

```
A-Z a-z 0-9 $ @ ♯ . / - _ % & ? ! : | " = ¬ , ; < >
```

REMOTESYSNET(*netname*)
Specifies the network name (APPLID) of the region that owns the terminal. The name can be up to 8 characters in length. It follows assembler language rules, and must start with an alphabetic character.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

REMOTESYSNET is used where there is no direct link between the region in which this definition is installed and the terminal-owning region. You do not need to specify REMOTESYSNET if either of the following is true:

- You are defining a local terminal; that is, REMOTESYSTEM is not specified, or specifies the sysid of the local system.
- REMOTESYSTEM names a direct link to the terminal-owning region. However, if the terminal-owning region is a member of a z/OS Communications Server generic resources group and the direct link is an APPC connection, you might need to specify REMOTESYSNET. REMOTESYSNET is needed in this case if the NETNAME specified on the CONNECTION definition for the direct link is the generic resource name of the TOR (not the applid).

REMOTESYSTEM(*connection*)
Specifies the name that identifies the intercommunication link to the system that owns the terminal. The name can be up to 4 characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

REMOTESYSTEM is one of these identifiers:

- For IPIC connections, the first 4 characters of the IPCONN name on the IPCONN definition, which is in service and acquired.
- For MRO and APPC connections, the CONNECTION name on the CONNECTION definition. If the CONNECTION name is not specified, or if the name is specified as the sysid of the local system, the terminal is local to this system. If the name is the name of another system, the terminal is remote. You can therefore use the same definition for the terminal in both the local system and a remote system.

If you have intermediate systems between this CICS and the terminal-owning region, REMOTESYSTEM specifies the first link in the path to the TOR. If more than one path is possible, REMOTESYSTEM specifies the first link in the preferred path.

REMOTESYSTEM is ignored if you specify AUTINSTMODEL(YES) or AUTINSTMODEL(ONLY).

SECURITYNAME(*securityname*)
Specifies the security name of the remote system.

In a CICS system with security initialized (SEC=YES), the security name is used to establish the authority of the remote system.

If USERID is specified in the session definition (DEFINE SESSIONS command) associated with the connection definition, the user ID overrides the user ID specified in the SECURITYNAME attribute, and is used for link security.

The security name, or USERID on the sessions definition, must be a valid RACF user ID on your system. Access to protected resources on your system is based on the RACF user profile and its group membership.

For more information on defining security for MRO, LUTYPE6.1, and APPC connections, see Intercommunication security.

SOLICITED(NO|YES)
Specifies whether CICS messages issued to a console will be treated by NetView® as solicited or unsolicited.

NO
CICS messages will be treated as unsolicited

YES
CICS messages will be treated as solicited. When SOLICITED(YES) is specified for a console, CICS adds the console name or the console identification number and a command and response token to each console message.

The SOLICITED attribute applies only to consoles; it is ignored for other TERMINAL definitions.

TASKLIMIT({NO|*number*})
Specifies the number of concurrent tasks allowed to run in a pipeline session or in a pool of pipeline sessions.

NO
No concurrent tasks are allowed.

number
The number of concurrent tasks allowed to run, in the range 1 - 32767.

When you define a 3600 pipeline logical unit, you generate a TCTTE that is associated with a pool of TCTTEs. When messages enter CICS from the 3600 pipeline logical unit, a task is attached to process this message, using as an anchor block one of the TCTTEs from the pool. In this way, consecutive messages sent using the pipeline logical unit can be processed concurrently, the number of concurrent transactions being limited by the number of TCTTEs in the pool. The number of TCTTEs in the pool should represents the high-water mark of inquiry activity. In this way, the pipeline facility allows fewer TCTTEs to be defined to CICS than the total number of pipeline inquiry terminals.

TERMINAL(*name*)
Specifies the terminal identifier. For a terminal, the name can be up to 4 characters in length.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

For an APPC LU6.2 single session terminal, the valid characters are as follows:

Acceptable characters:

```
A-Z a-z 0-9 $ @ #
```

If the name supplied is fewer than 4 characters, it is left-justified and padded with blanks up to 4 characters. You should not have a TERMINAL definition and a CONNECTION definition with the same name.

Note: If you use a comma (,) in a name, you cannot use commands such as:

```
CEMT INQUIRE TERMINAL(value1,value2)
CEMT SET     TERMINAL(value1,value2)
```

where the comma serves as a list delimiter. See List of resource identifiers for information about using lists of resource identifiers.

Also, if you are allowing CICS to automatically generate session names or terminal IDs for consoles, avoid using any of the following symbols as the first or fourth character in the terminal name:

-
The hyphen is used by CICS for automatically generated terminal names for APPC sessions

> *and* <
The > (greater than) and < (less than) symbols are used by CICS for automatically generated terminal names for IRC sessions

¬
The ¬ (logical not) symbol is used by CICS for automatically generated terminal names for MVS consoles.

The name specified becomes the name of the TCT entry, when this TERMINAL definition is installed. For this reason, the TERMINAL name must be unique. Note that the value CERR is reserved for the identification generated for the error console. If you specify AUTINSTMODEL(ONLY), you do not need a unique TERMINAL name, because it is not used as the name of a TCT entry. If you specify AUTINSTMODEL(YES), the TERMINAL name is used as the name of the TCT entry that is installed in the TCT when the TERMINAL definition is installed; the names of the TCT entries for the autoinstalled terminals are determined by the autoinstall user program.

If the terminal is to be associated with a transient data destination, the terminal name and the transient data queue name in the TDQUEUE resource definition must be the same.

TERMPRIORITY({0|*priority*})
Specifies the terminal priority. This decimal value (0 - 255) is used in establishing the overall transaction processing priority. Transaction processing priority is equal to the sum of the terminal priority, transaction priority, and operator priority, not exceeding 255.

TRANSACTION(*transaction*)
Specifies a 1- to 4-character name of the transaction that is to be initiated each time input is received from the terminal when there is no active task.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

For z/OS Communications Server non-3270 devices, a TRANSACTION name of fewer than 4 characters requires a delimiter.

For information on what happens when a transaction is initiated, see Terminal control.

If you specify this operand for a 3270 display, the only CICS functions the operator can start, other than this transaction, are paging commands and print requests.

You are unlikely to specify the TRANSACTION attribute for a 3270 display or SCS printer. It is optional for 3601 logical units, but is mandatory for 3614 logical units.

If this operand is specified for a 3790 Communication System, and multiple sessions are used to connect the same 3791, specify the same TRANSACTION name for all sessions.

TYPETERM(*typeterm*)
Specifies the name of the TYPETERM definition to be associated with this TERMINAL definition. The name can be up to 8 characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

The TYPETERM definition specifies many attributes for a number of similar terminals. For more information, see "TYPETERM attributes" on page 230 and Ways of defining CICS resources. The TYPETERM attribute is mandatory for all TERMINAL definitions.

The TYPETERM definition must already be installed when you install this TERMINAL definition.

USEDFLTUSER ({NO|YES}) (APPC only)
Specifies the kind of security checking that will take place for each inbound attach FMH.

NO
Indicates that each inbound attach FMH will be checked for the presence of those fields required by the ATTACHSEC option. If the required fields are not present, a protocol violation message is issued and the attach fails. NO is the default.

YES
Indicates that some checks on the validity of the attach Function Management Header (FMH) are bypassed. See Implementing LU6.2 security.

USERID({name|*EVERY|*FIRST})
Specifies a user identifier used for sign-on and referred to in security error messages, security violation messages, and the audit trail. It must be a valid user ID defined to the security manager.

The USERID attribute provides the only way to specify a user identifier for devices such as printers that are unable to sign on using CESN. You can also specify USERID for a display device; if so, the display is permanently signed on. Operators are unable to sign on. All access to protected resources depends on the authorizations permitted by RACF for the specified USERID.

For an APPC single session terminal, USERID overrides any SECURITYNAME that you have specified for the connection.

name
This name can be up to 8 characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

***EVERY**
This special operand is for autoinstalled consoles only. It means that CICS is to use the user ID passed on the MVS MODIFY command every time a MODIFY command is received. The console is signed on using the MVS user ID as the preset user ID for the console being autoinstalled. The console remains signed on with this user ID until the console is deleted or another MODIFY is received with another user ID. If a MODIFY command is received without a user ID, CICS signs on the default CICS user ID until a MODIFY command is received that has a valid user ID. For non-console terminals, or if security is not enabled, this value is ignored.

***FIRST**
This special is operand for autoinstalled consoles only. It means that CICS is to use the user ID passed on the first MVS MODIFY command that requires the console to be autoinstalled. The console is signed on with the MVS user ID as the preset user ID. The console remains signed on with this user ID until the console is deleted. If a MODIFY command is received without a user ID, CICS signs on the default CICS user ID. For non-console terminals, or if security is not enabled, this value is ignored.

If this terminal definition defines a console, the user ID must be authorized to the appropriate profile in the CONSOLE general resource class. See Defining port of entry profiles for information about preset security on consoles and terminals.

Installing terminal definitions

For a new or changed TERMINAL resource definition to become active in a CICS region, you must install it.

About this task

TERMINAL and TYPETERM definitions are resolved during installation to become a TCT entry, sometimes known as a TCT terminal entry or TCTTE. Each definition contains only some of the information necessary to build a TCT entry.

Each TYPETERM definition must be installed before or at the same time as the TERMINAL definitions that reference it. When you use CEDA to install groups, if the TYPETERM definitions are in a separate group from the TERMINAL definitions, you must install the TYPETERM group before the TERMINAL group. You can include TYPETERM definitions and TERMINAL definitions in the same group for testing purposes, although it is not recommended for long-term use.

Because the TERMINAL definition is not necessarily in the same group as its associated TYPETERM definition, the global catalog is used to store the TYPETERM definitions.

Changing a TYPETERM definition and then reinstalling it has no effect on an already installed terminal entry, even though the TERMINAL definition used to create it refers to the TYPETERM. To change the terminal entry, you must reinstall both the TYPETERM and the TERMINAL definitions.

Use the following procedure for groups of TERMINAL definitions.

Procedure

1. Make sure that nobody is using the terminals that you want to install. Remember that the **CEMT INQUIRE TERMINAL** command puts a lock on the TCT entry, and this also prevents installation of a group containing that terminal.
2. Make sure that there are no ATIs outstanding for the terminals.
3. Install the resource definitions. Use the following command:

```
CEDA INSTALL GROUP(groupname)
```

4. Use CEMT to make the terminals available again. Use the following command:

```
CEMT SET TERMINAL(trmid) ACQUIRED INSERVICE ATI
```

Checking terminal definitions

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

About this task

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

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

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

```
          .
          .
TERMINAL(AZnn) GROUP(AZTERMS)

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

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

Terminal resource definitions

Terminal resource definitions describe the unique characteristics of the terminal devices, including visual display units, printers, and operating system consoles, with which CICS communicates.

TRANCLASS resources

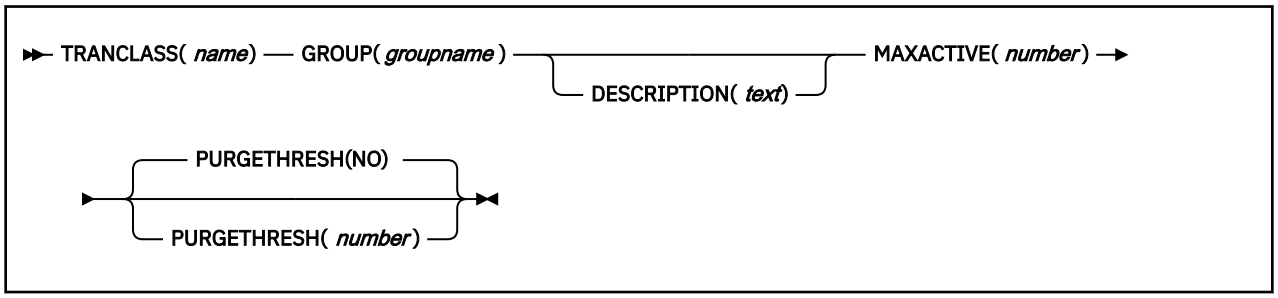
A TRANCLASS resource defines the characteristics of a transaction class.

Transactions that are defined as belonging to a transaction class are subject to scheduling constraints before they are allowed to execute. If transactions belonging to an active transaction class are already running, any new transactions are queued. Use the MAXACTIVE attribute to specify the maximum number of transactions that you want to run. To limit the size of the queue, you can use the PURGETHRESH attribute.

By putting your transactions into transaction classes, you can control how CICS dispatches tasks. For example, you can separate transactions into those that are heavy resource users and those that are of lesser importance, such as the “Good morning” broadcast messages. You can then use the attributes on the TRANCLASS definition to control the number of active tasks allowed from each transaction class.

TRANCLASS attributes

Describes the syntax and attributes of the TRANCLASS resource.



The transaction class definition attribute descriptions are:

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

MAXACTIVE(*number*)
Specifies the maximum number of transactions in this transaction class that are allowed to be active. You must specify a MAXACTIVE value when you define a transaction class, in the range 0 through 999.

New transactions attached when the number of active transactions has reached the MAXACTIVE limit are considered for queueing subject to the PURGETHRESH limit.

Defining a transaction class with a zero MAXACTIVE value signifies that **all** tasks are to be queued.

PURGETHRESH({NO|*number*})
This is an optional purge threshold for the transaction class; it defines a threshold number at which transactions queuing for membership of the transaction class are purged. Specify it if you want to limit the number of transactions queueing in this transaction class. It can have the following values:

NO
The size of the queue is unlimited (other than by the storage available to attach tasks).

number
The purge threshold number in the range 1—1 000 000.

If you specify this as 1, no transactions are allowed to queue. If you specify it as any other number (*n*), the size of the queue is restricted to *number-1*. All new transactions attached after the limit of *n-1* is reached are purged.

Example of PURGETHRESH: In the case of a transaction class where the maximum number of active tasks (MAXACTIVE) is set to 50, and the purge threshold (PURGETHRESH) is set to 10 to limit queuing transactions, CICS begins to abend new transactions for the class when:

- The number of active transactions reaches 50, and
- The number of transactions queuing for membership of the transaction class has reached 9

CICS accepts new transactions for this transaction class queue only when the number queued falls below the maximum size of the queue (9 in our example).

TRANCLASS(*name*)
Specifies the name of the transaction class. Transactions belonging to a transaction class are subject to scheduling constraints before they are allowed to execute. The reserved TRANCLASS name DFHTCL00 is used to indicate that the transaction does not belong to any transaction class.

For compatibility with releases that support a TCLASS attribute, CICS provides the following TRANCLASS equivalents:

TCLASS	TRANCLASS
NO	DFHTCL00
1	DFHTCL01
2	DFHTCL02
3	DFHTCL03
4	DFHTCL04
5	DFHTCL05
6	DFHTCL06
7	DFHTCL07
8	DFHTCL08
9	DFHTCL09
10	DFHTCL10

Sample definitions for these transaction classes are in group DFHTCL, supplied as part of DFHLIST.

Note: If a transaction is run and its associated TRANCLASS definition is not installed, the transaction runs without any of the scheduling constraints specified in the TRANCLASS. Attention message DFHXM0212 is issued.

TRANCLASS can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

Transaction class definitions

Transaction class definitions describe the operational characteristics for transactions belonging to the class.

TRANSACTION resources

A TRANSACTION resource defines transaction attributes that relate to functions provided by CICS.

A CICS application consists of one or more programs written to perform a specific function. A particular invocation of such an application is known as a *transaction*, and the CICS transaction manager identifies it by its transaction identifier (TRANSID). The TRANSACTION resource can also be used as an application entry point. Entry points are described in Application entry points.

You tell CICS how you want your transaction to run, primarily in a TRANSACTION definition, by providing such information as the transaction priority, security key, and the length of the transaction work area (TWA). The name of this definition, the TRANSACTION name, is the same as the TRANSID. You also link the transaction with other resources by coding the names of their definitions in the TRANSACTION definition. These other resources are PROGRAM, PROFILE, PARTITIONSET, REMOTESYSTEM, and TRANCLASS.

PROGRAM
You specify options related to the software implementation of your application in the PROGRAM definition. The program to which control is to be given to process the transaction is then defined. The TRANSACTION definition references the PROGRAM definition.

PROFILE
You do not have to specify, for each transaction, the attributes that control the interaction with a terminal or logical unit. Instead, the TRANSACTION definition references a PROFILE definition, which specifies them for a number of transactions.

REMOTESYSTEM
For transaction routing, instead of specifying a PROGRAM name in the TRANSACTION definition, you specify the name of a REMOTESYSTEM. The value can be the name of another CICS system, which itself is defined to this CICS system in a CONNECTION definition or IPCONN definition of the same name.

If you name a REMOTESYSTEM in a CONNECTION definition, you can also supply a REMOTENAME, which is the name of the transaction to be run in the remote system. The remote system decides which program it gives control to.

If you specify a REMOTESYSTEM name that corresponds to the system in which the definition is installed, CICS installs a local transaction resource. If the REMOTESYSTEM name does not correspond to the system in which the definition is installed, CICS installs a remote transaction resource. If no REMOTESYSTEM name is specified, CICS will install a local transaction resource.

TRANCLASS
This specifies the name of the transaction class to which the transaction belongs. Transactions belonging to a transaction class are subject to scheduling constraints before they are allowed to

execute. The constraints are specified in the associated TRANCLASS definition. The TRANCLASS definition is described in “TRANCLASS resources” on page 192.

TRANSACTION resources in CICS bundles

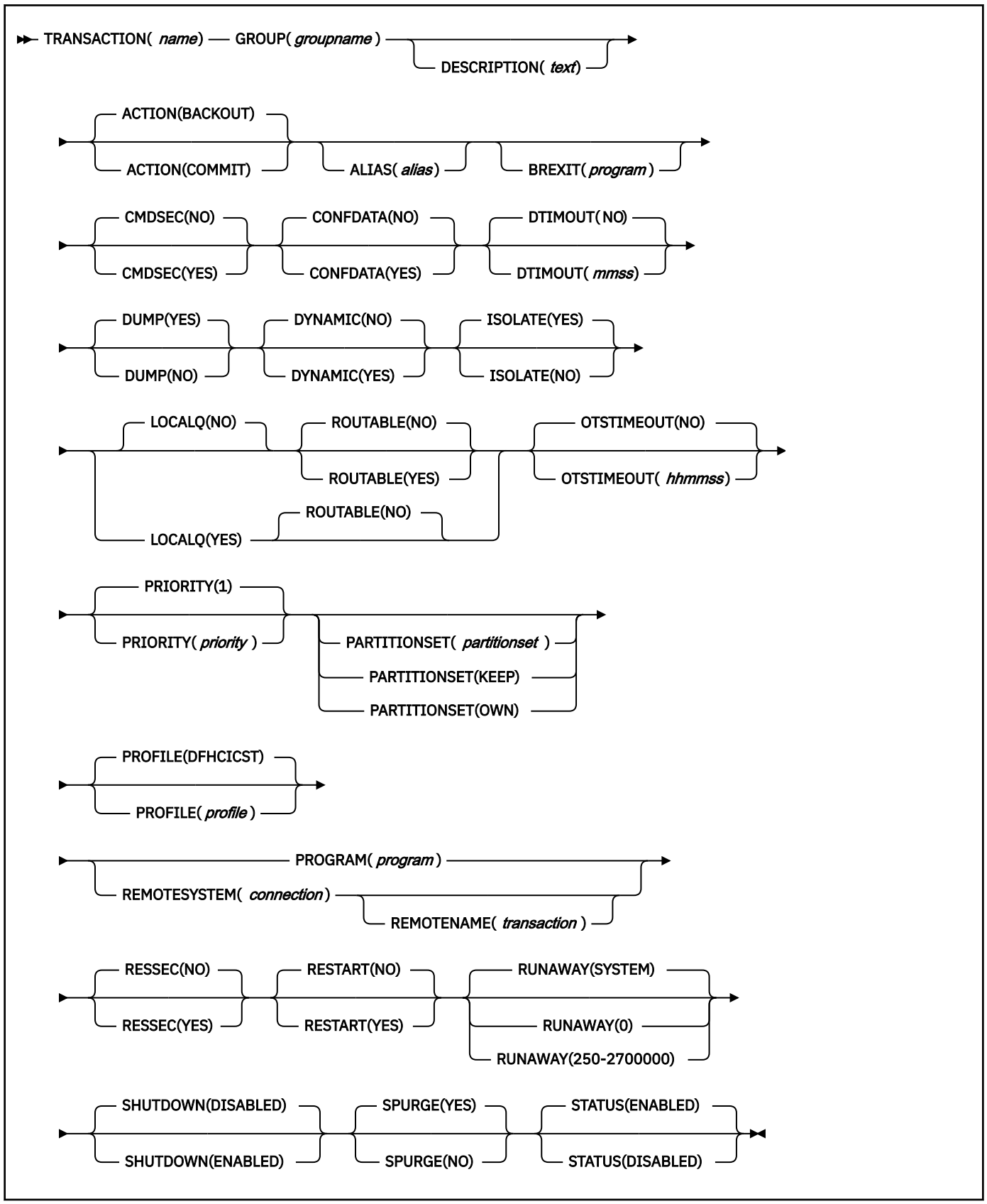
You can use a CICS bundle to create, edit, and install a TRANSACTION resource definition. If you create a TRANSACTION resource in this way, you must use the CICS bundle to manage the lifecycle of that resource, and you cannot manage the resource independently. You can inquire on a TRANSACTION resource that is dynamically generated by a CICS bundle, by using the **EXEC CICS INQUIRE TRANSACTION** or **CEMT INQUIRE TRANSACTION** command. However, you cannot issue SET or DISCARD commands against a TRANSACTION resource that is dynamically generated by a CICS bundle. You must issue the commands against the BUNDLE resource, and CICS applies them to the TRANSACTION resource. You can declare a TRANSACTION resource as an application entry point through a CICS Bundle.

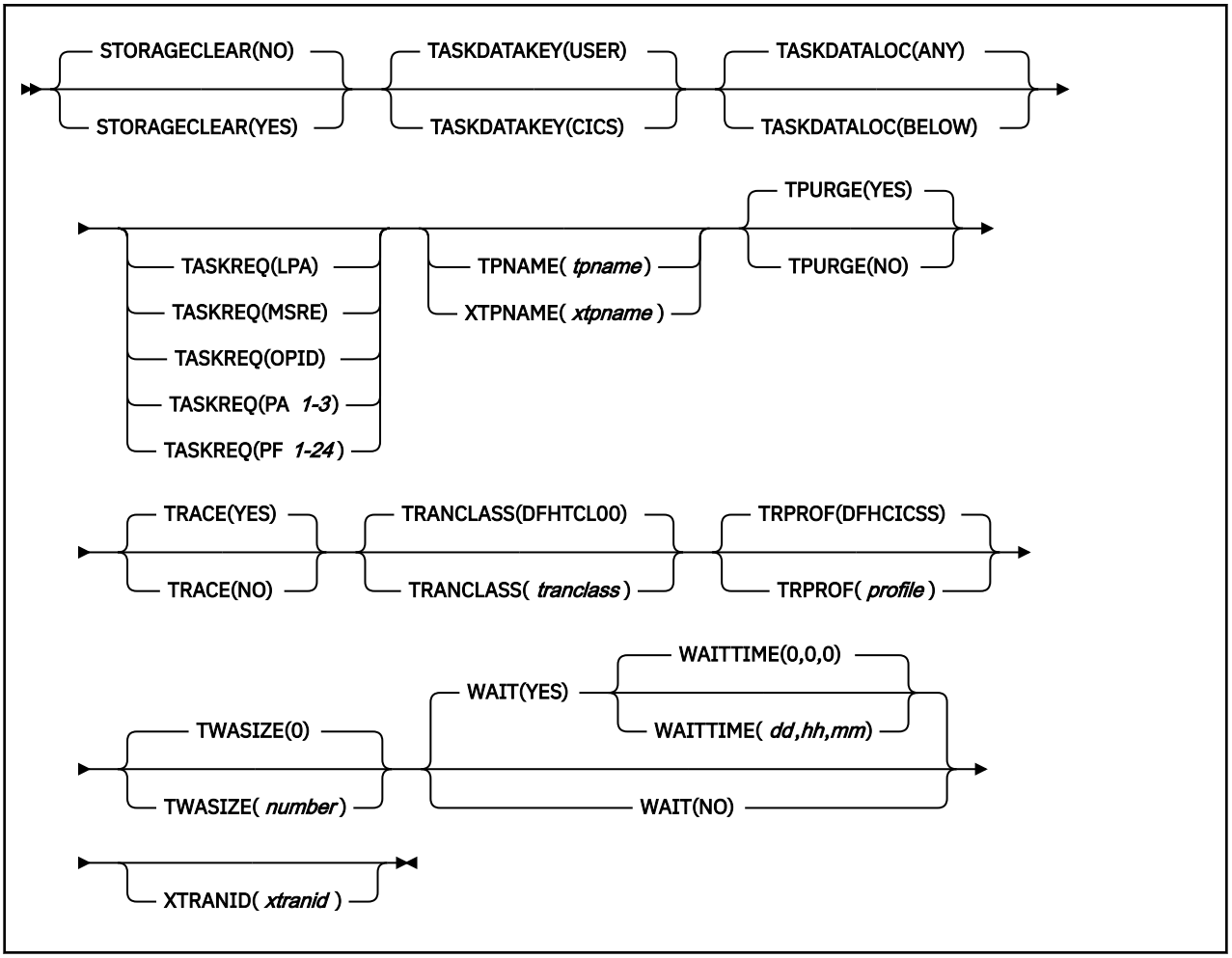
A transaction might be allocated work that is due to be started, by an **EXEC CICS START** command, at a scheduled time. If that work is scheduled to start after the CICS bundle is disabled, the scheduled work is canceled, and no new work is allowed to start for that transaction. If the CICS bundle is then re-enabled, the canceled work is not rescheduled, and remains canceled.

For more information about defining resources in CICS bundles, see Defining CICS bundles.

TRANSACTION attributes

Describes the syntax and attributes of the TRANSACTION resource.





ACTION({BACKOUT|COMMIT})
Specifies the action to be taken when a CICS region fails, or loses connectivity with its coordinator, during 2-phase commit processing after the unit of work has entered the indoubt period. The action depends on the WAIT attribute. If WAIT specifies YES, ACTION has no effect unless the WAITTIME expires before recovery from the failure.

If WAIT specifies NO, the action taken is one of the following:

BACKOUT
All changes made to recoverable resources are backed out, and the resources are returned to the state they were in before the start of the UOW.

COMMIT
All changes made to recoverable resources are committed, and the UOW is marked as completed.

ALIAS(*alias*)
Specifies an alias transaction name for this transaction. The name can be up to 4 characters in length. This option is useful if you want to run on a terminal defined with UCTRAN(NO), or a transaction that allows mixed case input (PROFILE UCTRAN(NO)). For example, you can start the same transaction as ABCD by means of ALIAS(abcd) .

When you install a TRANSACTION definition that contains the ALIAS attribute, the result depends on whether the alias name is already in use in the system:

- If the alias name is in use as a primary transaction ID, the ALIAS attribute is ignored.
- If the alias name is in use as the alias for a different transaction, the original alias is replaced by the new one. In other words, after the TRANSACTION definition has been installed, the alias name refers to the new transaction, and not the original.

BREXIT(*program*)
Defines the name of the default bridge exit to be associated with this transaction, if it is started in the 3270 bridge environment with a **START BREXIT** command that does not specify a name on its BREXIT option. The name can be up to 8 characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

If BREXIT is defined, do not specify REMOTESYSTEM, REMOTENAME, DYNAMIC(YES), or RESTART(YES) because they are ignored.

Note: The Link3270 mechanism is now the recommended way to use the 3270 bridge.

CMDSEC({NO|YES})
Specifies whether security checking is to be applied on system programming commands.

NO
No check is made. The commands are always run.

YES
A call is made to the external security manager (ESM). CICS either authorizes or prevents access. If the ESM cannot identify the resource or resource type, access is prevented.

CONFDATA({NO|YES})
Specifies whether CICS is to suppress user data from CICS trace entries when the CONFDATA system initialization parameter specifies HIDETC. If the system initialization parameter specifies CONFDATA=SHOW, CONFDATA on the transaction definition is ignored.

If the system initialization parameter specifies CONFDATA=HIDETC, the following options are effective:

NO
CICS does not suppress any user data. z/OS Communications Server and MRO initial user data is traced in trace point AP FC9B. FEPI user data is traced in the normal CICS FEPI trace points. IPIC user data is traced in the normal CICS IS trace points.

YES
CICS suppresses user data from the CICS trace points.

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

DTIMOUT({NO|*mmss*})
Specifies whether deadlock timeout is to be applied to the task. If the task is suspended (for example, through lack of storage), a purge of the task is initiated if the task stays suspended for longer than the DTIMOUT value. If the purge leads to a transaction abend, the abend code used depends on which part of CICS suspended the task. When using CEDF, the user task, if possible, specifies DTIMOUT(NO) or a large value. This value is also now used as the timeout on all RLS file requests if DTIMOUT is nonzero; otherwise, the request gets the SIT FTIMEOUT value. FTIMEOUT applies to transactions that do not have a deadlock timeout interval active. If a time value is specified for the DTIMOUT keyword of the TRANSACTION definition, this value is used as the file timeout value for that transaction.

When using CEDF, if any DTIMOUT value has been specified for the user task, the DTIMOUT value is ignored while the user task is suspended and a CEDF task is active. Therefore, the suspended user task cannot end with a deadlock timeout (abend AKCS) while a CEDF task is waiting for a user response.

For DTIMOUT to be effective in non-RLS usage, set SPURGE to YES.

CICS inhibits deadlock timeout at certain points.

DTIMOUT is not triggered for terminal I/O waits. Because the relay transaction does not access resources after obtaining a session, it has little need for DTIMOUT except to trap suspended allocate requests. However, for I/O waits on a session, the RTIMOUT attribute can be specified on PROFILE definitions for transaction routing on IPIC or MRO sessions and mapped APPC connections.

You must define some transactions with a DTIMOUT value, because deadlock timeout is the mechanism that CICS uses to deal with short-on-storage (SOS) situations.

NO
The deadlock timeout feature is not required.

mmss
The length of time (*mm* for minutes and *ss* for seconds) after which deadlock timeout ends a suspended task. The maximum value that you can specify is 68 minutes (6800); this value is accurate to one second.

DUMP({YES|NO})
Specifies whether a call is to be made to the dump domain to produce a transaction dump if the transaction ends abnormally.

This operand has no effect on the following dump operations:

- An **EXEC CICS DUMP** command, which always produces a dump.
- The system dumps for dump codes AP0001 and SR0001 that CICS produces with ASRA, ARSB, or ASRD abends. If you specify NO on the transaction DUMP attribute, CICS suppresses the transaction dump, but not the system dump.

YES
CICS calls the dump domain to produce a transaction dump. Note that the final production or suppression of the transaction dump is controlled by the transaction dump table. For more information about the dump table, see The dump code options you can specify.

If no transaction dump table entry exists for the given dump code when a transaction abends, CICS creates a temporary entry for which the default is to produce a transaction dump.

You control dump table entries for transaction dumps using the **CEMT SET TRDUMPCODE** command or the SET TRANDUMPCODE SPI command.

NO
No call is made to the dump domain, suppressing any potential transaction dump.

DYNAMIC({NO|YES})
Specifies whether the transaction can be dynamically routed to a remote region, using the CICS dynamic transaction routing facility.

NO
Creates a local or remote definition according to the REMOTESYSTEM attribute.

YES
Allows the dynamic transaction routing program to determine the local or remote status dynamically at invocation time. For programming information about the dynamic transaction routing program, see Writing a dynamic routing program.

EXTSEC
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

INDOUBT
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS.

ISOLATE({YES|NO})
Specifies whether CICS is to isolate the transaction user-key task-lifetime storage to provide transaction-to-transaction protection. (See the TASKDATAKEY attribute for a description of user-key storage.) Isolation means that the user-key task-lifetime storage is protected from both reading and writing by the user-key programs of other transactions; that is, from programs defined with EXECKEY(USER). The effect of the ISOLATE attribute of storage access shows the effect of the ISOLATE attribute.

To use transaction isolation, you must set storage protection on with the STGPROT system initialization parameter and set transaction isolation on for the whole region with the TRANISO system initialization parameter.

Note:

1. The ISOLATE attribute does not provide any protection against application programs that run in CICS key; that is, from programs defined with EXECKEY(CICS).
2. VSAM nonshared resources (NSR) are not supported for transactions that use transaction isolation. You must specify ISOLATE(NO) when you define transactions that access VSAM files using NSR. You can also function ship the file request to a remote region. The DFHMIRS program that carries out the request is defined with an EXECKEY of CICS. A CICS-key program has read and write access to CICS-key and user-key storage of its own task and all other tasks, whether transaction isolation is active.
3. Transaction isolation does not apply to 64-bit storage.

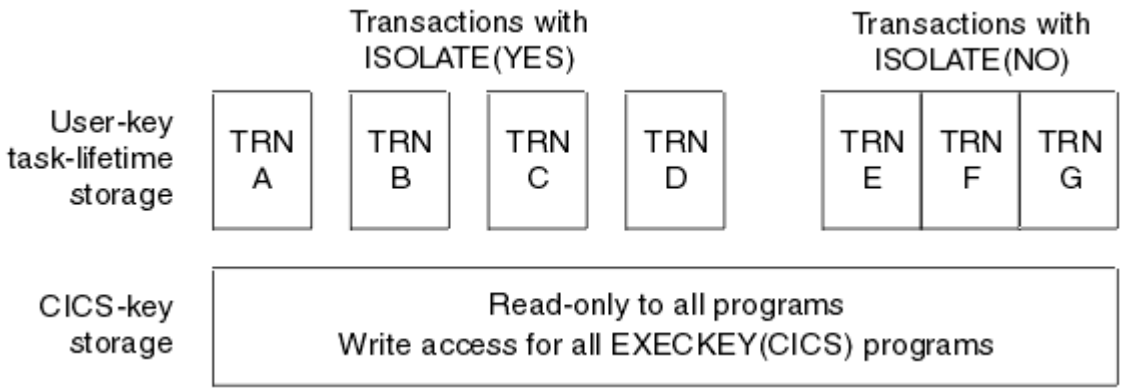
YES
The transaction user-key task-lifetime storage is isolated from the user-key programs of all other transactions; that is, from programs defined with EXECKEY(USER), but not from programs defined with EXECKEY(CICS).

Also, the user-key task-lifetime storage of *all* other transactions is protected *from* the user-key programs of transactions defined with ISOLATE(YES).

NO
If you specify ISOLATE(NO), the transaction task-lifetime storage is isolated from the user-key programs of those transactions defined with ISOLATE(YES). The transaction storage is not, however, isolated from user-key programs of other transactions that also specify ISOLATE(NO) because, with this option, the transactions are all allocated to the common subspace.

Note also that the user-key task-lifetime storage of all transactions defined with ISOLATE(YES) is protected *from* the user-key programs of transactions defined with ISOLATE(NO).

Specify ISOLATE(NO) for those transactions that share any part of their user-key task-lifetime storage.



This figure shows four transactions, A, B, C, and D, defined with the ISOLATE(YES), and three transactions, E, F, and G, defined with ISOLATE(NO):

- The user-key task-lifetime storage of each of the transactions A, B, C, and D is isolated from all other transactions.
- The user-key task-lifetime storage of transactions E, F, and G is accessible by all the user-key application programs of transactions E, F, and G, but is isolated from the user-key programs of transactions A, B, C, and D.
- All the transactions have read-only access to CICS-key storage.

Figure 5. The effect of the ISOLATE attribute of storage access

LOCALQ({NO|YES})
Specifies whether queuing on the local system is to be performed.

NO
No local queuing is to be performed.

YES
Local queuing can be attempted for a START command with NOCHECK option request when the system or IPIC connection is not available and the system name is valid.

A system is defined as not available in the following circumstances:

- The system is OUT OF SERVICE when the request is initiated.
- The attempt to initiate any session to the remote system fails and the corrective action taken by the abnormal condition program (DFHZNAC) or the node error program (DFHZNEP) is to place the system OUT OF SERVICE.
- No sessions to the remote system are immediately available, and your XISCONA global user exit program specifies that the request is not to be queued in the issuing region.

An IPIC connection is defined as not available in the following circumstances:

- The IPIC connection is not acquired.
- A session is not available and CICS does not queue the request for a new session.

Use local queuing only for those START commands that represent time-independent requests. The delay implied by local queuing affects the time at which the request is started. It is your responsibility to ensure that this condition is met.

If you specify LOCALQ(YES), you cannot specify ROUTABLE(YES).

You can use the global user exits, XISLCLQ or XISQLCL, to override the setting of the LOCALQ attribute. For programming information about the user exits in the intersystem communication program, see Global user exit points.

OTSTIMEOUT({NO|*hhmmss*})
Specifies, in hours, minutes, and seconds, the length of time for which an Object Transaction Service (OTS) transaction, created in an enterprise beans environment and running as a task under this CICS transaction, is allowed to run before the initiator of the OTS transaction must take a sync point or roll back the transaction. If the specified period expires, CICS purges the task.

The initiator of the OTS transaction can be one of these:

- The client of the enterprise bean.
- The EJB container. The container issues a sync point at the end of the bean method.
- A session bean that manages its own OTS transactions.

Methods of session beans that manage their own OTS transactions can override the default timeout value by using the `setTransactionTimeout` method of the `javax.Transaction.UserTransaction` interface.

NO
OTS transactions will not timeout. NO is the default.

hhmmss
The time (in HHMMSS format) before the task is purged. The maximum period is 24 hours (240000).

PARTITIONSET({*partitionset*|KEEP|OWN})
Specifies the name of the partition set that is to be the default application partition set. The name can be up to 8 characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

If you do not specify a partition set name or either of the reserved names, CICS destroys existing partitions before the first BMS output to the terminal from the transaction.

partitionset
CICS destroys existing partitions and loads the named partition set before the first BMS output to the terminal from the transaction. (Existing partitions are not destroyed if the terminal partition set matches the application partition set.)

This name must not be the same as that specified in PROGRAM(*name*).

KEEP
The transaction uses the application partition set for this terminal. This option is typically used for successor transactions in a chain of pseudoconversational transactions.

OWN
The transaction performs its own partition management.

PRIMEDSIZE
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS.

PRIORITY({1|*priority*})
Specifies the transaction priority. This 1- to 3-digit decimal value from 0 to 255 is used to establish the overall transaction processing priority. Transaction processing priority is equal to the sum of the terminal priority, transaction priority, and operator priority, not exceeding 255. The higher the number, the higher the priority.

PROFILE({DFHCICST|*profile*})
Is the name of the PROFILE definition that specifies the processing options used with the terminal that initiated the transaction.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

The default is DFHCICST.

The processing options provided by the default DFHCICST are shown in PROFILE definitions in group DFHISC. DFHCICST is not suitable for use with distributed transaction processing. Instead, specify DFHCICSA, which has INBFMH=ALL.

PROGRAM(*program*)
Specifies the name of the program to which CICS gives control to process this transaction. The name can be up to 8 characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

Ensure that this name is not the same as that specified in PARTITIONSET(name).

If a name is specified for REMOTESYSTEM, and it differs from that of the current system, you do not need to specify a name for PROGRAM. If, in these circumstances, you do specify a name for PROGRAM, it might be ignored.

If this transaction definition is for use on a remote program link request, the program name that you specify in this attribute must be the name of the CICS mirror program, DFHMIRS. See the TRANSID attribute on the PROGRAM definition in PROGRAM attributes.

REMOTENAME(*transaction*)
Specifies the name of this transaction as it is known in a remote system, if it is to run in a remote system or region using intersystem communication. The remote system can be another CICS region or an IMS system. REMOTENAME can be 1 - 4 characters in length if the REMOTESYSTEM attribute specifies another CICS region, or 1 - 8 characters in length if REMOTESYSTEM specifies an IMS system. IMS uses 8-character names and, if REMOTENAME has fewer than 8 characters, IMS translates it into a usable format.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

If you specify REMOTESYSTEM and omit REMOTENAME, the value of REMOTENAME defaults to the local name; that is, the TRANSACTION name on this definition. Note that the transaction does not have to be on the remote system or region.

REMOTESYSTEM(*connection*)
Specifies the name that identifies the intercommunication link on which the transaction attach request is sent.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

REMOTESYSTEM is one of these identifiers:

- For IPIC connections, the first 4 characters of the IPCONN name on the IPCONN definition, which is in service and acquired.
- For MRO and APPC connections, the CONNECTION name on the CONNECTION definition.

REMOTESYSTEM is used for CICS function request shipping (asynchronous processing and transaction routing).

Ensure that the REMOTESYSTEM options which are specified for a set of transactions do not refer to any other transaction in that set; that is, they are not self-referent, which can lead to looping starts of multiple mirror transactions.

RESSEC({NO|YES})
Specifies whether resource security checking is to be used for resources accessed by this transaction.

NO
All resources are available to any user who has the authority to use this transaction.

YES
An external security manager is used. For more details about external security checking, see Security of resource definitions

RESTART({NO|YES})
Specifies whether the transaction restart facility is to be used to restart those tasks that end abnormally and are later backed out by the dynamic transaction backout facility.

If RESTART(YES) is specified, the task that failed is restarted from the beginning of the initial program. If dynamic transaction backout fails, or if restart is suppressed dynamically, DFHPEP is called in the normal way. The transaction restart facility is especially useful in situations such as a program isolation deadlock, where the task can be restarted automatically rather than resubmitted manually. A terminal-initiated transaction is allowed to restart during CICS shutdown even if SHUTDOWN(DISABLED) is specified. For more details about automatic transaction restart, see Splitting the application into transactions .

NO
The restart facility is not required.

YES
The restart facility is to be used.

ROUTABLE({NO|YES})
Applies to terminal-related transactions only. Specifies whether, if the transaction is the subject of an eligible **EXEC CICS START** command, it is routed using the enhanced routing method.

NO
If the transaction is the subject of a START command, it is routed using the traditional method.

YES
If the transaction is the subject of an eligible START command, it is routed using the enhanced method.

If you specify ROUTABLE(YES), you cannot specify LOCALQ(YES).

For details of the enhanced and traditional methods of routing transactions called by **EXEC CICS START** commands, see Routing transactions invoked by START commands.

RSL
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS.

RUNAWAY({SYSTEM|0|250-2700000})
The time, in milliseconds, for which any task running under this transaction definition can have control of the processor before it is assumed to be in a runaway condition (logical loop). When this interval expires, CICS can abnormally end the task.

SYSTEM
CICS is to use the ICVR system initialization parameter value as the runaway time limit for this transaction.

0
No limit applies and no runaway task detection is required for the transaction.

250-2700000
The runaway time limit in the range 250 - 2700000. When checking whether a task is in a runaway condition, CICS rounds the value you specify downwards, to a multiple of 250.

SHUTDOWN({DISABLED|ENABLED})
Applies to all transactions, and specifies whether the transaction can be run during CICS shutdown. This option supplements the XLT option of the **PERFORM SHUTDOWN** command. For a transaction to be attached during shutdown, it must either be defined as SHUTDOWN(ENABLED) or, in the case of terminal-based transactions, be named in the XLT specified in the **PERFORM SHUTDOWN** command.

DISABLED
The transaction is disabled from running during CICS shutdown.

ENABLED
The transaction is enabled to run during CICS shutdown.

SPURGE({YES|NO})
Specifies whether the transaction is initially **system purgeable** or not.

SPURGE=NO prevents a transaction being purged by these means:

- Deadlock timeout (DTIMOUT)
- An **EXEC CICS ... PURGE** command
- TWAOCT (Cancel Task) being set in the node error program (NEP)
- A **CEMT SET ... PURGE** command

SPURGE=YES allows such purges to go ahead as far as the user is concerned. CICS might, however, prevent the purge if it is not safe to allow a purge at the point the transaction has reached.

Note that SPURGE=NO does not prevent a transaction being purged by the read timeout (RTIMOUT) facility, an EXEC CICS SET ... FORCEPURGE command, or a CEMT SET TRANSACTION(tranid) FORCEPURGE command. SPURGE determines only the initial value, which can be changed by the transaction while it is running.

YES
The transaction is initially system purgeable.

NO
The transaction is not initially system purgeable.

STATUS({ENABLED|DISABLED})
Specifies the transaction status.

ENABLED
Allows the transaction to be run normally.

DISABLED
Prevents the transaction from running.

This attribute is ignored for TRANSACTION resources that are dynamically generated by a CICS bundle. The initial status of a TRANSACTION resource is derived from the initial status of the bundle that defines the resource.

STORAGECLEAR({NO|YES})
Specifies whether task-lifetime storage for this transaction is to be cleared on release. Use STORAGECLEAR to prevent other tasks accidentally viewing any confidential or sensitive data that was being stored by this transaction in task-lifetime storage.

TASKDATAKEY({USER|CICS})
Specifies the storage key of the storage that CICS allocates at task initialization for the duration of the task (task-lifetime storage), and which is accessible by the application. These storage areas are the EXEC interface block (EIB) and the transaction work area (TWA).

TASKDATAKEY also specifies the key of the storage that CICS obtains on behalf of all programs that run under the transaction. The program-related storage that CICS allocates in the specified key includes these items:

- The copies of working storage that CICS obtains for each run of an application program.
- The storage that CICS obtains for the program in response to implicit and explicit GETMAIN requests. For example, the program can request storage by a GETMAIN or GETMAIN64 command, or as a result of the SET option on other CICS commands.

You must specify TASKDATAKEY(USER) if any of the programs in the transaction is defined with EXECKEY(USER). If you specify TASKDATAKEY(CICS) for a transaction, an attempt to run any program in user key under this transaction leads to a task abend, with abend code AEZD.

USER
CICS obtains user-key storage for this transaction. Application programs running in any key can both read and modify these storage areas.

User-key programs of transactions defined with ISOLATE(YES) have access only to the user-key task-lifetime storage of their own tasks.

User-key programs of transactions defined with ISOLATE(NO) also have access to the user-key task-lifetime storage of other tasks defined with ISOLATE(NO).

See the description of the EXECKEY attribute on the PROGRAM definition for more information about task storage protection.

CICS
CICS obtains CICS-key storage for this transaction. Application programs running in CICS key can both read and modify these storage areas. Application programs running in user key can only read these storage areas.

TASKDATALOC({ANY|BELOW})
Specifies whether task-lifetime storage acquired by CICS for the duration of the transaction can be located above the 16 MB line in virtual storage. These areas, which relate to specific CICS tasks, include the EXEC interface block (EIB) and the transaction work area (TWA).

You must specify TASKDATALOC(BELOW) if any of the programs that make up the transaction runs in 24-bit addressing mode. This restriction also applies to task-related user exits running on behalf of the transaction).

For transactions that do not satisfy any of these conditions, you can specify ANY to obtain the associated virtual storage constraint relief.

CICS monitors the use of TASKDATALOC(ANY), particularly the following actions:

- An attempt to call an AMODE 24 program running under a transaction defined with TASKDATALOC(ANY) results in an AEZC abend.
- An attempt to issue an EXEC CICS command or call a task-related user exit while running AMODE(24) with TASKDATALOC(ANY) specified results in an AEZA abend.
- An AMODE 31 program running as a transaction with TASKDATALOC(ANY), which attempts to call a task-related user exit that is forced to run AMODE(24), results in an AEZB abend.
- If a task-related user exit that is forced to run in AMODE 24 is enabled for task start, CICS forces TASKDATALOC(BELOW) for all transactions for the remainder of the CICS run.

ANY
Storage areas that CICS acquires for the transaction can be located above or below the 16 MB line in virtual storage.

BELOW
Storage areas that CICS acquires for the transaction must be located below the 16 MB line.

TASKREQ(*value*)
Specifies whether a transaction is to be initiated by pressing a function (F or PF) key, by using a light pen, or by using a card. Possible values are as follows:

- PA1, PA2, or PA3 for PA keys.
- PF1 through PF24 for function keys.
- OPID for the operator identification card reader.
- LPA for a light-pen-detectable field on a 3270 device.
- MSRE for the 10/63 character magnetic slot reader.

For PA and PF keys, be aware of the following:

- If a PA or PF key is specified in the PRINT system initialization parameter, you cannot use the same PF key as the TASKREQ to initiate a transaction.
- PA or PF keys specified in the SKR*xxxx* system initialization parameter as page retrieval keys are interpreted as such during a page retrieval session. You can use the same keys to initiate transactions at other times. Define the keys with the following values:

```
TASKREQ=KEY-ID
PROGRAM=DFHTPR
TWASIZE=1024
TPURGE=NO
SPURGE=NO
```

- If you define a transaction with PROGRAM(DFHTPR) and define a TASKREQ key, the key initiates the transaction and opens the page retrieval session at the same time.

TCLASS
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS.

TPNAME(*name*)
Specifies the name of the transaction that can be used by an APPC partner if the 4-character length limitation of the TRANSACTION attribute is too restrictive. This name can be up to 64 characters in length.

Acceptable characters:

```
A-Z a-z 0-9 $ @ ♯ . / - _ % & ? ! : | " = ¬ , ; < >
```

If this range of characters is not sufficient for a name that you want to specify, you can use the XTPNAME attribute instead of TPNAME.

TPURGE({YES|NO})
Specifies, for non-z/OS Communications Server terminals only, whether the transaction can be purged because of a terminal error.

YES
The task can be purged when a terminal error occurs.

NO
The task cannot be purged when a terminal error occurs. Manual intervention by the main terminal operator is required when this happens.

TRACE({YES|NO})
Specifies whether the activity of this transaction is to be traced.

YES
Trace the activity for this transaction.

NO
Do not trace the activity for this transaction.

The CICS-provided transaction definitions for CEDF and CSGM specify TRACE(NO).

TRANCLASS(DFHTCL00|*tranclass*)
Specifies the name of the transaction class to which the transaction belongs. Transactions belonging to a transaction class are subject to scheduling constraints before they are allowed to run. The

reserved TRANCLASS name DFHTCL00 is used to indicate that the transaction does not belong to any transaction class.

If a transaction is run and its associated TRANCLASS definition is not installed, the transaction runs without any of the scheduling constraints specified in the TRANCLASS. Message DFHXM0212 is issued as a warning.

TRANCLASS can be up to 8 characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

TRANSACTION(*name*)
Specifies the name of the transaction or transaction identifier (TRANSID). The name can be up to 4 characters in length.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

It is preferable to not use names that start with C because this character is reserved for use by CICS. CICS might delete or discard such resources and replace them with CICS resources when you upgrade or provide new function or fixes.

Note:

1. If you use a comma (,) in a name, you cannot use commands such as these:

   ```
   CEMT INQUIRE TRANSACTION(value1,value2)
   CEMT SET     TRANSACTION(value1,value2)
   ```

   where the comma serves as a list delimiter. See List of resource identifiers for information about using lists of resource identifiers.
2. If you protect your transient data queues using RACF, avoid using % and & in the name. RACF commands assign a special meaning to these characters when they are used in a profile name. See Transaction security.

If you want to use other special characters in a transaction identifier, use the XTRANID attribute to specify another name that can be used to initiate the transaction. You must also specify a TRANSACTION name, because this name is the one by which the TRANSACTION definition is known on the CSD file.

When defining a transaction, you must also name either a PROGRAM or a REMOTESYSTEM.

TRANSEC
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS.

TRPROF({DFHCICSS|*profile*})
Specifies the name of the PROFILE for the session that carries intersystem flows during ISC transaction routing. The name can be up to 8 characters in length.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

You can specify this option only for remote transactions.

TWASIZE({0|number})
Specifies the size in bytes of the transaction work area to be acquired for this transaction. Specify a 1- to 5-digit decimal value in the range 0 - 32767.

Note:

1. Your storage might be corrupted if your TWASIZE is too small.
2. Do not change the TWASIZE of the CICS-supplied transactions.

WAIT({YES|NO})
Specifies whether an indoubt unit of work (UOW) is to wait, pending recovery from a failure that occurs after the UOW has entered the indoubt state.

Old-style transaction definitions using INDOUBT(WAIT) are accepted by CICS, and are interpreted as WAIT(YES) ACTION(BACKOUT).

YES
The UOW is to wait, pending recovery from the failure, to resolve its indoubt state and determine whether recoverable resources are to be backed out or committed. In other words, the UOW is to be *shunted*.

Recoverable resources can include one of the following:

- DBCTL databases
- DB2 databases
- Temporary storage queues
- Logically recoverable intrapartition transient data queues that specify WAIT(YES) in the TDQUEUE definition
- VSAM data sets
- BDAM data sets

The WAIT(YES) option takes effect *if none of the following applies*:

- The transaction has subordinate MRO sessions to back-level systems.
- The transaction has LU6.1 subordinate sessions. Note that, in this context, LU6.1 IMS sessions are not subordinates.
- The transaction has more than one session and its coordinator session is to a back-level system, or LU6.1.
- The task-related user exits attached to the transaction do not support the CICS indoubt protocols.

If none of the previous exceptions applies, but you have subordinate LU6.2 sessions to systems other than CICS Transaction Server for z/OS systems that do not use the CICS Transaction Server for z/OS indoubt architecture, CICS can indicate that the subordinate must wait by forcing session outage.

If any resources cannot wait for indoubt resolution by the coordinator, a decision is taken for the transaction in accordance with the ACTION attribute. In practice, the only circumstances that force decisions in this way are updates to transient data queues with WAIT(NO) specified in the TDQUEUE definition, and installations of terminal-related resources. The latter are typically installed using an INSTALL command.

Table 4 on page 210 shows how the WAIT attribute defined on a TRANSACTION definition and a logically recoverable TDQUEUE definition are resolved when they conflict.

NO
The UOW is not to wait. CICS immediately takes whatever action is specified on the ACTION attribute.

Table 4. Resolution of WAIT attributes on TRANSACTION and TDQUEUE definitions			
WAIT attribute of TDQUEUE definition	**WAITACTION attribute of TDQUEUE definition**	**WAIT attribute of TRANSACTION definition**	**Action**
NO	not applicable	YES	The TD WAIT(NO) overrides WAIT(YES) on the TRANSACTION definition. The UOW is forced either to commit or back out, in accordance with the ACTION attributes of the transaction.
NO	not applicable	NO	The UOW is forced either to commit or back out, in accordance with the ACTION attributes of the transaction.
YES	QUEUE	YES	The UOW waits; that is, it is shunted. A request from another task for a lock on the TD queue must wait, and is queued by CICS.
YES	QUEUE	NO	The transaction WAIT(NO) overrides the TDQUEUE definition. The UOW is forced either to commit or back out, in accordance with the ACTION attributes of the transaction.
YES	REJECT	YES	The UOW waits; that is, it is shunted. A request from another task for a lock on the TD queue is rejected with the LOCKED condition.
YES	REJECT	NO	The transaction WAIT(NO) overrides the TDQUEUE definition. The UOW is forced either to commit or back out, in accordance with the ACTION attributes of the transaction.

If the UOW references more than one transient data queue, and the queues have inconsistent WAIT options, WAIT(NO) always takes precedence and overrides a WAIT(YES). Therefore, a WAIT(NO) on one TDQUEUE definition forces a failed indoubt UOW to take either the BACKOUT or COMMIT attribute defined on the TRANSACTION definition of the UOW.

WAITTIME({00,00,00|*dd,hh,mm*})
Specifies how long a transaction is to wait before taking a decision about an indoubt unit of work, based on what is specified in the ACTION attribute.

00,00,00
The transaction waits indefinitely.

dd,hh,mm
The time, in days, hours, and minutes, for which the transaction is to wait. The maximum value is 99,23,59.

WAITTIME takes effect only if WAIT(YES) is specified.

XTPNAME(*value*)
You can use this attribute as an alternative to TPNAME. Enter a hexadecimal string up to 128 characters in length, representing the name of the transaction that can be used by an APPC partner. All hexadecimal combinations are acceptable *except* X'40'. To specify an XTPNAME more than 72 characters long to DFHCSDUP, put an asterisk in column 72. The asterisk causes the following line to be concatenated to the current line.

XTRANID(*xtranid*)
Specifies another name to be used instead of the TRANSACTION name for initiating transactions. The name can be up to 8 hexadecimal digits in length. Because XTRANID is specified in hexadecimal form, you can use a name that contains characters that you cannot specify in the TRANSACTION attribute.

See also TASKREQ, another transaction alias that can be specified.

value
A 4-byte transaction identifier in hexadecimal notation; the identifier therefore uses up to 8 hexadecimal digits. If you specify fewer than 8 hexadecimal digits, the identifier is padded on the right with blanks.

Certain values are reserved for use by CICS, and so the values that you can specify are restricted:

- The first byte must not be X'C3'.
- The first byte must not be less than or equal to X'40'.
- The value must not be X'00000000'.
- The last 3 bytes must not be X'FFFFFF'.

Avoid using values in the range X'00' through X'3F' in the second, third, and fourth bytes if the transaction is to be attached by unsolicited data received from a terminal defined as a 3270 device, because CICS interprets these values as control characters, and not as part of the transaction identifier. For example, if you issue EXEC CICS RETURN or EXEC CICS START and specify TRANSID(X'41303238'), the correct transaction is attached. However, if you issue EXEC CICS RETURN without specifying a TRANSID, and the 3270 device transmits data that begins with X'41303238', CICS attempts to attach a transaction as if X'41404040' had been transmitted.

Transaction resource definitions

Transaction definitions describe how transactions are to run in a CICS region.

TSMODEL resources

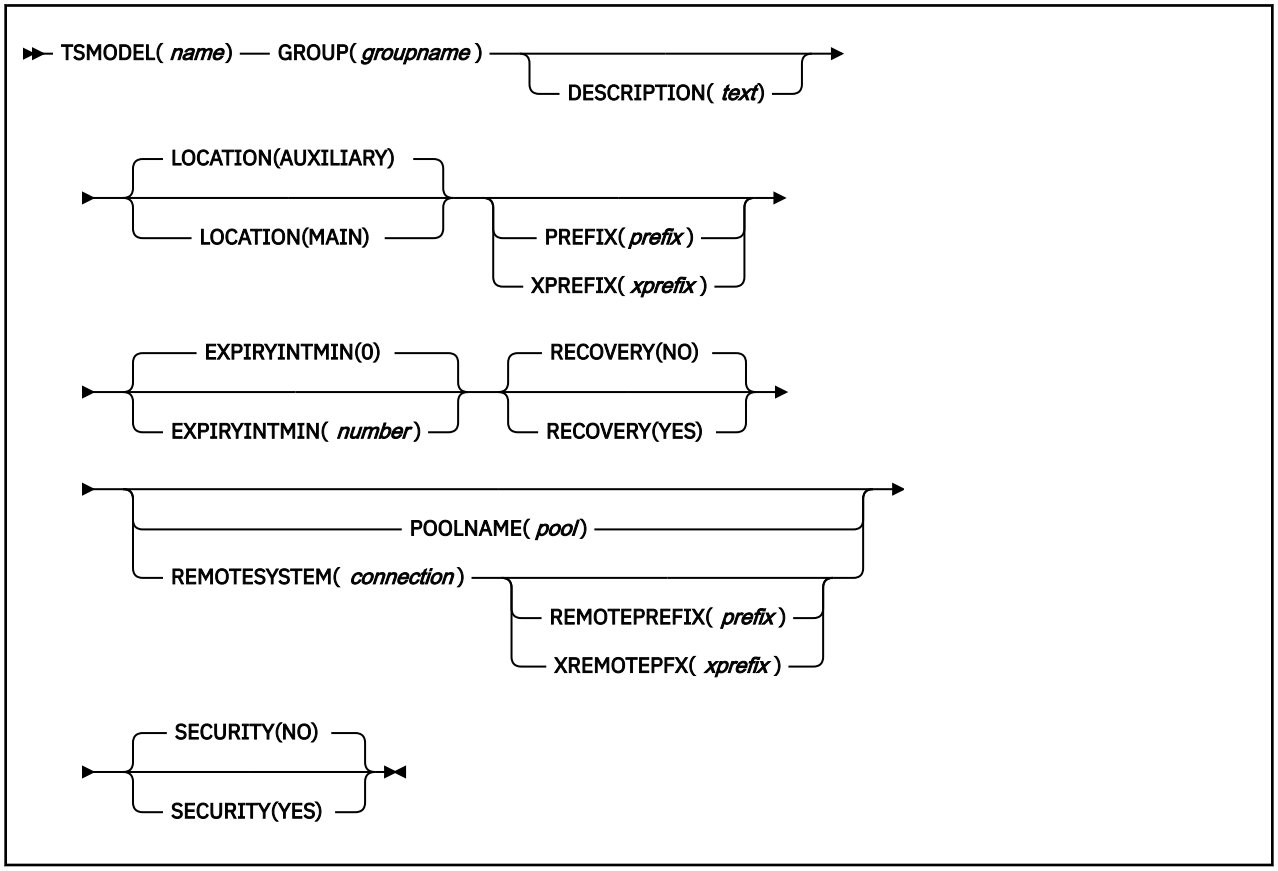
A temporary storage model specifies the properties of a set of TS queues. Individual TS queues are associated with a TSMODEL by the *prefix*, a character string that matches the leading characters of the queue name.

You can also map names directly to a shared TS pool (without the need for a shared sysid).

Note: CICS takes default actions on a region where a TSMODEL is not defined. This means that if you have an AOR and a QOR, and a TSMODEL defined in the AOR directs requests to the QOR, then unless a corresponding TSMODEL exists in the QOR, some queue attributes are taken from default values. For example, the location of a queue (MAIN or AUX) is determined from default settings within CICS. If there is no matching model, the location that is specified in the EXEC CICS command is used; if there is a model match, the location in this is used.

TSMODEL attributes

Describes the syntax and attributes of the TSMODEL resource.



DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

EXPIRYINTMIN({0|*number*})
Specifies the expiry interval, in minutes, for a temporary storage queue that matches this model. CICS uses the value rounded up to the nearest multiple of 10 minutes. The interval count begins after each use of the temporary storage queue. If the queue is not used again before the expiry interval is reached, the queue becomes eligible for CICS to delete it automatically.

0
No expiry interval applies to temporary storage queues that match this model, and they are never eligible for automatic deletion. This setting is the default.

number
An expiry interval in minutes, in the range 1 - 900000. After this expiry interval, a temporary storage queue that matches this model becomes eligible for automatic deletion if it has not been used again.

Expiry intervals apply to temporary storage queues in the following locations:

- Main temporary storage in the local CICS region.
- Nonrecoverable auxiliary temporary storage (DFHTEMP data set) associated with the local CICS region.
- Temporary storage queues in shared temporary storage pools (CICS TS 5.2 and higher).

Expiry intervals do not apply to the following types of temporary storage queues, so CICS never deletes them automatically:

- Queues in auxiliary temporary storage that are defined as recoverable.
- Queues in a remote CICS region. To make CICS delete remote temporary storage queues, specify an expiry interval in a suitable TSMODEL resource definition in the region that owns the queues.
- Queues that CICS creates for its own use.

If you change the expiry interval in a TSMODEL resource definition, existing temporary storage queues that match the model are not affected. Those queues continue to use the expiry interval that applied when they were created. If all the TSMODEL resource definitions with a nonzero expiry interval are deleted from a CICS region, CICS stops scanning for expired temporary storage queues.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

LOCATION({AUXILIARY|MAIN})
Specifies whether the queue is to be held in auxiliary or main storage:

AUXILIARY
Queues matching this model are to be held in auxiliary temporary storage. Whatever is specified on the API request is disregarded.

MAIN
Queues matching this model are to be held in main temporary storage. Whatever is specified on the API request is disregarded.

LOCATION is ignored for temporary storage models that relate to remote queues and to queues in shared temporary storage pools. Using LOCATION in a TSMODEL resource definition for a remote queue allows the same definition to be installed in both a local and remote region. See Shared resources for intercommunication.

POOLNAME(*pool*)
Specifies the name of the shared TS pool definition that you want to use with this TSMODEL resource definition. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ # _
```

Any lowercase characters that you enter are converted to uppercase.

Embedded blanks are not acceptable and a name consisting entirely of blanks is treated as though no pool name had been supplied.

You cannot specify POOLNAME if REMOTESYSTEM is also specified.

CICS does not search for a matching TSMODEL resource definition if an application program specifies a SYSID on the EXEC CICS temporary storage command, or if a SYSID is added by an XTSEREQ global user exit program. To enable CICS to find the name of a temporary storage data sharing pool when the application program explicitly specifies a SYSID, you must use a temporary storage table (TST) with a suitable TYPE=SHARED entry.

PREFIX(*prefix*)
Specifies the character string that CICS uses to identify matching temporary storage queues. The prefix can be up to 16 characters in length.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

In the simplest case, a prefix matches the TS queue names that start with the prefix. For example, the prefix ABCD matches queue names ABCD, ABCDE, and ABCD99.

You can use the generic character (+) one or more times within the prefix to match any character in a TS queue name. For example, the prefix A++D matches queue names: ABCD, A99D, and ABCD99. It does not match ABD. You do not need to specify + characters at the end of a prefix; you can think of a prefix as being padded on the right with + characters, up to the maximum length of a TS queue name (16 characters). For example, a prefix of ABC is exactly equivalent to a prefix of ABC+ or ABC+++++.

When you use the + character in a prefix, you might find that, considered individually, more than one prefix matches a TS queue name. In this situation, CICS uses the following rules to select the matching TS model:

- Characters that are exactly specified in the prefix are a stronger match than the + character.
- Characters are compared one at a time, from left to right, and matching characters are a stronger match the further to the left they are.

For example:

- Queue name ABCD99 matches prefix ABCD rather than AB+D: In the first prefix, the four matching characters are specified exactly; in the second, only three characters are specified exactly.
- Queue name ABCD99 matches prefix ABC+ rather than AB+D: In both prefixes, the same number of matching characters are specified exactly. However, the third character that matches exactly is further to the left in the first prefix.

Creating a TSMODEL definition with a blank prefix can produce unexpected results. A TSMODEL definition with a blank prefix matches any queue name that is not matched by any other TSMODEL definition.

To enable CICS to find the name of a temporary storage data sharing pool when the application program explicitly specifies a SYSID, you must use a temporary storage table (TST) with a suitable TYPE=SHARED entry.

RECOVERY({NO|YES})
Specifies whether queues matching this model are to be recoverable.

NO
Queues matching this model are unrecoverable.

YES
Queues matching this model are recoverable.

RECOVERY(YES) is not allowed with LOCATION(MAIN).

REMOTEPREFIX(*prefix*)
Specifies the character string that CICS uses to identify matching temporary storage queues in the remote system. The prefix can be up to 16 characters in length.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

REMOTEPREFIX is not allowed unless REMOTESYSTEM is also specified. The length of the REMOTEPREFIX attribute must be the same as the PREFIX attribute.

The rules for character matching in the REMOTEPREFIX attribute are the same as for the PREFIX attribute. If you use + characters for generic matching, they must be in the same position in PREFIX and REMOTEPREFIX. For example:

PREFIX: A++D
REMOTEPREFIX: X++Y

REMOTESYSTEM(*connection*)
Specifies the name of the connection that links the local system to the remote system where the temporary storage queue resides.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

Embedded blanks are not acceptable and a name consisting entirely of blanks is treated as though no remote system had been specified.

REMOTESYSTEM and POOLNAME are mutually exclusive. If REMOTESYSTEM is specified, POOLNAME is ignored.

SECURITY({NO|YES})
Specifies whether security checking is to be performed for queues matching this model.

NO
Security checking is not to be performed for queues matching this model.

YES
Security checking is to be performed for queues matching this model.

For more information, see Security for temporary storage.

TSMODEL(*name*)
Specifies the name of this TSMODEL resource definition. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

Embedded blanks are not acceptable.

This name is used to identify the TSMODEL resource definition on the CSD file. It is not used within the active CICS system.

XPREFIX(*xprefix*)
Can be used as an alternative to PREFIX. Enter a hexadecimal string, up to 32 characters in length, that is to be used as the prefix for this model. Because XPREFIX is specified in hexadecimal form, you can use a name that contains characters that you cannot enter in the PREFIX attribute.

To specify generic characters in the XPREFIX attribute, code X'4E'.

The rules for matching the XPREFIX to TS queue names are the same as they are for the PREFIX attribute.

XREMOTEPFX(*xprefix*)
Can be used as an alternative to REMOTEPREFIX. Enter a hexadecimal string, up to 32 characters in length, that is to be used as the prefix on the remote system. Because XREMOTEPFX is specified

in hexadecimal form, you can use a name that contains characters that you cannot enter in the REMOTEPREFIX attribute.

To specify generic characters in the XPREFIX attribute, code X'4E'.

The rules for matching the XREMOTEPFX to TS queue names are the same as they are for the REMOTEPREFIX attribute.

Temporary storage model definitions

Temporary storage definitions describe the attributes that govern the characteristics of CICS temporary storage queues.

TYPETERM resources

A TYPETERM resource defines a set of attributes that are common to a group of terminals.

The resource is a logical extension of the TERMINAL resource. If you have a number of terminals with the same properties, you would define one TYPETERM with the required values, and then name that TYPETERM in each TERMINAL definition (or in the autoinstall model definition if you are using autoinstall).

Each TERMINAL definition must name a TYPETERM definition. This single attribute represents many other characteristics, and thus can save considerable effort, and reduce the chance of making mistakes. TYPETERMs make it easier to define your terminals if you have many terminals of the same kind.

Two TYPETERM attributes are worthy of note here, because they further simplify the terminal definition process:

DEVICE
QUERY

DEVICE
specifies the **device type** that the TYPETERM represents. This is a key attribute, because the default values for a number of other attributes depend on the value you supply for it:

- Some attributes are always the same for every device of the same type. You do not need to define all these attributes yourself, because RDO knows what they are. All you need to tell RDO is the device type of your terminals, when you define the TYPETERM for them. Values for the fixed attributes are supplied automatically.
- Other attributes are given default values, depending on the device type. However, you do not have to use the values that CICS supplies; you can specify different values if you want. If you change the device type in a TYPETERM definition, the default values are not reset.

You must supply a value for the DEVICE attribute, because there is no default.

For a list of terminals supported by RDO, see Devices supported. There is also a list of valid values for the DEVICE attribute of TYPETERM in Default values for TYPETERM attributes. This shows you the other attribute values supplied for different device types. In some cases, these values depend also on your values for SESSIONTYPE and TERMMODEL, but these too have defaults that depend on the DEVICE specified.

Apart from ordinary display devices, printers, and other more specialized input and output devices, you can create a TYPETERM definition for your CICS consoles.

QUERY
The QUERY attribute allows you to leave some features of your terminals undefined until they are connected. Information about these attributes can then be obtained by CICS itself using the QUERY structured field.

All attributes for which you can use QUERY are also TYPETERM attributes. They are:

ALTPAGE
ALTSCREEN
APLTEXT

BACKTRANS
CGCSGID
COLOR
EXTENDEDDS
HILIGHT
MSRCONTROL
OBFORMAT
OUTLINE
PARTITIONS
PROGSYMBOLS
SOSI
VALIDATION

The use of QUERY overrides any value that is explicitly defined for any of the TYPETERM attributes listed here, **except ALTSCREEN**. QUERY-supplied ALTSCREEN values are used only if no ALTSCREEN value is explicitly defined in the TYPETERM.

You can use QUERY for 3270 devices with the extended 3270 data stream. The DEVICE types for which you can use QUERY are:

3270
3270P
LUTYPE2
LUTYPE3
SCSPRINT

You can specify that QUERY be used in one of two ways:

- QUERY(COLD) specifies that the QUERY is to be issued only when the terminal is first connected after an initial or a cold start.
- QUERY(ALL) specifies that the QUERY is to be issued each time the terminal is connected.

The QUERY function is particularly useful with configurable devices, such as the IBM Personal System/2 (PS/2) and the IBM 3290. It enables you to reconfigure the device between logging off and logging on to CICS, without having to change any resource definitions.

The QUERY function is also particularly useful when used in conjunction with autoinstall.

Note that the QUERY facility obtains only the information required by CICS. If an application program needs to determine other device characteristics, it still needs to send a QUERY structured field and analyze the reply.

To summarize, you may need only one TYPETERM definition for each device type. If the attributes that can be determined by QUERY differ among the terminals, you still need only one TYPETERM for each device type. If other attributes of your terminals vary, you may need more than one TYPETERM definition for a device type.

There are some CICS-supplied TYPETERM definitions suitable for the more frequently used terminals. These are described in TYPETERM definitions in group DFHTYPE.

When all your terminals are basically the same, you can have only one TYPETERM definition, and one TERMINAL definition with AUTINSTMODEL(YES). You might like to use QUERY to deal with different features used by your terminals.

Default values for TYPETERM attributes

When you specify the DEVICE, SESSIONTYPE and TERMMODEL in a TYPETERM definition, CICS supplies default values for many of the other attributes.

The default values are shown in Table 5 on page 218. Note that for some attributes, the supplied values are mandatory, and you cannot change them.

Table 5. Default values for TYPETERM attributes			
DEVICE	**SESSIONTYPE**	**TERMMODEL**	**Default values**
3270 3277 L3277 See note “1” on page 225		1	DEFSCREEN(12,40) PAGESIZE(12,40) AUTOPAGE(N) BRACKET(Y) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3270		2 (default value)	DEFSCREEN(24,80) PAGESIZE(24,80) AUTOPAGE(N) BRACKET(Y) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3275		1	DEFSCREEN(12,40) PAGESIZE(12,40) AUTOPAGE(N) BRACKET(Y) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3275		2 (default value)	DEFSCREEN(24,80) PAGESIZE(24,80) AUTOPAGE(N) BRACKET(Y) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3270P 3284 L3284 3286 L3286 See note “2” on page 225		1	DEFSCREEN(12,40) PAGESIZE(12,40) AUTOPAGE(Y) BRACKET(Y) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3270P		2 (default value)	DEFSCREEN(24,80) PAGESIZE(24,80) AUTOPAGE(Y) BRACKET(Y) BUILDCHAIN(N) ROUTEDMSGS(ALL)
APPC			DEFSCREEN(0,0) PAGESIZE(1,40) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(Y) (mandatory) ROUTEDMSGS(NONE) (mandatory)

Table 5. Default values for TYPETERM attributes (continued)

DEVICE	SESSIONTYPE	TERMMODEL	Default values
CONSOLE			DEFSCREEN(0,0) (mandatory) PAGESIZE(1,124) (mandatory) AUTOPAGE(N) BRACKET(N) (mandatory) BUILDCHAIN(N) (mandatory) ROUTEDMSGS(NONE)
LUTYPE2		1	DEFSCREEN(12,40) PAGESIZE(12,40) AUTOPAGE(N) BRACKET(Y) (mandatory) BUILDCHAIN(Y) (mandatory) ROUTEDMSGS(ALL)
LUTYPE2		2 (default value)	DEFSCREEN(24,80) PAGESIZE(24,80) AUTOPAGE(N) BRACKET(Y) (mandatory) BUILDCHAIN(Y) (mandatory) ROUTEDMSGS(ALL)
LUTYPE3		1	DEFSCREEN(12,40) PAGESIZE(12,40) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
LUTYPE3		2 (default value)	DEFSCREEN(24,80) PAGESIZE(24,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
LUTYPE4			DEFSCREEN(0,0) PAGESIZE(50,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
BCHLU	(default value)		DEFSCREEN(0,0) PAGESIZE(12,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)

Table 5. Default values for TYPETERM attributes (continued)			
DEVICE	**SESSIONTYPE**	**TERMMODEL**	**Default values**
BCHLU	BATCHDI		DEFSCREEN(0,0) PAGESIZE(12,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
BCHLU	USERPROG		DEFSCREEN(0,0) PAGESIZE(12,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
INTLU			DEFSCREEN(0,0) PAGESIZE(12,80) AUTOPAGE(Y) BRACKET(Y) BUILDCHAIN(N) ROUTEDMSGS(ALL)
SCSPRINT			DEFSCREEN(0,0) (mandatory) PAGESIZE(24,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
TLX or TWX	CONTLU (default value)		DEFSCREEN(0,0) PAGESIZE(1,40) AUTOPAGE(Y) BRACKET(Y) BUILDCHAIN(N) ROUTEDMSGS(ALL)
TLX or TWX	INTLU		DEFSCREEN(0,0) PAGESIZE(1,40) AUTOPAGE(Y) BRACKET(Y) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3600	(default value)		DEFSCREEN(0,0) PAGESIZE(1,40) AUTOPAGE(Y) BRACKET(Y) BUILDCHAIN(N) ROUTEDMSGS(ALL)

Table 5. Default values for TYPETERM attributes (continued)

DEVICE	SESSIONTYPE	TERMMODEL	Default values
3600	PIPELINE		DEFSCREEN(0,0) PAGESIZE(6,30) AUTOPAGE(Y) BRACKET(Y) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3614			DEFSCREEN(0,0) PAGESIZE(1,40) AUTOPAGE(Y) BRACKET(Y) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3650	USERPROG (default value)		DEFSCREEN(0,0) PAGESIZE(3,80) AUTOPAGE(Y) BRACKET(Y) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3650	3270		DEFSCREEN(12,40) PAGESIZE(23,80) AUTOPAGE(Y) BRACKET(Y) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3650	3653		DEFSCREEN(0,0) PAGESIZE(6,30) AUTOPAGE(Y) BRACKET(Y) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3650	PIPELINE		DEFSCREEN(0,0) PAGESIZE(6,30) AUTOPAGE(Y) BRACKET(Y) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3767			DEFSCREEN(0,0) PAGESIZE(12,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)

Table 5. Default values for TYPETERM attributes (continued)			
DEVICE	**SESSIONTYPE**	**TERMMODEL**	**Default values**
3767C			DEFSCREEN(0,0) PAGESIZE(12,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3767I			DEFSCREEN(0,0) PAGESIZE(12,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3770	(default value)		DEFSCREEN(0,0) PAGESIZE(12,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3770	USERPROG		DEFSCREEN(0,0) PAGESIZE(12,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3770	BATCHDI		DEFSCREEN(0,0) PAGESIZE(12,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3770B	(default value)		DEFSCREEN(0,0) PAGESIZE(12,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3770B	BATCHDI		DEFSCREEN(0,0) PAGESIZE(12,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)

Table 5. Default values for TYPETERM attributes (continued)			
DEVICE	**SESSIONTYPE**	**TERMMODEL**	**Default values**
3770B	USERPROG		DEFSCREEN(0,0) PAGESIZE(12,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3770C			DEFSCREEN(0,0) PAGESIZE(12,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3770I			DEFSCREEN(0,0) PAGESIZE(12,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3790	(default value)		DEFSCREEN(0,0) PAGESIZE(1,40) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(NONE) (mandatory)
3790	SCSPRINT		DEFSCREEN(0,0) PAGESIZE(24,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3790	USERPROG		DEFSCREEN(0,0) PAGESIZE(12,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3790	BATCHDI		DEFSCREEN(0,0) PAGESIZE(12,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)

Table 5. Default values for TYPETERM attributes (continued)			
DEVICE	**SESSIONTYPE**	**TERMMODEL**	**Default values**
3790	3277CM See note “3” on page 225	1	DEFSCREEN(12,40) PAGESIZE(12,40) AUTOPAGE(N) BRACKET(Y) (mandatory) BUILDCHAIN(Y) (mandatory) ROUTEDMSGS(ALL)
3790	3277CM See note “3” on page 225	2 (default value)	DEFSCREEN(24,80) PAGESIZE(24,80) AUTOPAGE(N) BRACKET(Y) (mandatory) BUILDCHAIN(Y) (mandatory) ROUTEDMSGS(ALL)
3790	3284CM See note “4” on page 225	1	DEFSCREEN(12,40) PAGESIZE(12,40) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3790	3284CM See note “4” on page 225	2 (default value)	DEFSCREEN(24,80) PAGESIZE(24,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3790	3286CM See note “4” on page 225	1	DEFSCREEN(12,40) PAGESIZE(12,40) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)
3790	3286CM See note “4” on page 225	2 (default value)	DEFSCREEN(24,80) PAGESIZE(24,80) AUTOPAGE(Y) BRACKET(Y) (mandatory) BUILDCHAIN(N) ROUTEDMSGS(ALL)

Table 5. Default values for TYPETERM attributes (continued)			
DEVICE	**SESSIONTYPE**	**TERMMODEL**	**Default values**

Note:

1. If you specify DEVICE(3277) or DEVICE(L3277), CICS replaces the value you specify with DEVICE(3270)
2. If you specify DEVICE(3284), DEVICE(3286), DEVICE(L3284) or DEVICE(L3286), CICS replaces the value you specify with DEVICE(3270P)
3. If you specify DEVICE(3790) and SESSIONTYPE(3277CM), CICS replaces the values you specify with DEVICE(LUTYPE2). There is no SESSIONTYPE value DEVICE(LUTYPE2).
4. If you specify DEVICE(3790) and SESSIONTYPE(3284CM) or SESSIONTYPE(3286CM), CICS replaces the values you specify with DEVICE(LUTYPE3). There is no SESSIONTYPE value DEVICE(LUTYPE3).

Devices supported

This topic lists the device names that you can use on the TYPETERM definition.

Table 6 on page 225 is a list of terminal types supported by RDO. To use this table, find the family number of your device in the leftmost column (headed **Terminal or System Type**). Then look across at the second column (headed **Units**) to see if your type of units is specifically mentioned. Next look at the third column (headed **Attachment**) to see what device type you should use on your TYPETERM definition.

The notes that follow the table provide further explanation where required. The following abbreviations are used:

local
channel or adapter attached

s/s
start/stop transmission

SDLC
synchronous data link control

sw
switched

BSC
binary synchronous

nonsw
nonswitched communications

Table 6. z/OS Communications Server terminals and subsystems supported by RDO

Terminal or System Type	Units	Attachment	Notes
3101		Supported as TWX 33/35	"9" on page 229
3230		Supported as INTLU (z/OS Communications Server)	
3270	3178, 3179, 3180, 3262, 3271, 3272, 3274, 3276, 3290	local, SDLC, BSC nonsw	"1" on page 228, "2" on page 228
	3275, 3277, 3278, 3279, 3284, 3286, 3287, 3288, 3289	BSC sw or nonsw	"2" on page 228
3270PC	3270PC, 3270PC/G, 3270PC/GX	Supported as 3270	

Table 6. z/OS Communications Server terminals and subsystems supported by RDO (continued)			
Terminal or System Type	**Units**	**Attachment**	**Notes**
3287	models 11, 12	SDLC supported as SCSPRT	“12” on page 229
3600	3601, 3602, 3690, 3604, 3610, 3612, 3618, 3614, 3624	SDLC, BSC nonsw	“3” on page 228,“4” on page 228,“13” on page 229
3630	3631, 3632, 3643, 3604	attached as 3600	“3” on page 228,“10” on page 229
3640	3641, 3644, 3646, 3647	SDLC attached as INTLU	“12” on page 229
	3642	SDLC attached as SCSPRT	“12” on page 229
	3643	SDLC supported as LUTYPE2	“12” on page 229
	3645	SDLC supported as SCSPRT	“12” on page 229
3650	3651, 3653, 3275, 3284	SDLC	“3” on page 228
3680	3684	Supported as 3790/3650	“3” on page 228
3730	3791	Supported as 3790	“3” on page 228
3767		SDLC s/s supported as 2740/2741	
3770	3771, 3773, 3774	SDLC	“3” on page 228,“5” on page 229
	3775, 3776, 3777	BSC supported as 2770	
3790	3791	SDLC or local	“3” on page 228,“6” on page 229
4300	4331, 4341, 4361, 4381	BSC or SDLC	“3” on page 228,“7” on page 229
4700	4701-1	Supported as 3600	“3” on page 228,“4” on page 228
5280		Supported as 3270 (z/OS Communications Server)	
5520		SDLC supported as 3790 full-function LU BSC supported as 2770 SDLC attached as APPC	“3” on page 228

Table 6. z/OS Communications Server terminals and subsystems supported by RDO (continued)			
Terminal or System Type	**Units**	**Attachment**	**Notes**
5550		Supported as 3270	
5937		SDLC/BSC attached as 3270	“2” on page 228
6670		SDLC BSC supported as 2770	
8100	8130/8140 processors with DPCX	Supported as 3790	“3” on page 228
	DPPX/BASE using Host Presentation Services or Host Transaction Facility	Attached as 3790	“3” on page 228
	DPPX/DSC or DPCX/DSC (including 8775 attach)	Supported as 3270	“3” on page 228,“11” on page 229
8775		SDLC supported as LUTYPE2	
8815		Supported as APPC	
Displaywriter		Supported as APPC SNA for EDDS Supported as 3270; attached as 2741 (s/s)or 3780 (BSC) SDLC attached as APPC	
Personal Computer		Supported as 3270 and as APPC	“13” on page 229
PS/2		Supported as 3270 and as APPC	“13” on page 229
Scanmaster		Supported as APPC	
Series/1		Attached as System/3;supported as 3650 Pipeline (z/OS Communications Server) or 3790 (full function LU)	“3” on page 228
System/32	5320	SDLC supported as 3770 BSC supported as 2770	“3” on page 228,“8” on page 229
System/34	5340	SDLC supported as 3770 BSC attached as System/3	“3” on page 228,“8” on page 229
System/36		Supported as System/34 SDLC attached as APPC	
System/38	5381	SDLC attached as 3770 SDLC attached as APPC BSC attached as System/3	“3” on page 228,“8” on page 229

Table 6. z/OS Communications Server terminals and subsystems supported by RDO (continued)			
Terminal or System Type	**Units**	**Attachment**	**Notes**
AS/400	5381	SDLC attached as 3770 SDLC attached as APPC BSC attached as System/3	"3" on page 228,"8" on page 229
System/370		SDLC attached as APPC	"3" on page 228
System/390®		SDLC attached as APPC	"3" on page 228
z Systems®		SDLC attached as APPC	"3" on page 228
TWX 33/35		z/OS Communications Server (via NTO) ss sw	"9" on page 229
WTTY		z/OS Communications Server (via NTO) ss nonsw	"9" on page 229

Note:

1. CICS supports the 3290 through both the terminal control and the BMS interfaces. The 3290 can be in one of three states: default, alternate, or partitioned. Up to 16 partitions can be defined. The 3290 also has the programmed symbols and extended highlighting features, as well as two kinds of data validation feature, mandatory fill and mandatory enter. A 3290 terminal can be configured as from one to five logical units. You define the size of each logical unit when setting up the 3290. You must ensure that the resource definitions of each logical unit match the setup size, to prevent unpredictable results. Up to four interactive screens in any configuration can be active at the same time, but only one interactive screen can be defined as having programmed symbols at any time; that is, all programmed symbol sets must be assigned to the interactive screen.

   To display long lines of data, such as the 132-character lines of CEMT output, specify a default screen width of 132 characters.

   If you intend to use the large buffer, you might have to specify a much larger value for IOAREALEN (see "TYPETERM attributes" on page 230). Whether you need to do this depends on whether operators are likely to modify, or enter, large quantities of data. If a terminal is used for inquiry only, or for limited data entry, IOAREALEN need not be large.
2. SDLC 3270s are supported only through z/OS Communications Server. Printers attached to local or SDLC 3274s and SDLC 3276s are supported through z/OS Communications Server either as LU Type 3 using the 3270 printer data stream or as LU Type 1 using the SCS data stream (which is a subset of that used for SDLC 3767, 3770, and 3790 printers). The 3288 is supported as a 3286 Model 2. CICS supports the 3270 copy feature (#1550).
3. Devices and features supported by a system or programmable controller are generally transparent to CICS. In some cases, CICS provides specific device support, in which case the units are listed.
4. SDLC is supported through z/OS Communications Server. The 3614 is supported both for loop attachment to the 3601 and SDLC attachment to the host via a 3704/3705 Communications Controller.

   The 3614 is supported by CICS as BSC only when loop-attached to the 3601/3602 Controllers. The 3624 is supported as a 3614.

   The 3690 is supported as a 3602.

5. CICS supports the Data Transfer Function of the SDLC Programmable Models of the 3770 Data Communication System. In using this function, you are responsible for allocating data sets and managing the program library.
6. CICS does not support the 3790/Data Entry Configuration using 3760s. The #9165 or #9169 configuration is required to support the CICS enhancement first made available in Version 1 Release 3.0.

   Printers on 3790 systems are supported with one of the following:

   - A function program that you provide
   - 3270 data stream compatibility with a 3270 printer data stream (LU3)
   - An SCS data stream supporting a subset of that for SDLC 3767 (LU1)

   When operating in 3270 mode, the 3288 Model 2 is supported as a 3286 Model 2.
7. The 4300 attachment by BSC requires a suitable telecommunications program (for example, the VSE/3270 Bisync Pass Through Program) in the system connected to CICS. CICS is **not** a suitable program for the remote CPU. Attachment by SDLC is supported by CICS intersystem communication.
8. The System/32 with its SNA/SDLC workstation system utility program, and the System/34 and System/38, are supported as compatible versions of an appropriately featured 3770 Communication System operating as a batch logical unit. The System/34 or System/38 user-written program is responsible for supporting the correct SNA sequences of the attached subsystem.
9. TWX and WTTY are supported through z/OS Communications Server via the Network Terminal Option licensed program (5735-XX7), with attachments as defined by NTO.

   WTTY is attached at 50 bps on common-carrier switched networks where the terminals supported are those interfacing through IBM World Trade Corporation Telegraph Terminal Control with Telegraph Line Adapter.

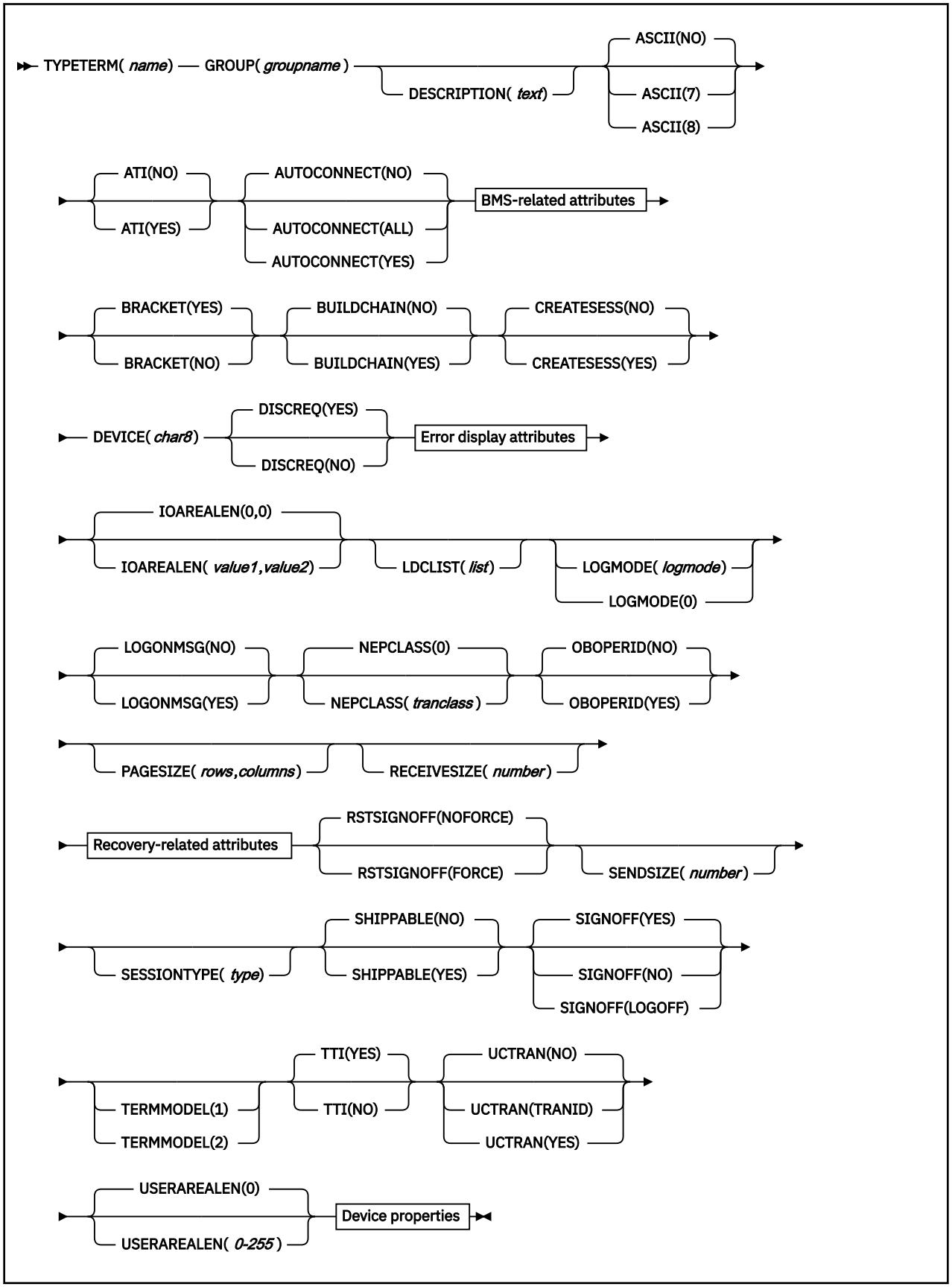
   The transmission code used is International Telegraph Alphabet No. 2 (CCITT No. 2). CICS does not support autocall or automatic host disconnect via WRITE break.
10. The 3643 is supported as a 3604.
11. 8775 support includes validation of mandatory fill and mandatory enter field attributes.
12. Attachment is via the Loop Adapter #4830, #4831, and Data Link Adapter #4840 of the 4331 processor.
13. 3270 support requires that the 3278/3279 Emulation Adapter be installed in the Personal Computer or PS/2.
14. The 3600 pipeline logical unit is designed to provide high throughput for particular types of transaction, such as credit card authorization or general inquiry applications. To achieve a high throughput of inquiry messages and their replies, the CICS pipeline session uses a restricted set of the communication protocols that are used with the 3601 logical unit.

    These restrictions result in a full duplex message flow whereby many inquiry messages are outstanding at any one time, and the replies may flow back in a different order from that of the original inquiries. The 4700/3600 application program controlling the inquiry terminals is responsible for maintaining the protocols as well as correlating replies with inquiries, and controlling message flow to the group of terminals associated with the pipeline logical unit.

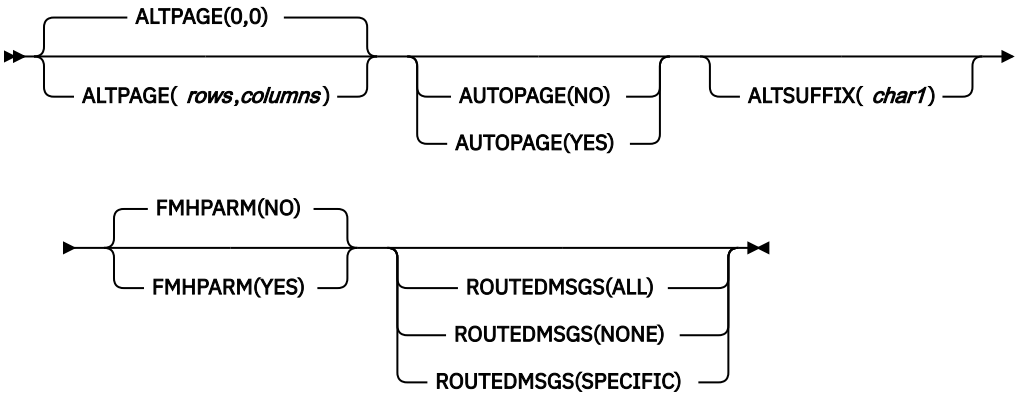
    CICS does not support automatic transaction initiation (ATI) on pipeline terminals.

TYPETERM attributes

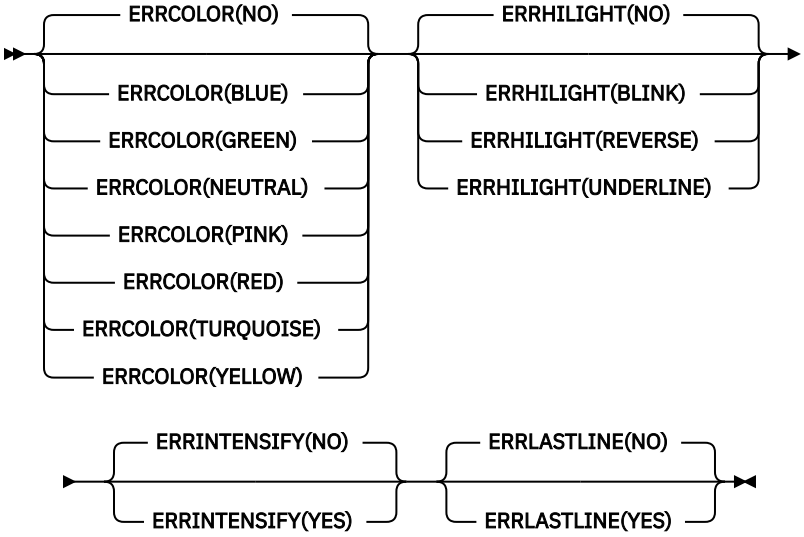
Describes the syntax and attributes of the TYPETERM resource.



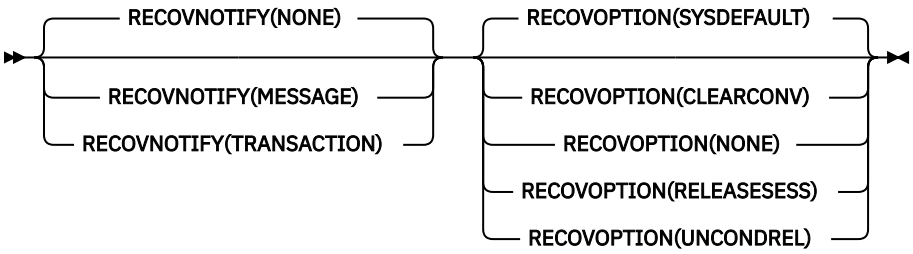
BMS-related attributes



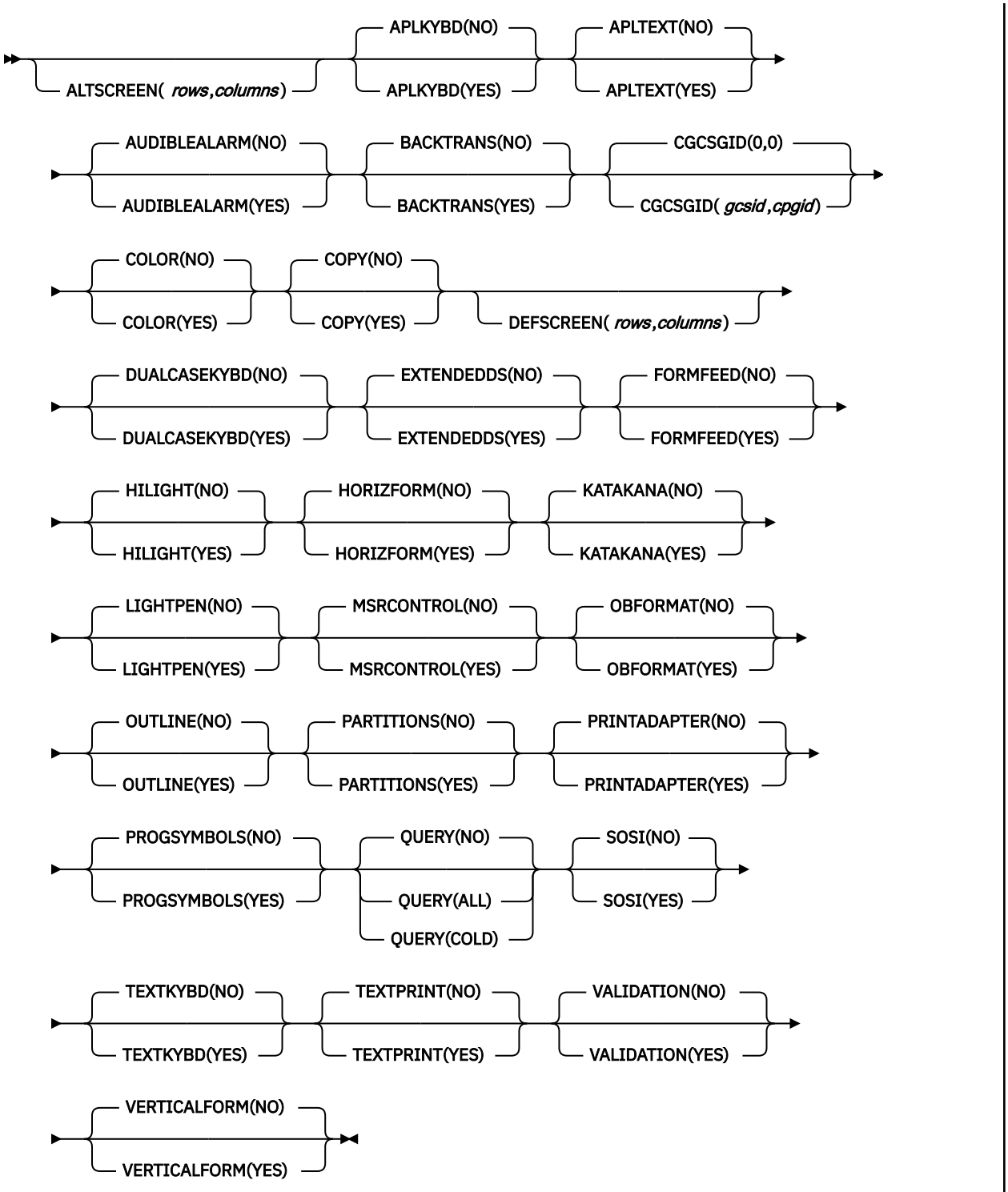
Error display attributes



Recovery-related attributes



Device properties



The typeterm definition attribute descriptions are:

ALTPAGE({0|*rows*},{0|*columns*})
Specifies the page size to be used by BMS for this terminal entry when ALTSCREEN has been selected as the screen size. The default is the PAGESIZE. The values for both *rows* and *columns* must be in the range 0 through 255. The product of *rows* and *columns* must not exceed 32767.

You will get unexpected results if the *columns* value of ALTPAGE is different from that of ALTSCREEN. The *rows* value of ALTPAGE can usefully be less than that of ALTSCREEN, perhaps to reserve the bottom line of the screen for error messages.

If you use the QUERY structured field, the alternate page size used is the size set up as the alternate screen size. For terminals that can be queried, you can set ALTPAGE to zero and have the ALTSCREEN value defined explicitly by the CINIT BIND. If ALTPAGE is not zero, it is possible to have different values for the ALTPAGE and the ALTSCREEN.

ALTSCREEN(*rows,columns*)

Specifies the 3270 screen size to be used for a transaction that has an alternate screen size specified in its profile definition. The values for both *rows* and *columns* must be in the range 0 through 255. The values that can be specified are:

Device	Alternate Screen Size
3276-1, 3278-1	(12,80)
3276-2, 3278-2	(24,80)
3276-3, 3278-3	(32,80)
3276-4, 3278-4	(43,80)
3278-5	(27,132)
3279-2A, 3279-2B	(24,80)
3279-3A, 3279-3B	(32,80)

No validity checking is performed on the screen size selected, and incorrect sizes may lead to unpredictable results.

For BSC devices, both the alternate and default screen sizes are determined by the device hardware. The alternate screen size is the maximum screen size. For the 3290 display, both the default and alternate screen sizes are determined by the customer setup procedure.

For SNA devices (LUTYPE2 and LUTYPE3), you can specify any value for both alternate and default screen sizes, up to the maximum physical screen size. In particular, both the alternate and default screen sizes can be the maximum screen size, or the default screen size can be the maximum screen size with no alternate screen size specified. The SNA bind is generated by CICS from this information. You do not need to provide logmode table entries, or to customize the device.

For non-SNA 3270 and LUTYPE2 devices, you can use the QUERY structured field to determine the alternate screen size that has been set up for the display. To use QUERY, leave the DEFSCREEN to default to (24,80) and leave ALTSCREEN unspecified. The alternate screen size is the size set up by the terminal user. Otherwise, QUERY(COLD) or QUERY(ALL) has no effect on the alternate screen size. Leaving ALTSCREEN unspecified without using QUERY under the conditions described results in an alternate screen size of (00,00).

If you use dual screen sizes, you can make a CICS transaction use the alternate screen size by coding SCRNSIZE(ALTERNATE) in its associated profile. If an application consists of several pseudo-conversationally linked transactions, specify SCRNSIZE(ALTERNATE) in the profile for each of these transactions if the application uses the alternate screen size.

For 3287 and 3289 printers, the value specified must equal the buffer size of the particular device. For non-SNA 3287 and 3289 printers, the sizes depend on the feature ordered, not on the model number. For SNA printers, there are no features, and any two sizes can be specified from the list of valid sizes. When printing to a printer whose associated TERMINAL definition has PRINTERCOPY(YES) specified, the ALTSCREEN value should match the screen size of the terminal whose screen is to be printed. If the values differ, unpredictable results may occur.

ALTSUFFIX({blank|*number*})

A 1-character numeric suffix that BMS is to append to map set names (specified in the SUFFIX operand of the DFHMSD TYPE={DSECT|MAP} macro).

blank

Leave this attribute blank if you do not want a suffixed map set.

number
BMS appends this suffix to map set names if the screen size being used is the same value as the alternate screen size; that is, if the transaction has an alternate screen size specified in the PROFILE definition, or if the default and alternate screen size are the same. In this case, BMS map selection routines attempt to load the map set with the suffix specified in the ALTSUFFIX operand.

If there is no such map set, BMS tries to load a map set suffixed with M or L and, if this load fails, BMS tries to load an unsuffixed map set version.

If the transaction uses default screen size, BMS first tries to load a map set suffixed with M or L and, if this load fails, BMS tries to load an unsuffixed map set version.

To use a suffixed map set, you must specify the BMS=(,,,DDS) system initialization parameter.

APLKYBD({NO|YES})
Specifies whether the 3270 device has the APL keyboard feature:

YES
The 3270 device has the APL keyboard feature.

NO
The 3270 device does not have the APL keyboard feature.

APLTEXT({NO|YES})
Specifies whether the 3270 device has the APL text feature:

YES
The 3270 device has the APL text feature.

NO
The 3270 device does not have the APL text feature.

Do not specify YES for a 3288 printer, with or without TEXTPRINT(YES). The APLTEXT feature is used in conjunction with the TEXTKYBD and APLKYBD operands.

You can use the QUERY structured field to determine whether the device is set up to use the APL text feature.

ASCII({NO|7|8})
Specifies whether the terminal has an ASCII feature.

NO
This terminal does not have an ASCII feature.

7
Specify this value to communicate with ASCII-7 terminals. Devices configured with the ASCII-7 feature must be LUTYPE2 or LUTYPE3 without extended 3270 features. Only the following devices are supported:

3274 Model 1C and 51C
3276 Model 12
3278
3287

Any terminal configured with the ASCII-7 option has all FM data outbound from CICS converted to ASCII-7, and all FM data inbound to CICS converted to EBCDIC. Only FM request data is translated. All other data in the RU such as LU status or sense data is assumed to be in EBCDIC on output. ASCII-7 does **not** support data streams that contain extended attributes, such as structured fields and function management headers.

The ASCII-7 support is available on 3274-1C as an option on the configuration of the standard microcode. The use of the ASCII-7 option is determined at session initiation by BIND parameters set by CICS as a result of the TCT definition described.

8
Specify this to communicate with ASCII-8 terminals. Devices configured with the ASCII-8 feature can be LUTYPE1, LUTYPE2, or LUTYPE3 with or without extended 3270 and SCS data stream features.

Any terminal configured with the ASCII-8 option has all FM data outbound from CICS converted to ASCII-8, and all FM data inbound to CICS converted to EBCDIC. All FM request data is translated. This includes the AID, cursor address, FM headers and structured fields. Any other form of the RU such as LU status or sense data is assumed to be in EBCDIC on input and is transmitted in EBCDIC on output.

This ASCII-8 support is intended only for devices that operate in EBCDIC but translate or retranslate the data stream to or from ASCII-8, as is done by this CICS support. This is because the data stream is treated as a character string, and any binary number fields are translated byte by byte as though they were graphic characters. Thus they may not represent their true value while in ASCII form.

The ASCII-8 support is available as a microcode RPQ on the 3274 and is mutually exclusive with the ASCII-7 option. The use of the ASCII-8 option is determined at session initiation by BIND parameters set by CICS as a result of the TCT definitions described.

ATI({NO|YES})
Specifies whether transactions can start at the terminal by automatic transaction initiation:

YES
Transactions can start at the terminal by automatic transaction initiation.

NO
Transactions cannot start at the terminal by automatic transaction initiation.

ATI(YES) allows transactions to be started at the terminal by transient data control or by an EXEC CICS START command issued by another transaction. If there is already a transaction at the terminal, the ATI transaction is held until it ends. If you specify ATI(YES), you must specify an IOAREALEN of at least one byte, except for DEVICE(APPC) when ATI and IOAREALEN have forced default values of YES and 0.

If ATI is specified as YES and CREATESESS is specified as YES, and if a transaction is initiated when the terminal is not ACQUIRED, it is automatically acquired.

See also the TTI attribute.

AUDIBLEALARM({NO|YES})
Specifies whether the audible alarm feature is installed for a 3270 display or for a 3270 printer attached to a 3651 controller:

YES
The audible alarm feature is installed.

NO
The audible alarm feature is not installed.

AUTOCONNECT({NO|YES|ALL})
Specifies whether autoconnect processing is to occur for the terminal. AUTOCONNECT(YES) or (ALL) specifies that the session with the terminal is to be established (that is, BIND is to be performed) during CICS initialization, or when communication with z/OS Communications Server is started using the **SET VTAM OPEN** command. If the connection cannot be made at this time because the terminal is unavailable, the link must be subsequently acquired using the CEMT SET TERMINAL(*termid*) INSERVICE ACQUIRED command, unless the terminal becomes available in the meantime and itself initiates communications.

Note: If you use the z/OS Communications Server LOGAPPL function, do not specify AUTOCONNECT(YES), because this can lead to race conditions causing errors or hung logical units.

NO
CICS does not attempt to bind sessions when the connection is established.

YES
CICS attempts to bind as a contention winner session, when the connection is established.

ALL
Not applicable.

For background information about AUTOCONNECT, see Defining intercommunication resources in Configuring.

AUTOPAGE({NO|YES})
Specifies whether BMS autopaging is to be used. Specify YES for printers and NO for display devices. The default depends on the value you specify for the DEVICE attribute. The default values are indicated in "Default values for TYPETERM attributes" on page 217

BACKTRANS({NO|YES})
Specifies whether the device has the background transparency feature:

NO
The device does not have the background transparency feature.

YES
The device does have the background transparency feature.

You can use the QUERY structured field to determine whether the device is set up to use the background transparency feature.

BRACKET({YES|NO})
Specifies whether bracket protocol is to be enforced for this logical unit. The default depends on the value you specify for the DEVICE attribute (see "Default values for TYPETERM attributes" on page 217).

YES
Bracket protocol is to be used. This option is required for the 3790 inquiry and full function logical units. BRACKET(YES) is forced for many DEVICE types

NO
Bracket protocol is not to be used. You must specify BRACKET(NO) for a 3614 logical unit and the 3650 Host Command Processor (HCP) session.

Bracket protocol is a feature of SNA; if you specify BRACKET(YES) for non-SNA devices, CICS will neither follow, nor enforce, strict bracket protocol.

BUILDCHAIN({NO|YES})
Specifies whether CICS is to perform chain assembly before passing the input data to the application program.

The default depends on the value you specify for the DEVICE attribute.

NO
Any terminal input/output area (TIOA) received by an application program from this logical unit contains one request unit (RU).

YES
Any TIOA received by an application program from this logical unit contains a complete chain.

CGCSGID({0,0|*value1,value2*})
The coded graphic character set global identifier (CGCSGID) enables application programs to determine the character set supported at the device.

You can get this information from a QUERY structured field for some devices. For others, you must supply this information here, so that application programs can retrieve it using the **EXEC CICS ASSIGN** command.

0,0
No CGCSGID is specified.

gcsid,cpgid
The CGCSGID consists of two 5-digit decimal numbers which can take values in the range 1 through 65535. *gcsid* is the graphic character set global identifier (GCSGID) and *cpgid* is a specification of the code points for the set, the code page global identifier (CPGID).

COLOR({NO|YES})
Specifies whether the 3270 device or the SCS printer has the extended color feature, which allows colors to be selected for each field or character:

NO
The device does not have the extended color feature.

YES
The device has the extended color feature.

You can use the QUERY structured field to determine whether the device is set up to use the color feature.

COPY({NO|YES})
Specifies whether the copy feature for a 3270 display or printer is included in the 3270 control unit:

NO
The copy feature is included.

YES
The copy feature is not included.

Leave it to default to COPY(NO) for 3270 compatibility mode logical units, because COPY(YES) is ignored.

See also the PRINTERCOPY and ALTPRINTCOPY attributes of the TERMINAL definition.

For further details about screen copying, see the CICS 3270 Data Stream Device Guide.

CREATESESS({NO|YES})
Specifies whether sessions are to be created.

NO
Specify this to prevent internally generated session requests from creating a session. During CICS execution, this can be changed only by a CEMT command.

CREATESESS(NO) prevents EXEC START requests and automatic transaction initiation (ATI) requests for this terminal causing a session to be created. This means that the requests are either queued or rejected when no session is currently established.

YES
Specify this for a status that allows internally generated session requests to create a session. During CICS execution, this status can be generated only by a CEMT command.

CREATESESS(YES) allows EXEC START requests and automatic transaction initiation (ATI) requests for this terminal to cause a session to be created automatically.

DEFSCREEN(*rows , columns*)
Specifies the 3270 screen size or 3270 printer page size to be used on this device when attached to a transaction or used by BMS for which the default screen size has been specified in the profile definition. The values for both *rows* and *columns* must be in the range 0 through 255. The default depends on the value you specify for the DEVICE attribute (see "Default values for TYPETERM attributes" on page 217). The values that can be specified for a BSC 3270 are:

Device	Screen size
3278-1	(12,40)
3278-2	(24,80)
3276-3, 3278-3	(24,80)
3276-4, 3278-4	(24,80)

Device	Screen size
3278-5	(24,80)
3279-2A, 3279-2B	(24,80)
3279-3A, 3279-3B	(24,80)

For BSC devices, both default and alternate screen sizes are determined by the terminal hardware. The default screen size is (24,80), except for the 3278-1 where it is (12,40).

For SNA devices (LUTYPE2 and LUTYPE3), both default and alternate screen sizes can be any value you choose, up to the maximum physical screen size (see ALTSCREEN). In particular, both default and alternate screen sizes can be the maximum screen size; or the default screen size can be the maximum screen size with no alternate screen size specified. The SNA bind is generated by CICS from this TCT information. You do not need to provide logmode table entries, or to customize the device.

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

DEVICE(*device*)
specifies the device type which this TYPETERM defines. This attribute is mandatory for all TYPETERM definitions.

If you type DEVICE(*xxxx*), where *xxxx* is a valid device type, on the command line, together with SESSIONTYPE and TERMMODEL if appropriate, other attributes are given appropriate default values. For further guidance, see "Default values for TYPETERM attributes" on page 217. Entering or overtyping the DEVICE, SESSIONTYPE, or TERMMODEL values on the overtype-to-modify panel does **not** provide these defaults.

The valid attributes and the defaults for each device type are listed in "Default values for TYPETERM attributes" on page 217. The recommended attributes for non-SNA z/OS Communications Server 3270 devices are 3270 and 3270P for displays and printers, respectively. The following attributes can also be specified and are retained for compatibility with previous releases:

- Displays: 3277 and L3277
- Printers: 3284 and L3284, 3286 and L3286

For SNA z/OS Communications Server 3270 devices, use the LUTYPE2 or LUTYPE3 attribute as appropriate. LUTYPE2 logical units are those defined by SNA, which accept a 3270-display data stream. LUTYPE3 logical units are those defined by SNA, which accept a data stream similar to that for a 3270 printer.

For a list of device types supported by CICS, see "DFHTCT: CICS terminals list" on page 412. See also "Default values for TYPETERM attributes" on page 217 for a list of valid device names and the default attributes that they generate.

DISCREQ=({YES|NO})
Specifies whether disconnect requests are to be honored.

YES
CICS is to honor a disconnect request for a z/OS Communications Server device, and issue a z/OS Communications Server CLSDST macroinstruction to terminate the z/OS Communications Server session with that logical unit.

In addition, CESF LOGOFF or GOODNIGHT from the terminal causes disconnection if you specify YES.

YES is essential if the TYPETERM definition is referenced by AUTINSTMODEL TERMINAL definitions, so that autoinstalled terminal entries can be deleted automatically.

YES is the default, but in some situations, if your resource definition does not specify this attribute, CICS enforces the NO value for the attribute if this value is required for compatibility with other options in your resource definition.

NO
CICS is not to honor a disconnect request for a z/OS Communications Server device.

DUALCASEKYBD({NO|YES})
Specifies whether a 3270 display has a typewriter keyboard or an operator console keyboard. Both uppercase and lowercase data can be transmitted with either of these keyboards

NO
The device does not have a dual-case keyboard.

YES
The device has a dual-case keyboard.

ERRCOLOR({NO|*color*})
Specifies whether the error message is to be displayed in color. Coding ERRCOLOR(*color*) implies ERRLASTLINE(YES).

The colors you can specify are:

BLUE
RED
PINK
GREEN
TURQUOISE
YELLOW
NEUTRAL

ERRHILIGHT({NO|BLINK|REVERSE|UNDERLINE})
Specifies the highlighting, if any, with which error messages are to be displayed.

ERRINTENSIFY({NO|YES})
Specifies whether the error message is to be displayed in an intensified field. Coding ERRINTENSIFY(YES) implies ERRLASTLINE(YES).

ERRLASTLINE({NO|YES})
Specifies where error messages are to be displayed.

NO
An error message is displayed at the current cursor position and without any additional attributes.

YES
An error message is displayed starting at the beginning of the line nearest the bottom of the screen so that the whole message fits on the screen.

Because all error messages occupy the same line, if the messages are received in quick succession, they overlay one another and earlier messages may disappear before they have been read.

EXTENDEDDS({NO|YES})
Specifies whether the 3270 device or the SCS printer supports extensions to the 3270 data stream:

NO
The device does not support 3270 data stream extensions.

YES
The device supports 3270 data stream extensions.

EXTENDEDDS(YES) is implied if you specify YES for any one of the COLOR, HILIGHT, PROGSYMBOLS, QUERY, or VALIDATION (3270 only) attributes.

If extended data stream (EXTENDEDDS) is set to YES, the device will support the write structured field COMMAND and Outbound Query structured field.

You can use the QUERY structured field to determine whether the device is set up to use the extended data stream. Using the QUERY structured field sets EXTENDEDDS to YES if query is valid.

FMHPARM({NO|YES})
Specifies whether BMS is to accept user-supplied parameters for inclusion in the function management header built by BMS:

NO
Do not accept user-supplied parameters for inclusion in the function management header built by BMS.

YES
Accept user-supplied parameters for inclusion in the function management header built by BMS.

Specify YES only if the DEVICE type is 3650.

FORMFEED({NO|YES})
Specifies whether or not the device has the forms feed feature, which means that BMS uses the form-feed character when formatting output documents:

NO
The device does not have the form feed feature.

YES
The device has the form feed feature.

If DEVICE(SCSPRINT) is specified, BMS inserts a form-feed character at the beginning of the data stream. This causes the device to skip to the top margin of a new page before starting to print.

The top margin is defined by a set vertical format (SVF) data stream, and may be a line number equal to or greater than one. If a SVF data stream has not been sent to the printer, the top margin is line one. The line counter in the device is set to 1 when the operator sets up the paper.

Note that the device may also perform an automatic form feed if you try to print beyond a bottom margin. The bottom margin is also determined by the SVF data stream and defaults to the maximum presentation line (MPL). The MPL is the last line on the page and its value represents the page or form length as a number of lines (that is, physical page size times the line density). Both the MPL and the line density can be determined by the SVF data stream. Otherwise the MPL (the number of lines) can be set up on the device by the operator.

If DEVICE(3270), DEVICE(3270P), DEVICE(LUTYPE2), or DEVICE(LUTYPE3) is specified, use FORMFEED(YES) in conjunction with the FORMFEED option in the BMS SEND commands. Using form feed on display devices provides for a skip to a new page when the screen data is copied to a printer.

The options discussed here for SCSPRINT operation do not apply when the devices are operating as 3270P or LUTYPE3 devices. In this case there is only the concept of a form length, and this can be set on the device only by the operator.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

HILIGHT({NO|YES})
Specifies whether the 3270 device or SCS printer has the extended highlight facility, which enables fields or characters to be displayed in reverse-video, underline mode, or blink (3270 only):

NO
The device does not have the extended highlight facility.

YES
The device has the extended highlight facility.

You can use the QUERY structured field to determine whether the device is set up to use the extended highlight facility.

HORIZFORM({NO|YES})
Specifies whether or not the device has the horizontal form feature, which means that BMS should use the horizontal tabbing when formatting output documents:

NO
The device does not have the horizontal form feature.

YES
The device has the horizontal form feature.

The devices that can use this feature are batch, batch data interchange, interactive, SCSPRT or LUTYPE4 logical units.

NO
The HTAB option in the BMS map definition is ignored.

YES
BMS uses horizontal tabbing when formatting output documents.

IOAREALEN({0|*value1*},{0|*value2*})
Specifies the length in bytes of a terminal input/output area to be passed to a transaction.

If you specify ATI(YES), you must specify an IOAREALEN of at least one byte.

value1
Value1 specifies the minimum size of a terminal input/output area to be passed to an application program when a **RECEIVE** command is issued.

value2
You can specify *value2* as greater than or equal to *value1*. In this case, when the size of an input message exceeds *value1*, CICS uses a terminal input/output area *value2* bytes long. If the input message size also exceeds *value2*, the node abnormal condition program sends an exception response to the terminal.

If *value2* is not specified, or is less than *value1*, it defaults to the value of *value1*.

The maximum value that you can specify for IOAREALEN is 32767 bytes.

KATAKANA({NO|YES})
Specifies whether Katakana support is required. Katakana terminals cannot display mixed case output; uppercase characters appear as uppercase English characters, but lowercase characters appear as Katakana characters. If you have any Katakana terminals connected to your CICS system, specify UPPER on the MSGCASE system initialization parameter.

NO
Katakana support is not required.

YES
Katakana support is required. All lowercase characters sent to the terminal from the following transactions are translated to uppercase:

CBRC CDBC CDBI CEBR CECI CEDA CEDF CEMT CEOT CESN CEST CMSG CRTE CSPG CWTO

Important: For emulated Katakana terminals, results depend on the code page that is in use. In some cases, lowercase English characters are not translated to uppercase.

LDCLIST(*list*)
Specifies the name of a logical device code (LDC) list. The name can be up to eight characters in length. The name follows assembler language rules. It must start with an alphabetic character.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

Define the LDCLIST and its contents using macroinstruction(s).

A local LDC list is defined by:

```
    listname  DFHTCT TYPE=LDCLIST,
    LDC(aa=nnn,bb=nnn,....)
```

An extended local LDC list is defined by:

```
listname  DFHTCT TYPE=LDC,LOCAL=INITIAL
          DFHTCT TYPE=LDC=(aa=nnn)....
          DFHTCT TYPE=LDC=(bb=nnn)....
          DFHTCT TYPE=LDC,LOCAL=FINAL
```

You specify this *listname* as the value for the LDCLIST attribute on the TYPETERM definition.

This attribute applies only to 3600, 3770 batch, 3770, and 3790 batch data interchange, and LUTYPE4 logical units. The list specifies which LDCs are valid for this logical unit and, optionally, which device characteristics are valid for each LDC. CICS uses the first LDC generated in this list when choosing a default LDC for a logical unit. For further guidance, see "DFHTCT logical device codes: z/OS Communications Server non-3270" on page 398.

LIGHTPEN({NO|YES})
Specifies whether a 3270 display has the selector pen feature:

NO
The 3270 display does not have the selector pen feature.

YES
The 3270 display has the selector pen feature.

LOGMODE({blank|0|*logmode*})
Specifies how CICS is to build the BIND to be sent to the logical unit.

blank
A defined terminal definition uses the BIND image generated by the CICS definitions for this device by means of this TYPETERM definition and its associated terminal definitions. An autoinstalled terminal uses the fields specified in the incoming CINIT.

name
This is the LOGMODE name from a z/OS Communications Server logon mode table that has been set up for use by this logical unit. The name may be up to eight characters in length and must follow assembler language rules. The name must start with an alphabetic character.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

This allows you to override the BIND image provided by CICS for the logical unit. For further information, see the appropriate CICS subsystem guide.

You cannot code a LOGMODE name when the terminal is a cross-domain resource.

The TCTTE is updated to reflect the logmode bind image fields. These include SEND and RECEIVE sizes and default and alternate screen sizes. If the logmode indicates that the terminal is not queriable, the extended data stream fields are all set to zero.

0 (zero)
This causes CICS to use some of the information from the BIND image contained in the CINIT coming from the logical unit. The BIND image in the CINIT was created by z/OS Communications Server based on the LOGMODE entry defined for the logical unit requesting to log on to CICS. The node initialization block (NIB) is built with LOGMODE=0 and BNDAREA=0. When the TYPETERM's SENDSIZE and RECEIVESIZE have been specified as zero, CICS replaces them with the values from the LOGMODE's RUSIZES.

The TCTTE is updated to reflect the incoming CINIT fields. These include SEND and RECEIVE sizes and default and alternate screen sizes. If the logmode indicates that the terminal is not queriable, the extended data stream fields are all set to 0. Use LOGMODE(0) only in exceptional circumstances. Although the LU is bound with the z/OS Communications Server definition, CICS keeps the main session characteristics from the CICS definition. For example, if a printer is defined to z/OS Communications Server as LUTYPE1 but to CICS as an LUTYPE3 with LOGMODE(0), CICS accepts the bind but sends LUTYPE3 control characters to the printer, giving rise to incorrect results. This restriction does not apply to pipeline terminals.

Note:

1. You should only need to use this value for the logmode attribute in exceptional circumstances.
2. For a logical unit in a cross-domain environment, specify LOGMODE(0) and provide the logical unit mode information in the DLOGMOD and MODETAB operands of the z/OS Communications Server(LU) statement. In a cross-domain environment, LOGMODE with a name causes a z/OS Communications Server error.

LOGMODECOM
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS.

LOGONMSG({NO|YES})
Specifies whether the 'good morning' transaction, specified in the **GMTRAN** system initialization parameter, will be:

- Automatically initiated when the logical unit is first logged on to CICS through z/OS Communications Server
- Initiated after the terminal user's TIMEOUT period has expired under certain conditions.

If you have specified ERRLASTLINE(YES), the messages written by the transaction do not overwrite the error message line.

NO
CICS does not run the 'good morning' transaction.

Note: If you are using a non-SNA terminal such as a Telnet 3270, LOGONMSG(NO) does not automatically release the keyboard lock. You need to press the Reset key to release the keyboard lock.

YES
CICS runs the 'good morning' transaction when the OPNDST exit is successfully completed and a session is established. The transaction is initiated by automatic task initiation (ATI) and competes with other ATI transactions for use of the terminal. Specify ATI(YES) for this TYPETERM.

Note: If you are using a non-SNA terminal such as a Telnet 3270, LOGONMSG(YES) also automatically releases the keyboard lock.

MSRCONTROL({NO|YES})
Specifies whether the terminal, an 8775 or 3643, has a magnetic slot reader. This option is not valid for SCS printers.

You can use the QUERY structured field to determine whether the device is set up to use a magnetic slot reader.

NEPCLASS({0|*tranclass*})
Specifies the node error program transaction class.

0
This results in a link to the default node error program module.

tranclass
The transaction class for the (nondefault) node error program module. *tranclass* can be in the range 1 through 255. For programming information about the node error program, see Writing a node error program.

OBFORMAT({NO|YES})
Specifies whether outboard formatting is used. If the devices for which you are defining this TYPETERM use BMS outboard formatting, specify OBFORMAT(YES). OBFORMAT(YES) can be specified for two device types only:

- 3650, SESSIONTYPE(3270)
- LUTYPE2, for an 8100 Information System using the DPPX operating system with DPPX/DPS Version 2 for presentation services

Use the QUERY structured field to determine whether the device is set up to use outboard formatting.

OBOPERID({NO|YES})
Specifies whether CICS uses the outboard operator identifiers to support the BMS routing facilities required for this terminal. This option applies only to the 3790 and 3770 batch data interchange logical units.

NO
CICS does not use the outboard operator identifiers.

YES
CICS uses the outboard operator identifiers.

OUTLINE({NO|YES})
Specifies whether the device supports field outlining:

NO
The device does not support field outlining.

YES
The device supports field outlining.

Use the QUERY structured field to determine whether the device is set up to use field outlining.

PAGESIZE(*rows*,*columns*)
Specifies the default page size for this printer. The values for both *rows* and *columns* must be in the range 0 through 255. The default page size is used by BMS when the default screen size has been selected in the DEFSCREEN attribute.

rows
Indicates the number of rows in the page. The PAGESIZE *rows* value can usefully be less than the DEFSCREEN *rows* value, perhaps to reserve the bottom line of a screen for error messages (see the ERRLASTLINE attribute), if the same BMS map is being used for both printing and display.

columns
Indicates the number of characters in each line. Unexpected results occur if the *columns* value specified in PAGESIZE differs from the *columns* value specified in DEFSCREEN.

The product of *rows* and *columns* must not exceed 32767.

The default value depends on the value you specify for the DEVICE attribute. See “Default values for TYPETERM attributes” on page 217 for details.

BMS uses the page size values when preparing output data streams. The specified number of characters in each line of the page should not exceed the physical line width of the terminal. In the case of printers that automatically perform a new-line function on reaching the end of the carriage (for example, 3270 printers), the line width specified here should be less than the physical line width.

This ensures that the formatting of the output data is governed entirely by the new-line (NL) characters supplied by BMS or by you, not by new-line functions performed by the device itself,

which would produce additional lines of output, resulting in a physical page depth greater than that specified here.

For 3270-type printers, the hardware limits the amount of data that BMS may transmit. If the map or application program request specifies L40, L64, or L80, or does not specify NLEOM on the SEND MAP command, the product of *lines* and *columns* specified in PAGESIZE must not be greater than the buffer size.

If the BMS request specifies NLEOM, the page length may be any number of lines, but the product of *lines* and *columns* specified in the DEFSCREEN or ALTSCREEN attributes must not exceed the buffer size of the device. In other words, the number of characters that BMS transmits must not exceed the physical buffer size of the printer.

Note: BMS divides a large page into smaller segments for transmission. PAGESIZE should therefore correspond to the required **logical** page size (the product of *lines* and *columns*), and the DEFSCREEN value should correspond to the actual buffer size.

For a z/OS Communications Server 3600, the PAGESIZE specified is used if a BMS page build operation is attempted without specifying a logical device code (LDC). A default device type of 3604 is assumed.

For 3770, LUTYPE4, or 3790 batch data interchange logical units, the PAGESIZE specified is used if a BMS page build operation is requested without specifying a logical device code (LDC). The default device type is the console printer.

Take care when routing a message to a list of terminals. If the PAGESIZE you have defined (or allowed to default) is too small to accommodate the message, the transaction abends.

For cumulative text processing, the maximum allowed buffer size is 32767. If this is exceeded, BMS internally forces a reduced page length to ensure that the PAGESIZE stays within the limit.

PARTITIONS({NO|YES})
Specifies whether a device is to use partitions. This option is not valid for SCS printers.

NO
The device is not to use partitions.

YES
The device is to use partitions.

You can use the QUERY structured field to determine whether the device is set up to use partitions.

PRINTADAPTER({NO|YES})
For the 3275: specifies whether the printer adapter feature and corresponding 3284 Printer Model 3 are present on the 3275 Display Station. This feature makes the 3284 eligible for print requests through the PA key from the host 3275.

NO
The printer adapter feature and corresponding 3284 Printer Model 3 are not available.

YES
The printer adapter feature and corresponding 3284 Printer Model 3 are available.

For LUTYPE2 logical units: specifies whether, for print requests initiated by the PRINT key or by an ISSUE PRINT command, printer allocation is handled by the 3790, or by the 3274 or 3276, according to the printer authorization matrix for both z/OS Communications Server and non-z/OS Communications Server attachments.

NO
Print requests are not handled according to the printer authorization matrix for both z/OS Communications Server and non-z/OS Communications Server attachments.

YES
Print requests are handled according to the printer authorization matrix for both z/OS Communications Server and non-z/OS Communications Server attachments.

Further, 3270 printers attached to the same 3790 are available for print requests sent to the 3270-display logical unit by a terminal control print request or initiated by the operator. If PRINTADAPTER is NO, printer allocation is determined by the PRINTER and ALTPRINTER attributes of the TERMINAL definition.

If output is created on the screen by BMS requests with the PRINT option, by BMS requests with the NLEOM option, or by the CMSG command, the contents of the screen are automatically copied to a 3270 printer, whether or not the CICS-defined PRINT key (usually a PA key) was pressed.

PROGSYMBOLS({NO|YES})
Specifies whether the programmed symbol (PS) facility can be used on this 3270 device or SCS printer. The facility enables up to six 191-character sets, with customer-defined and program-loaded fonts and codes, to be stored and accessed.

NO
Programmed symbol (PS) facility cannot be used.

YES
Programmed symbol (PS) facility can be used.

You can use the QUERY structured field to determine whether the device is set up to use programmed symbols.

QUERY({NO|COLD|ALL})
Specifies whether CICS should use the QUERY structured field to determine the characteristics of the device.

NO
CICS does not use the QUERY function.

COLD
CICS uses the QUERY function to determine the characteristics of the device only when the device is first connected after an initial or a cold start of CICS. The device characteristics are stored in the CICS global catalog for use on subsequent warm and emergency starts.

ALL
CICS uses the QUERY function to determine the characteristics of the device each time the device is connected.

RECEIVESIZE(*number*)
For a **defined nonautoinstalled terminal**, specify the maximum size of a request unit that can satisfy a z/OS Communications Server RECEIVE request. The RECEIVESIZE value is transmitted to the connected logical unit, and must be in the range 0 through 30720. It may be rounded down by CICS, because it must be transmitted in an architected form.

The effect of RECEIVESIZE depends on whether a RECEIVE RUSIZE is present in the z/OS Communications Server LOGMODE table. Table 7 on page 246 shows the RECEIVE RUSIZE used to bind a session for each possible combination of TYPETERM and LOGMODE values.

Table 7. RECEIVE RUSIZE for defined (nonautoinstalled) terminals

RECEIVE RUSIZE (VTAM)	TYPETERM RECEIVESIZE	RUSIZE used in bind
0	0	0
0	specified	TYPETERM RECEIVESIZE size
specified	0	This combination is invalid and results in a bind failure with message DFHZC2403
specified	specified	TYPETERM RECEIVESIZE size

Table 7. RECEIVE RUSIZE for defined (nonautoinstalled) terminals (continued)		
RECEIVE RUSIZE (VTAM)	**TYPETERM RECEIVESIZE**	**RUSIZE used in bind**
Note: The exception to this table is LOGMODE(0). If you specify this in your TYPETERM definition, VTAM values are used, irrespective of what else is specified.		

APPC terminal For an APPC (LUTYPE6.2) single session terminal, 256 would be a suitable value.

Autoinstalled terminal For an autoinstalled terminal, a nonzero value for RECEIVESIZE specifies either the maximum or actual RECEIVE RUSIZE value used in binding a session for a logical unit defined with this TYPETERM.

The effect of RECEIVESIZE depends on whether a RECEIVE RUSIZE is present in the z/OS Communications Server LOGMODE table. Table 8 on page 247 shows the RECEIVE RUSIZE used to bind a session for each possible combination of TYPETERM and LOGMODE values.

Table 8. RECEIVE RUSIZE for autoinstalled terminals		
RECEIVE RUSIZE (VTAM)	**TYPETERM RECEIVESIZE**	**RUSIZE used in bind**
0	0 and BUILDCHAIN(YES)	256
0	0 and BUILDCHAIN(NO)	0
0	specified	TYPETERM RECEIVESIZE size
specified	0	VTAM RECEIVE RUSIZE size
specified less than or equal to TYPETERM RECEIVESIZE	specified	VTAM RECEIVE RUSIZE size
specified greater than TYPETERM RECEIVESIZE	specified	this combination is invalid and results in message DFHZC5963

RECOVNOTIFY({NONE|MESSAGE|TRANSACTION})

In a CICS region running with persistent session support, this option specifies how a terminal end user is notified that the terminal session has been recovered; in a CICS region running with XRF support, it specifies how the terminal user is notified that an XRF takeover has occurred.

This option is not applicable to APPC sessions.

NONE
There is no notification that a terminal session has been recovered, or that an XRF takeover has occurred.

MESSAGE
A message is displayed on the screen to say that the system has recovered. The message is specified in two BMS maps (DFHXRC1 and DFHXRC2) for XRF and in one BMS map (DFHXRC3) for z/OS Communications Server persistent sessions. These maps are in map set DFHXMSG. If reduced takeover time is important, use MESSAGE rather than TRANSACTION.

The terminal must be defined with the ATI(YES) option, and must be capable of displaying a BMS map.

TRANSACTION
A transaction is initiated at the terminal. The name of the transaction is specified by the **RMTRAN** system initialization parameter. For z/OS Communications Server persistent sessions, only the first transaction named in the **RMTRAN** system initialization parameter is used.

Tip: The default transaction for RMTRAN is the one specified in the **GMTRAN** system initialization parameter: the good-morning transaction.

For the TRANSACTION option, the terminal must be defined with the ATI(YES) option. If reduced takeover time is important, use MESSAGE rather than TRANSACTION.

RECOVOPTION({SYSDEFAULT|CLEARCONV|RELEASESESS| UNCONDREL|NONE})
This option applies to the recovery of sessions in a CICS region running with z/OS Communications Server persistent sessions, or with XRF.

In a CICS region running with persistent session support, this option specifies how you want CICS to recover the session, and return the terminal to service on system restart within the persistent session delay interval.

For all recovery options other than NONE, if the action taken is a z/OS Communications Server UNBIND, the UNBIND is followed by a z/OS Communications Server SIMLOGON.

SYSDEFAULT is the default, but in some situations, if your resource definition does not specify this attribute, CICS enforces the NONE value for the attribute if this value is required for compatibility with other options in your resource definition.

SYSDEFAULT
In a CICS region running with persistent sessions support, this specifies that CICS is to select the optimum procedure to recover a session on system restart within the persistent session delay interval, depending on the session activity and on the characteristics of the terminal.

Although sessions are recovered, any transactions in-flight at the time of the failure are abended and not recovered. Transactions are also abended if the recovered session is being used by another CICS region over an APPC connection.

CICS recovers the session with the least possible impact, in one of the following ways:

- If the terminal was not executing a transaction at the time of the CICS failure, no recovery action is required, and CICS takes the appropriate recovery notification action as defined by the RECOVNOTIFY attribute.
- If the terminal was busy (that is, executing a transaction) when CICS failed, CICS first tries to recover the session by sending a z/OS Communications Server end-bracket indicator. If the end-bracket does not recover the session (for example, CICS may be in RECEIVE mode), CICS issues a CLEAR command. If the terminal does not support the CLEAR command, the recovery action taken is a z/OS Communications Server UNBIND followed by a SIMLOGON.

  See Application design considerations for more information about persistent sessions.

CLEARCONV
Prevents CICS from sending an end-bracket indicator to close an in-bracket session. Instead CICS sends a CLEAR request, to reset the conversation states. If the session does not support the CLEAR request, CICS sends an UNBIND request. The CLEAR or UNBIND is sent only if the session was busy at the time of system restart (in the case of persistent sessions) or takeover (in the case of XRF).

RELEASESESS
Requires CICS to send an UNBIND request to release the active session. The UNBIND is sent only if the session was busy at the time of system restart (in the case of persistent sessions), or takeover (in the case of XRF). Following the UNBIND, the session is queued for SIMLOGON. If the session is not busy, the requested recovery notification is carried out.

UNCONDREL
Requires CICS to send an UNBIND request to release the active session. The UNBIND is sent whether or not the session was busy at the time of system restart (in the case of persistent sessions support) or the takeover (in the case of XRF). Following the UNBIND, the session is queued for SIMLOGON.

NONE
In a CICS region running with persistent sessions support, this specifies that the terminal session is not to be recovered at system restart within the persistent session delay interval: in effect, the terminal has no persistent sessions support. LU6.2 sessions are unbound, but the latest negotiated CNOS value is returned to the CICS system after the restart. After system restart, the

terminal is reconnected automatically if you specify AUTOCONNECT(YES), subject to the operation of the **AIRDELAY** system initialization parameter (AIRDELAY=0 overrides AUTOCONNECT(YES), and the terminal is not reconnected).

Use RECOVOPTION(NONE) if this terminal or autoinstall model is to be used with persistent sessions (**PSDINT** = nnn in the SIT) but the terminal may be the subject of an **EXEC CICS ISSUE PASS LUNAME() LOGONLOGMODE**.

RELREQ=({NO|YES})
Specifies whether CICS is to release the logical unit upon request by another z/OS Communications Server application program.

NO
CICS is not to release the logical unit.

YES
CICS is to release the logical unit, if the logical unit is not currently part of a transaction.

ROUTEDMSGS({ALL|NONE|SPECIFIC})
Specifies which messages are to be routed to this terminal by an **EXEC CICS ROUTE** command. The default depends on the value you specify for the DEVICE attribute. See "Default values for TYPETERM attributes" on page 217 for details.

ALL
BMS routes to this terminal messages that are destined for **all** terminals as well as those specifically destined for **this** terminal.

NONE
BMS does not route any messages to this terminal, whether they are destined for all terminals or for this terminal specifically.

SPECIFIC
BMS routes messages to this terminal when they are destined specifically for this terminal, but not when they are destined for **all** terminals.

RSTSIGNOFF({NOFORCE|FORCE})
Specifies whether the terminal user should be signed off in the event of a persistent sessions restart or an XRF takeover.

FORCE
The terminal will be signed off after a persistent sessions restart or XRF takeover.

NOFORCE
The terminal will remain signed on after a persistent sessions restart or XRF takeover, provided that the **RSTSIGNOFF** system initialization parameter and the XRFSOFF entry in the CICS segment of the RACF user profile are both set to NOFORCE.

SENDSIZE(*number*)
Defined terminal (nonautoinstalled): For a nonautoinstalled terminal, this is the maximum size in bytes of a request unit that can satisfy a z/OS Communications Server **VTAM SEND** request. The SENDSIZE value is transmitted to the connected logical unit, and must be in the range 0 through 30720. It may be rounded down by CICS, because it must be transmitted in an architected form.

The effect of SENDSIZE depends on whether a SEND RUSIZE is present in the z/OS Communications Server LOGMODE table. Table 9 on page 249 shows the SEND RUSIZE used to bind a session for each possible combination of TYPETERM and LOGMODE values.

Table 9. SEND RUSIZE for defined (nonautoinstalled) terminals

SEND RUSIZE (VTAM)	TYPETERM SENDSIZE	RUSIZE used in bind
0	0	0
0	specified	TYPETERM SENDSIZE size
specified	0	0

Table 9. SEND RUSIZE for defined (nonautoinstalled) terminals (continued)		
SEND RUSIZE (VTAM)	**TYPETERM SENDSIZE**	**RUSIZE used in bind**
specified	specified	TYPETERM SENDSIZE size
Note: The exception to this table is LOGMODE(0). If you specify this in your TYPETERM definition, VTAM values are used, irrespective of what else is specified. **Note:** VTAM is now z/OS Communications Server.		

APPC terminal: For an APPC (LUTYPE6.2) single session terminal, 256 is a suitable value.

Autoinstalled terminal: For an autoinstalled terminal, a nonzero value for SENDSIZE specifies either the maximum or actual SEND RUSIZE value used in binding a session for a logical unit defined with this TYPETERM.

The effect of SENDSIZE depends on whether a SEND RUSIZE is present in the z/OS Communications Server LOGMODE table. Table 10 on page 250 shows the SEND RUSIZE used to bind a session for each possible combination of TYPETERM and LOGMODE values.

Table 10. SEND RUSIZE for autoinstalled terminals		
SEND RUSIZE (VTAM)	**TYPETERM SENDSIZE**	**RUSIZE used in bind**
0	0	0
0	specified	TYPETERM SENDSIZE size
specified	0	VTAM SEND RUSIZE size
specified less than or equal to TYPETERM SENDSIZE	specified	VTAM SEND RUSIZE size
specified greater than TYPETERM SENDSIZE	specified	This combination is invalid and results in message DFHZC5963

SESSIONTYPE(*type*)
Specifies the type of session that can be used for a z/OS Communications Server SNA logical unit. For details, see "Default values for TYPETERM attributes" on page 217.

SHIPPABLE({NO|YES})
Specifies whether the definition is allowed to be sent to a remote system if this device tries to initiate a remote transaction.

NO
This definition cannot be shipped to a remote system.

YES
This definition can be shipped to a remote system.

This function may be used for any terminal, whether autoinstalled or with its own TERMINAL definition. The shipping does not work unless the terminal has a definition installed, by one of these methods, in the local system.

Using SHIPPABLE(YES) means that you do not need to ensure that a definition of the terminal exists on the remote system for a locally defined terminal to initiate a transaction in that system. This can be useful when the remote system cannot share the CSD file with the local system.

A definition for the terminal must already be installed in (or already shipped to) the remote system.

For guidance on deciding whether to use SHIPPABLE(YES), see "Terminals for transaction routing" on page 176.

SIGNOFF({YES|NO|LOGOFF})
Specifies the actions taken when GNTRAN (CESF or user-defined transaction) is attached and attempts to sign off the terminal. If you are using RACF, specify the TIMEOUT limit in the RACF segment.

YES
When the specified time has elapsed after the last input from the operator, the terminal is automatically signed off from CICS.

NO
The terminal is not timed out.

LOGOFF
When the specified time has elapsed after the last input from the operator, the terminal is automatically signed off from CICS and then logged off from z/OS Communications Server. LOGOFF is useful for an autoinstall model, because virtual storage is not wasted on entries for terminals that have been timed out.

If GNTRAN fails to attach because of unprocessed data in the terminal buffer (resulting in a BID failure), the terminal will be signed off and logged off. GNTRAN will not run and will have no effect.

Note: You cannot change the value of this attribute when DEVICE(APPC) is specified. The default value in that case is SIGNOFF(NO).

SOSI({NO|YES})
Specifies whether the device supports mixed EBCDIC and double-byte character set (DBCS) fields.

NO
The device does not support mixed EBCDIC and double-byte character set (DBCS) fields.

YES
The device supports mixed EBCDIC and double-byte character set (DBCS) fields.

You can use the QUERY structured field to determine whether the device is set up to use mixed EBCDIC and DBCS fields.

TERMMODEL({1|2})
Specifies the model number of the terminal. If the device is a component of the 3270 Information Display System, specify the model number of the terminal:

1
Specify 1 for the 3270 Model 1 displays and printers (for example, 3277 Model 1) with a default screen or buffer size of 12x40 (480 bytes/characters). TERMMODEL(1) is the default for 3270 Model 1 printers and displays.

Specify 1 for the 3275 Display Station Model 11. The CICS support obtained is identical to that obtained by coding TERMMODEL(1) for 3275 Display Station Model 1.

2
Specify 2 for the 3270 displays and printers (for example, 3278 Model 4) with a default screen or buffer size of 24x80 (1920 bytes/characters). TERMMODEL(2) is the default for the 3286 printer in 3270 compatibility mode.

Specify 2 for the 3275 Display Station Model 12. The CICS support obtained is identical to that obtained by coding TERMMODEL(2) for 3275 Display Station Model 2.

TEXTKYBD({NO|YES})
Specifies whether the 3270 device has the text-keyboard feature.

NO
The 3270 device does not have the text-keyboard feature.

YES
The 3270 device has the text-keyboard feature.

TEXTPRINT({NO|YES})
Specifies whether the 3288 printer has the text-print feature.

NO
The 3288 printer doe not have the text-print feature.

YES
The 3288 printer has the text-print feature.

TTI({YES|NO})
Specifies whether transactions can be initiated at the terminal by a user.

YES
Transactions can be initiated at the terminal by a user. If you also specify ATI(YES), transactions can also be initiated automatically. In this case, the automatic transaction initiation, either by transient data control or interval control, sets a condition in an appropriate terminal control table terminal entry. If both ATI and TTI are specified as YES, and if there is no transaction at the terminal, terminal control initiates the user-defined task. This task is expected to send messages to the terminal.

For a terminal used in the processing of transactions such as inquiries or order entries, specify TTI(YES) and ATI(NO). This also applies to a display station or hard-copy terminal to which no messages are sent without a terminal request and through which transactions are entered. Note that this is the only specification allowed for 3790 inquiry logical units.

NO
Transactions cannot be initiated at the terminal by a user. If you specify NO, specify ATI(YES) to allow transactions to be initiated automatically. An example of this type of terminal is one that is located in a remote location, such as a warehouse, and is unattended but may receive messages.

TYPETERM(*name*)
Specifies the name of this extension of a TERMINAL definition. The name can be up to eight characters in length.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

This name is referred to in all the TERMINAL definitions using this TYPETERM. Note that this TYPETERM definition must be installed before or at the same time as the TERMINAL definitions that reference it.

UCTRAN({NO|YES|TRANID})
Specifies whether the input data stream from a terminal is to be translated to uppercase. The input data stream may include a transaction identifier as well as program data. CICS supports transaction identifier definition in mixed case, and the UCTRAN attribute can be used to ensure that the correct transaction is located. Uppercase translation is done for both 3270 and non-3270 data streams.

NO
No uppercase translation is performed.

YES
All the data input from the terminal, both the transaction identifier if present and the program data, is translated to uppercase before any processing.

TRANID
When the input data stream includes a transaction identifier, CICS translates it to uppercase before attempting to locate its definition. However, all the input data, both the transaction identifier and the program data, is passed to the program without any translation.

Therefore both the YES and the TRANID options allow transaction identifiers to be defined in uppercase and to be entered from the terminal in either uppercase or lowercase, but the TRANID option causes the transaction identifier and program data to be passed to the program without any translation.

You can also request translation to uppercase at the transaction level on PROFILE resources (see “PROFILE attributes” on page 114), but be aware that a TYPETERM UCTRAN(YES) definition overrides a PROFILE UCTRAN(NO) definition. So, if you specify TYPETERM UCTRAN(YES), a PROFILE UCTRAN(NO) has no effect. Translation can be overridden by the application program for all RECEIVE requests except the first, by using the ASIS option.

Table 11 on page 253 shows which portion of the terminal input is translated (transaction id and/or data) according to the setting of the UCTRAN on the PROFILE and TYPETERM resource definitions.

Table 11. The effect of UCTRAN attributes on tranid and data translation

UCTRAN in PROFILE	UCTRAN in TYPETERM	TRANID translated?	Data Translated?
YES	YES	Yes	Yes
YES	NO	No	Yes
YES	TRANID	Yes	Yes
NO	YES	Yes	Yes
NO	NO	No	No
NO	TRANID	Yes	No

Some national-language characters are not automatically translated when UCTRAN(YES) or UCTRAN(TRANID) is specified. If that is the case, you can use one of the methods described in Uppercase translation.

USERAREALEN({0|*number*})
Specifies the length in bytes (0 to 255) of the user area for this terminal. It should be made as small as possible. The terminal user area is initialized to zeros at system initialization.

The terminal user area might be located in 31-bit storage or in 24-bit storage, depending on the value of the **TCTUALOC** system initialization parameter. The default is that the terminal user area can be in either 31-bit storage or 24-bit storage, but CICS uses 31-bit storage wherever possible. If you require the terminal user area to be in 24-bit storage, because you have application programs that are not capable of 31-bit addressing, specify the system initialization parameter **TCTUALOC=BELOW** for the CICS region.

VALIDATION({NO|YES})
For the 8775, specifies whether the 8775 device has the extended validation feature, which allows fields to be defined as TRIGGER, MANDATORY FILL, or MANDATORY ENTER.

For the 3290, specifies whether the 3290 device has the validation feature, which allows fields to be defined as MANDATORY FILL or MANDATORY ENTER.

This option is not valid for SCS printers. If VALIDATION(YES) is specified for an SCS printer, an error message is raised and the option is ignored.

You can use the QUERY structured field to determine whether the device is set up to use the validation feature.

VERTICALFORM({NO|YES})
Specifies whether the device has the vertical form feature. The devices that can use this feature are batch, batch data interchange, interactive, SCSPRT or LUTYPE4 logical units.

NO
The device does not have the vertical form feature.

YES
The device has the vertical form feature.

XRFSIGNOFF
This attribute is obsolete, but is supported to provide compatibility with earlier releases of CICS.

Typeterm resource definitions

Typeterm definitions are partial terminal definitions that describe a set of common attributes for a group of terminals.

URIMAP resources

URIMAP definitions are resources that match the URIs of HTTP, Atom feed, or web service requests, and provide information on how to process the requests.

Requests from a web client, to CICS as an HTTP server

URIMAP definitions for requests for CICS as an HTTP server have a USAGE attribute of SERVER. These URIMAP definitions match the URLs of HTTP requests that CICS expects to receive from a web client, and they define how CICS should provide a response to each request. You can use a URIMAP definition to tell CICS:

- Provide a static response to the HTTP request, by using a document template or z/OS UNIX System Services file.
- Provide an application-generated response to the HTTP request, by using an application program.

The web client requests might qualify for being processed by directly attached user transactions, and bypassing the web attach task. For more information, see HTTP requests are processed by directly attached user transactions.

For information about configuring CICS as an HTTP server, see Enabling CICS web support for CICS as a HTTP server. For information about the CICS resources that are required for the HTTP server, see Defining resources for CICS as a HTTP server.

Requests to a server, from CICS as an HTTP client

URIMAP definitions for requests from CICS as an HTTP client have a USAGE attribute of CLIENT. These URIMAP definitions specify URLs that are used when a user application, acting as a web client, makes a request through CICS web support to an HTTP server. Setting up a URIMAP definition for this purpose means that you can avoid identifying the URL in your application program. You can also choose to pool opened connections for reuse by the same application or another application. For more information, see Creating a URIMAP resource for CICS as a HTTP client.

Requests from a SOAP client to CICS as a web service provider

URIMAP definitions for inbound web service requests have a USAGE attribute of PIPELINE. These URIMAP definitions associate a URI for an inbound web service request (that is, a request by which a client starts a web service in CICS) with the PIPELINE and WEBSERVICE resources that specify the processing to be performed.

For service providers deployed by using the CICS web services assistant, URIMAP resources are created automatically when the pickup directory is scanned. This scan occurs when the PIPELINE resource is installed or as a result of a **PERFORM PIPELINE SCAN** command. The URIMAP resource that provides CICS with the information to associate the WEBSERVICE resource with a specific URI is a required resource. The attributes for this resource are specified by a web service binding file in the pickup directory. The URIMAP resource that provides CICS with the information to associate the WSDL archive file or WSDL document with a specific URI is an optional resource and is created if either a WSDL file or WSDL archive file is present in the pickup directory.

Web services overview has further information about web services in CICS.

The SOAP client requests might qualify for being processed by directly attached user transactions, and bypassing the web attach task. For more information, see HTTP requests are processed by directly attached user transactions.

Requests to a web service provider from CICS as a web service requester

URIMAP definition for outbound web service requests by using the **INVOKE WEBSERVICE** command are optional and have a USAGE attribute of CLIENT. For example, you can use the URIMAP definition to specify the cipher suites or certificate label to use when you are establishing a socket connection that uses Transport Layer Security (TLS). You can also choose to pool opened connections to be reused for further **INVOKE WEBSERVICE** commands by the application, rather than opening a new connection each time. URIMAP definitions give administrators control of the connection to the remote web service: they can change its URI as needed, or enable and disable the URIMAP to allow or disallow the connection.

For service requesters, CICS does not create any URIMAP resources automatically when the PIPELINE resource is installed or as a result of a **PERFORM PIPELINE SCAN** command.

Requests from a web client to Java web application

URIMAP definitions for Java web applications have a USAGE attribute of JVMSERVER. These URIMAP definitions match an inbound request from a web client to a servlet or JSP that is running in a JVM server. URIMAP matching for web applications occur on the initial HTTP request and not on any forwarded requests. The URIMAP can specify a CICS transaction to run the request in the JVM server and optionally a user ID.

For more information, see Deploying a Java EE application in a CICS bundle to a Liberty JVM server.

Atom feed requests

URIMAP definitions for Atom feed requests have a USAGE attribute of ATOM. These URIMAP definitions match an inbound request from a web client for an Atom document. They reference an ATOMSERVICE resource definition that specifies the Atom document to be returned.

For more information, see Setting up CICS definitions for an Atom feed.

The Atom feed requests might qualify for being processed by directly attached user transactions, and bypassing the web attach task. For more information, see HTTP requests are processed by directly attached user transactions.

Business event emissions from the HTTP EP adapter

URIMAP definitions for CICS event processing have a USAGE attribute of CLIENT. These URIMAP definitions specify a URL that the HTTP EP adapter uses to emit events to an HTTP 1.1 compliant server that uses HTTP POST. You can choose to pool opened connections to be reused for further event emissions. Event processing overview has further information about CICS event processing.

For CICS as an HTTP server, URIMAP definitions incorporate most of the functions that were previously provided by the analyzer program that is associated with the TCPIPSERVICE definition. You can still use an analyzer program in the processing path if you need to do so.

URIMAP search order

If multiple URIMAPs are available, any URIMAPs with the same URI and a TCPIPSERVICE are searched first. URIMAPs containing wildcards are searched for a matching URI first and TCPIPSERVICE last. This search order ensures that the most specific URIMAP is used.

URIMAP resources in CICS bundles

You can use a CICS bundle to create, edit, and install a URIMAP resource definition. If you create a URIMAP resource in this way, you must use the CICS bundle to manage the lifecycle of that resource, and you cannot manage the resource independently. You can inquire on a URIMAP resource that is dynamically generated by a CICS bundle, by using the **EXEC CICS INQUIRE URIMAP** or **CEMT INQUIRE URIMAP** command. However, you cannot issue SET or DISCARD commands against a URIMAP resource that is dynamically generated by a CICS bundle. You must issue the commands against the BUNDLE

resource, and CICS applies them to the URIMAP resource. For more information about defining resources in CICS bundles, see Defining CICS bundles.

URIMAP resources as application entry points

The following usage scenarios are supported when you are using the URIMAP resource as an application entry point:

URIMAP resource with USAGE(JVMSERVER) using Liberty
If the URIMAP resource located by the Liberty JVM server is defined with an application entry point, the application context data is set for the JVM server request processor task, or its alias as specified by the transaction attribute.

URIMAP resource with USAGE(SERVER), or USAGE(PIPELINE) that does not use WebSphere MQ
If the URIMAP resource located by CWXN is defined with an application entry point, the application context data is set for the CWXN task. Then, because CWXN is a user task, the application context data can be automatically propagated to the alias transactions started by CWXN; for example, CPIH for USAGE(PIPELINE), CWBA for USAGE(SERVER), or the transaction that is specified by the **TRANSACTION** field.

The following usage scenarios are not supported when you are using the URIMAP resource as an application entry point:

URIMAP resource with USAGE(PIPELINE) using WebSphere MQ
This scenario is not supported.

URIMAP resource with USAGE(CLIENT), or USAGE(ATOM)
These scenarios are not supported. A bundle with URIMAP and USAGE(CLIENT) or a bundle with URIMAP and USAGE(ATOM) have the status DISABLED if the URIMAP is defined as an application entry point. If the bundle is installed and you attempt to enable the bundle, the ENTRYPOINT bundle part within the bundle has the status DISABLED and as a result the bundle has the status DISABLED.

Use the INQUIRE URIMAP and CEMT INQUIRE URIMAP commands to return the details of the application for which the URIMAP resource is defined as an entry point.

For more information about application entry points, see Application entry points.

URIMAP: related resources

The attributes and values you specify for a URIMAP resource must be consistent with those specified on other resources.

However, CICS does not check consistency of all related resources when the URIMAP definition is installed, and therefore does not report most inconsistencies at install time. The exception is the check described in “Installing URIMAP resource definitions” on page 270.

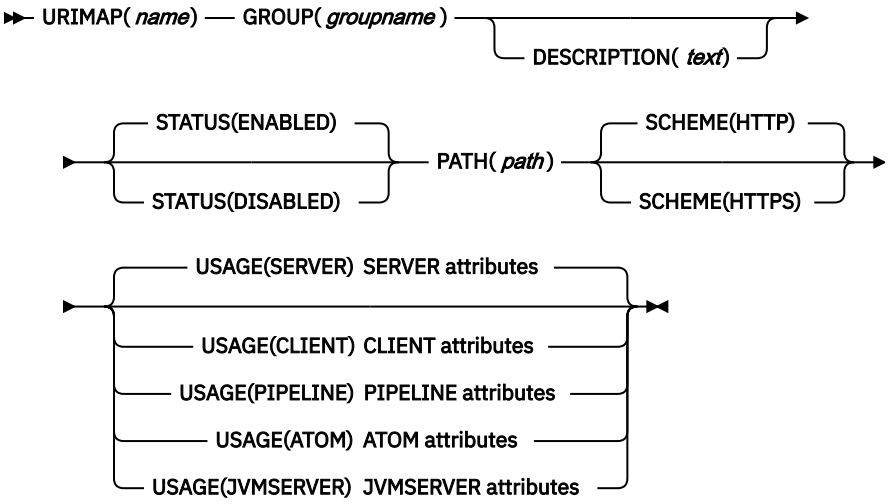
- In URIMAP definitions for CICS as an HTTP server and web services, the TCPIPSERVICE attribute specifies the name of a TCPIPSERVICE resource definition that defines an inbound port to which this URIMAP definition relates. The attribute is optional, and if it is not specified, the URIMAP definition applies to any inbound request on all TCPIPSERVICE definitions. If the TCPIPSERVICE attribute is specified:
  - The selected TCPIPSERVICE resource definition must specify PROTOCOL(HTTP).
  - If SSL(YES) or SSL(CLIENTAUTH) is specified in the TCPIPSERVICE resource definition, the SCHEME attribute of the URIMAP definition must be HTTPS. When a URIMAP definition with HTTPS as the scheme matches a request that a Web client is making, CICS checks that the inbound port used by the request is using SSL. If SSL is not specified for the port, the request is rejected with a 403 (Forbidden) status code.

  You must install the selected TCPIPSERVICE definition before the URIMAP definition can be used.
- In URIMAP definitions for CICS as an HTTP server where a static response is to be provided, the TEMPLATENAME attribute specifies the 1-48 character name of a CICS document template that will form the static response. The document template must be defined using a DOCTEMPLATE resource

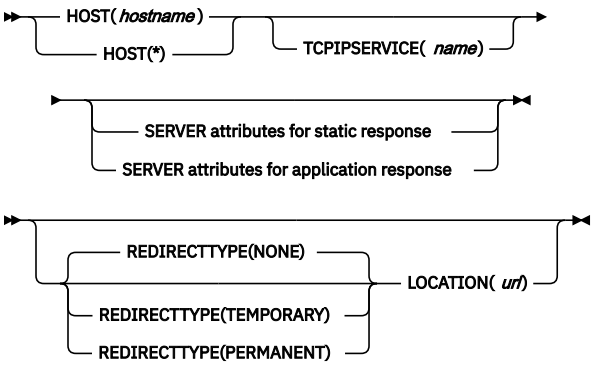
definition, and the TEMPLATENAME attribute of that definition specifies the name that is used in the URIMAP definition. The DOCTEMPLATE resource definition must be available before the URIMAP definition can be used.

URIMAP attributes

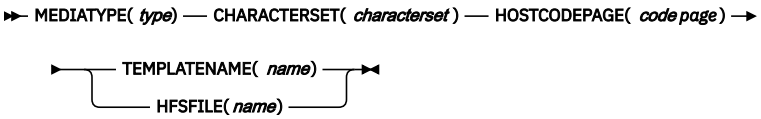
Describes the syntax and attributes of the URIMAP resource.



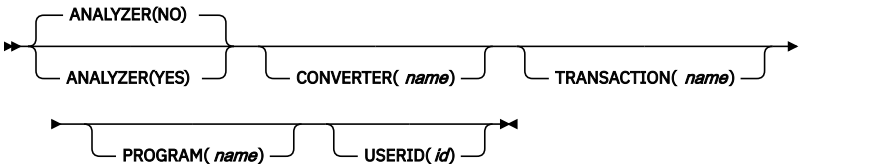
SERVER attributes



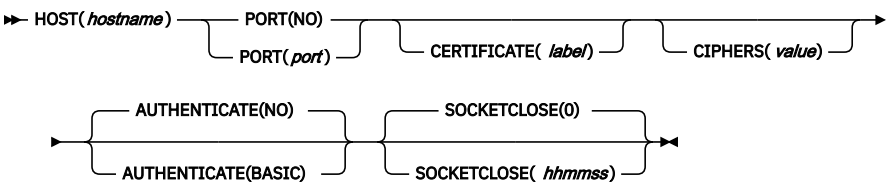
SERVER attributes for static response



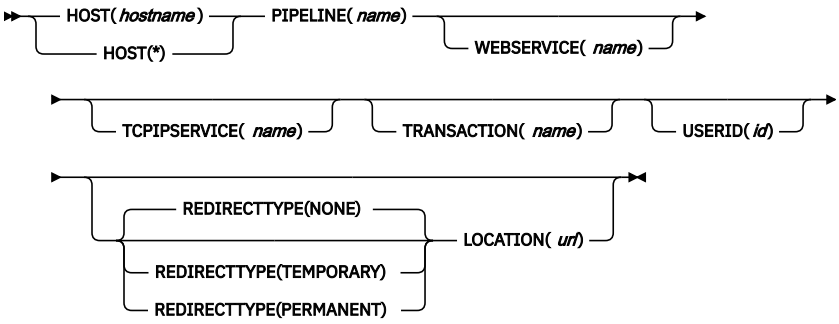
SERVER attributes for application response



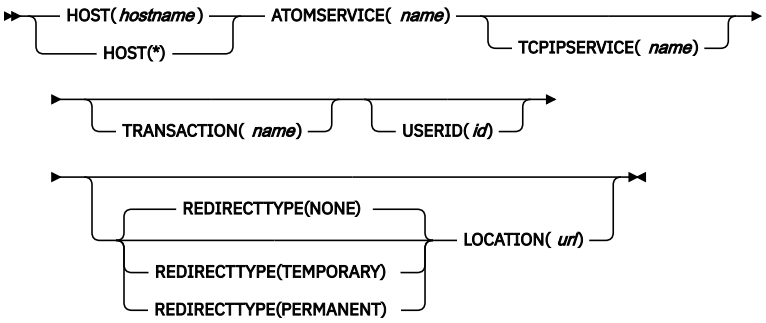
CLIENT attributes



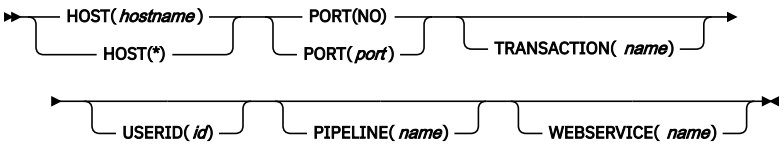
PIPELINE attributes



ATOM attributes



JVMSERVER attributes



ANALYZER({NO|YES})

This attribute is for USAGE(SERVER), where an application-generated response is to be provided. For other usage types, the attribute is forced to NO.

Setting ANALYZER to NO is required for HTTP requests to be processed by directly attaching the user transaction. For more information, see HTTP requests are processed by directly attached user transactions.

ANALYZER specifies whether an analyzer program is to be used in processing the HTTP request. The analyzer must be associated with the TCPIPSERVICE definition or definitions to which this URIMAP definition relates. (An analyzer program must be in the local CICS region.) YES runs the analyzer. NO means that the analyzer is not used.

The default analyzer DFHWBAAX and sample analyzer DFHWBADX do not analyze a request when a matching URIMAP definition is found for the request, even if the URIMAP specifies ANALYZER(YES).

If an analyzer program is used, you can still specify the CONVERTER, TRANSACTION, USERID, and PROGRAM attributes. The values that you specify for these attributes are used as input to the analyzer program, but they can be overridden by it. Alternatively, you can leave these attributes blank and let the analyzer program specify them.

ATOMSERVICE(*name*)

This attribute is for USAGE(ATOM).

When a client makes a request to CICS for an Atom feed by using the URI specified by this URIMAP definition, ATOMSERVICE specifies the 1- to 8-character name of the ATOMSERVICE resource definition for the Atom feed. The ATOMSERVICE resource definition defines an Atom service, feed, collection, or category document, and identifies the Atom configuration file, CICS resource or application program, and XML binding that are used to supply the data for the feed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

AUTHENTICATE({NO|BASIC})
This attribute is for USAGE(CLIENT).

AUTHENTICATE specifies whether to send HTTP basic authentication information to the HTTP server. AUTHENTICATE(BASIC) requires a user ID and password to be provided by the XWBAUTH global user exit, or as values on the API commands such as **WEB SEND** or **WEB CONVERSE**. The XWBAUTH global user exit is not called if USERNAME and PASSWORD are specified on the API command. If you specify an authentication value on the API command, this value is used instead of the AUTHENTICATE value that is specified in the URIMAP resource.

CERTIFICATE(*label*)
This attribute is for USAGE(CLIENT).

CERTIFICATE specifies the label of the X.509 certificate that is to be used as the SSL client certificate during the SSL handshake. Certificate labels can be up to 32 characters long. This attribute is used only when the URI specified by the URIMAP definition is to be used for an HTTPS request that is made by CICS as a client. It is up to the server to request an SSL client certificate, and, if it does, CICS supplies the certificate label that is specified in the URIMAP definition. If this attribute is omitted, the default certificate that is defined in the key ring for the CICS region user ID is used. The certificate must be stored in a key ring in the external security manager database. The certificate that is specified, or the default certificate, must have a private key available to it or the URIMAP cannot install. If you do not want to use a certificate, leave this field blank and ensure that the key ring used by the CICS region does not have a default certificate. For more information, see Building a key ring manually.

CHARACTERSET(*characterset*)
This attribute is for USAGE(SERVER), where a static response is to be provided.

CHARACTERSET specifies the 1- to 40-character name of the character set into which CICS converts the entity body of the response that is sent to the web client. CICS does not support all of the character sets that are named by IANA. HTML coded character sets lists the IANA character sets that are supported by CICS. The value of this attribute is included in the Content-Type header of the response.

You must specify CHARACTERSET if a static response is being provided and the MEDIATYPE attribute specifies a text type.

CIPHERS(*value*)
This attribute is for USAGE(CLIENT).

The CIPHERS attribute can be specified in either of two ways:

- **Recommended:** The name of the SSL cipher suite specification file, which is a z/OS UNIX file in the `security/ciphers` subdirectory of the directory that is specified by the **USSCONFIG** system initialization parameter. For example, if **USSCONFIG** is set to `/var/cicsts/dfhconfig` and **CIPHERS** is set to `strongciphers.xml`, the fully qualified file name is `/var/cicsts/dfhconfig/security/ciphers/strongciphers.xml`. For more information, see Creating an SSL cipher suite specification file.
- A string of up to 56 hexadecimal digits that is interpreted as a list of up to 28 2-digit cipher suite codes.

When you use the CEDA transaction to define the resource, CICS automatically initializes the attribute with a default list of acceptable codes. For CICS to initialize the attribute, the KEYRING system initialization parameter must be specified in the CICS region where you are running CEDA. If KEYRING is not set, CICS does not initialize the attribute. The default list of 2-digit ciphers is 3538392F3233.

You can reorder the cipher codes or remove them from the initial list. However, you cannot add cipher codes that are not in the default list for the specified encryption level. To reset the value to the default list of codes, delete all of the cipher suite codes. The field is automatically repopulated with the default list.

When you use DFHCSDUP to define the resource, the CIPHERS attribute is not populated with a default value. If the resource is used to establish a secure connection and the CIPHERS attribute is empty, CICS uses the default list of 2-digit ciphers. An INQUIRE on the resource that is created in this manner, returns an empty string for the CIPHERS value.

Any unsupported ciphers are removed at run time. A list of the removed ciphers is reported in messages DFHSO0145 and DFHSO0146.

For more information, see Cipher suites and cipher suite specification files.

CONVERTER(*name*)
This attribute is for USAGE(SERVER), where an application-generated response is to be provided.

CONVERTER specifies the 1- to 8-character name of a converter program that converts the request and response. Typically, a converter program transforms the HTTP request into a COMMAREA that can be used by an application program and transforms the output into an HTTP response. The converter program can be any converter program that is available in the local CICS region.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

Unlike the relationship between the analyzer program and the TCPIPSERVICE definition, the converter program and the TCPIPSERVICE definition have no association.

If the ANALYZER attribute is specified as YES, the CONVERTER attribute is used as input to the analyzer program, but it can be overridden by the analyzer program. If a converter program is used, you can still specify the PROGRAM attribute of the URIMAP definition, but the values that you specify for this attribute can be overridden by the converter program. Alternatively, you can leave this attribute blank and let the converter program specify it.

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

HFSFILE(*name*)
This attribute is for USAGE(SERVER), where a static response is to be provided.

HFSFILE specifies the fully qualified (absolute) or relative name of a z/OS UNIX System Services zFS file that forms the body of the static response that is sent to the HTTP request from the web client. Up to 255 characters can be used.

You can specify the name as an absolute path, including all directories and beginning with a slash; for example, `/u/facts/images/bluefish.jpg`. Alternatively, you can specify it as a relative path for example, `facts/images/bluefish.jpg`.

- For a static response file for a URIMAP resource that is defined by online resource definition, the file path is either fully qualified, if prefixed with a forward slash `/`, or is relative to the HOME directory of the CICS region user ID.
- For a static response file for a URIMAP resource that is defined in a CICS bundle, the file path is relative to the root directory of the CICS bundle. The zFS file must be packaged in the CICS bundle with the URIMAP resource. The MEDIATYPE and HOSTCODEPAGE attributes must be specified to inform CICS of the file type and the code page with which the file is encoded. When the file is defined in a CICS bundle and exported to zFS by using CICS Explorer, the code page with which the file is encoded is the same as the one used by CICS Explorer. For more information, see Referencing zFS artifacts in a bundle.

The value specified must be a valid name for a UNIX file:

- It must not contain embedded space characters.
- It must not contain consecutive instances of the / character.
- It is case-sensitive.

Acceptable characters:

```
A-Z a-z 0-9 . / _ # @ -
```

If TEMPLATENAME or HFSFILE is specified, ANALYZER must be set to NO. Other attributes that relate only to application-generated responses (TRANSACTION, CONVERTER, and PROGRAM) must remain blank.

If you want to use path matching, include an asterisk as a wildcard character at the end of the path for the zFS file and also at the end of the path that is specified by the PATH attribute. CICS takes the portion of the path of each HTTP request that is covered by the wildcard character and substitutes it as the last part of the file path.

For example, you can create a URIMAP definition with the PATH attribute specified as:

```
findout/pictures/*
```

and the HFSFILE attribute that is specified as:

```
/u/facts/images/*
```

The URIMAP definition is used to process an incoming HTTP request

```
http://www.example.com/findout/pictures/bluefish.jpg
```

CICS appends `bluefish.jpg` to the zFS file path specified in the URIMAP definition in place of the asterisk, so that the zFS file

```
/u/facts/images/bluefish.jpg
```

is used as the static response.

You cannot use an asterisk alone in the HFSFILE specification. At least one level of the directory structure must be specified.

If you are using IRIs (Internationalized Resource Identifiers) containing Unicode characters, the Unicode characters must be escaped to their percent-encoded representations in the zFS file name

and in the corresponding path. Tools are available on the web to help you. Search the web for "Unicode percent escaped".

A query string cannot be substituted into the zFS file contents, although you can define the zFS file as a CICS document template and specify it using the TEMPLATENAME attribute instead of the HFSFILE attribute.

HOST(*hostname*|*)

This attribute is for all USAGE options.

HOST specifies the host name of the URI to which the URIMAP definition applies, or its IPv4 or IPv6 address. The name can be up to 116 characters long. The components of a URL explains each of the components and how they are delimited.

The HOST attribute must be present. The HOST attribute must contain only alphanumeric characters, hyphens `-`, colons `:`, or periods `.`, although you cannot use colons when you are specifying a character host name instead of an IP address. CICS validates the host name at define time. A host name can be entered in any case, but if a character host name is specified instead of an IP address, the host name is converted to lowercase in the URIMAP definition.

When you specify USAGE(CLIENT) and when Transport Layer Security (TLS) is used in the connection to an HTTP server, CICS passes the host name in the Server Name Indication (SNI) extension during the TLS handshake. This allows CICS to use a TLS communication to a virtual host where the server supports multiple virtual hosts using a single IP address. See also CICS web support fundamentals: Virtual hosting.

When you specify USAGE(SERVER), USAGE(PIPELINE), USAGE(ATOM), or USAGE(JVMSERVER), you can use a single asterisk in the HOST attribute, making the URIMAP definition match any host name. You cannot use an asterisk as a wildcard in the HOST attribute with any other characters. Do not set a port number in this attribute for these usage types.

URIMAP resources support Internationalized Resource Identifiers (IRIs), which can contain Unicode characters. If you specify a host name that contains Unicode characters, you must convert the host name to Punycode format, which is described by RFC 3492. CICS does not provide a tool to carry out this conversion, but free applications are available on the Internet to support the conversion of Unicode to Punycode. If you use an asterisk as the host name, you do not need to use Punycode. For more information about IRIs, see Internationalized Resource Identifiers (IRIs).

You can specify IPv4 and IPv6 addresses in a number of acceptable formats. For more information about address formats, see IP addresses.

If CICS is an HTTP client and you specify USAGE(CLIENT), you can specify a port number in the request to the server:

- Use the PORT attribute to specify the port number. PORT replaces the use of the HOST attribute for specifying a port number.
- For compatibility purposes in existing programs that use native IPv4 addresses and host names, you can use the HOST attribute when you specify the port number. Native IPv4 addresses and host names are the only formats in which you can specify the port number, together with a preceding colon; for example, `1.2.3.4:80` or `hostname.com:443`.
- If you specify an IPv6 address (or a host name that resolves to an IPv6 address), ensure that you are operating in a dual-mode (IPv4 and IPv6) environment and that the client or server that you are communicating with is also operating in a dual-mode (IPv4 and IPv6) environment. For more information about IPv6, see Understanding IPv6 and CICS.
- For native IPv6 addresses, you must use the PORT attribute to specify the port number. IPv6 addresses require square brackets to separate the address from the port number, and, because square brackets are not fixed values in all EBCDIC character sets, square brackets are not supported in the HOST attribute.
- Specify the port number only if it is different from the default for the scheme; 80 for HTTP without SSL, or 443 for HTTPS and HTTP with SSL.

- If you specify a port number in the HOST attribute and a different port number in the PORT attribute, an error is returned. If you do not specify a port number in either the HOST or the PORT attribute, the default port number for the scheme is used.

HOSTCODEPAGE(*code page*)
This attribute is for USAGE(SERVER), where a static response is to be provided.

HOSTCODEPAGE specifies the 1- to 10-character name of the IBM code page (EBCDIC) in which the text document that forms the static response is encoded. CICS needs this information to convert the code pages for the entity body of the static response.

The standard CICS form of a host code page name consists of the code page number (or more generally CCSID) written by using 3 - 5 decimal digits as necessary then padded with trailing spaces. For code page 37, which is fewer than 3 digits, the standard form is 037. CICS accepts any decimal number, padded with trailing spaces, in the range 1 - 65535 as a code page name, even if it is not in the standard form.

You must specify HOSTCODEPAGE if a static response is being provided and the MEDIATYPE attribute specifies a text type.

LOCATION(*url*)
This attribute is for USAGE(SERVER), USAGE(PIPELINE), and USAGE(ATOM).

LOCATION specifies a URL of up to 255 characters to which the client request is redirected. The URL must be complete, including scheme, host, path components, and appropriate delimiters. CICS does not check that the URL is valid, so you must ensure that the destination exists and that the URL is specified correctly.

The description for the PATH attribute lists the characters that must be excluded from a URL. These characters must not be used in the LOCATION attribute. The exception is the # character, which can be used in the LOCATION attribute as a separator before a fragment identifier that follows the URL.

The REDIRECTTYPE attribute is used to specify the type of redirection. If temporary or permanent redirection is specified, the URL in the LOCATION attribute is used for redirection. If no redirection is specified, the URL in the LOCATION attribute is ignored. You can use the SET URIMAP command to change the REDIRECTTYPE attribute and the LOCATION attribute.

MEDIATYPE(*type*)
This attribute is for USAGE(SERVER), where a static response is to be provided.

MEDIATYPE specifies the media type (data content) of the static response that CICS provides to the HTTP request; for example, `image/jpg`, `text/html` or `text/xml`. Up to 56 characters can be used. The media type must contain exactly one forward slash (/). The media type can be entered in uppercase or lowercase, but it is converted to lowercase in the URIMAP definition.

The name for each formally recognized type of data content is defined by IANA. A list is available at http://www.iana.org/assignments/media-types/media-types.xhtml. CICS creates a Content-Type header for the response by using the value of this attribute.

This attribute has no default, and it must be specified. If the MEDIATYPE attribute specifies a text type, such as a type that begins with `text/`, or a type that contains `+xml`, you must also specify the CHARACTERSET and HOSTCODEPAGE attributes so that code page conversion can take place. Text media types are identified by RFC 3023, which is available at https://www.ietf.org/rfc/rfc3023.txt.

For a dynamic (application-generated) response, this attribute is not used. The media type for the response is specified by the WEB SEND command.

PATH(*path*)
This attribute is for all USAGE options.

PATH specifies the path component of the URI to which the URIMAP definition applies, which can be up to 255 characters, including the forward slash (/) at the beginning of the path component. If you do not include the forward slash, you can use only 254 characters to specify the path. The minimum possible path is a forward slash, which represents the root of the URL structure for the specified host name. You can include or omit the forward slash at the beginning of the path component; however,

if you omit it, CICS adds it at runtime. An example of a path is `software/htp/cics/index.html`. The components of a URL explains each of the components and how they are delimited.

The PATH attribute is specified in mixed case, and the case is preserved in the URIMAP definition. The PATH attribute must contain only the characters that are allowed in URIs. Specifically, the characters *< > # % " { } | \ ^ [ ] `* and embedded blanks must be excluded (except that *%* is allowed when it introduces a valid hexadecimal escape sequence; that is, when it is followed by two valid hexadecimal digits in uppercase or lowercase). The tilde character ~ cannot be specified in CICS and must be replaced by the corresponding hexadecimal escape sequence (%7E). CICS validates the use of characters at define time.

URIMAP resources support Internationalized Resource Identifiers (IRIs), which can contain Unicode characters. If you specify a path that contains Unicode characters, any Unicode characters must be escaped to the percent-encoded representation of the Unicode character. If you do not have an application that can convert Unicode characters to percent-encoded representations, free applications are available on the Internet to do this task. The path must still be 255 characters or less, and a character in this context means a single ASCII character, not the original Unicode character. For example, the Cyrillic character that has the percent-encoded representation %D0%B4 counts as 6 characters from the 255–character limit.

For URIMAP definitions that relate to CICS as an HTTP server and web services, if you want the URIMAP definition to match more than one path, you must use an asterisk as a wildcard character at the end of the path. For example, specifying the path `/software/htp/cics/*` makes the URIMAP definition match all requests with paths that begin with the string to the left of the asterisk. Specifying a path of */** makes the URIMAP definition match any requests that are directed to the host named in the HOST attribute. If an HTTP request is matched by more than one URIMAP definition, the most specific match is taken.

If a query component is present, and you want to apply the URIMAP definition to that specific query alone, you can include it as part of the path component. Include the question mark at the beginning of the string. The query string must contain only the characters that are allowed in URIs. A query string cannot itself include an asterisk as a wildcard, but it can follow a path that includes an asterisk as a wildcard. If you do not include a query string in the URIMAP definition, any query string that is present in the HTTP request is automatically ignored for matching purposes.

For URIMAP definitions for Atom feeds, you must use an asterisk as a wildcard character at the end of the path. The part of the path that you specify in the URIMAP definition is the part that is common to the Atom feed and Atom entry URLs. CICS matches the remainder of the URL to the URLs specified in all the *<atom:link>* elements in the Atom configuration file for the feed.

For URIMAP definitions for CICS as an HTTP client, you cannot use an asterisk as a wildcard; you must specify the complete path for the request. If the URIMAP definition is referenced on a WEB OPEN command, this path becomes the default path for **WEB SEND** commands that relate to that connection. If the URIMAP definition is referenced on a **WEB SEND** command, the path is used for that **WEB SEND** command. However, the host attribute for that URIMAP definition must match the host that is specified on the **WEB OPEN** command for the connection.

PIPELINE(*name*)
This attribute is for USAGE(PIPELINE). This attribute is deprecated for USAGE(JVMSERVER).

When a client makes an inbound web service request to CICS with the URI specified by this URIMAP definition, PIPELINE specifies the 1- to 8-character name of the PIPELINE resource definition for the web service. The PIPELINE resource definition provides information about the message handlers, which act on the service request from the client. PIPELINE resources describes these resource definitions.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

PORT({NO|*port*})
This attribute applies to the USAGE(CLIENT) and USAGE(JVMSERVER) options only.

For USAGE(CLIENT), the PORT attribute specifies the decimal number of the port that is used by a CICS application when it communicates with a server. The value must be a number in the range 1 - 65535.

The port number is combined with the HOST value to determine the destination for outbound requests for this URIMAP. Specify the port number only if it is different from the default for the scheme: 80 for HTTP without SSL, or 443 for HTTPS and HTTP with SSL.

If you specify a port number in the HOST attribute and a different port number in the PORT attribute, an error is returned. If you do not specify a port number in either the HOST or the PORT attribute, the default port number for the scheme is used.

For USAGE(JVMSERVER), the PORT attribute specifies the port number that is used to receive requests to access an application that is running in a Liberty profile server.

If you do not specify a value for the PORT attribute, PORT is set to NO to indicate that the attribute is not being used.

PROGRAM(*name*)
This attribute is for USAGE(SERVER), where an application-generated response is to be provided.

PROGRAM specifies the 1- to 8-character name of the user application program that composes the HTTP response. For CICS as an HTTP server, this attribute is required unless an analyzer or converter program is specified, or a template name or zFS file is specified, or redirection is specified.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

If the ANALYZER attribute is specified as YES, or a converter program is specified in the CONVERTER attribute, the PROGRAM attribute is used as input to the analyzer or converter program, but it can be overridden by those programs. Alternatively, you can leave this attribute blank and let the analyzer or converter program specify it.

REDIRECTTYPE({NONE|TEMPORARY|PERMANENT})
This attribute is for USAGE(SERVER), USAGE(PIPELINE), and USAGE(ATOM).

REDIRECTTYPE specifies the type of redirection for requests that match this URIMAP definition. The URL specified by the LOCATION attribute is used for redirection when required.

- NONE means that requests are not redirected. Any URL specified by the LOCATION attribute is ignored.
- TEMPORARY means that requests are redirected on a temporary basis. The URL specified by the LOCATION attribute is used for redirection, and the status code that is used for the response is 302 (Found).
- PERMANENT means that requests are redirected permanently. The URL specified by the LOCATION attribute is used for redirection, and the status code that is used for the response is 301 (Moved Permanently).

You can use the **SET URIMAP** command to change the REDIRECTTYPE attribute and the LOCATION attribute.

If REDIRECTTYPE(TEMPORARY) or REDIRECTTYPE(PERMANENT) is specified when you are creating a URIMAP definition, the following attributes are optional: ANALYZER, CONVERTER, HFSFILE, PIPELINE, PROGRAM, TEMPLATENAME, TRANSACTION, USERID, and WEBSERVICE. If you use a CEMT or EXEC CICS command to set the REDIRECTTYPE attribute to NONE after the URIMAP

definition is installed, any of the listed attributes that are specified in the URIMAP definition are activated.

SCHEME({HTTP|HTTPS})
This attribute is for all USAGE options.

SCHEME specifies the scheme component of the URI to which the URIMAP definition applies, which is either HTTP (without SSL) or HTTPS (with SSL). Do not include the delimiters `://` (colon and two forward slashes) that follow the scheme component in the URI.

A URIMAP specifying the HTTP scheme accepts web client requests made by using either the HTTP scheme or the HTTPS scheme. A URIMAP specifying the HTTPS scheme accepts only web client requests made by using the HTTPS scheme. However, if the transport is WebSphere MQ, a URIMAP specifying either the HTTP scheme or the HTTPS scheme accepts web client requests made by using either the HTTP scheme or the HTTPS scheme.

Inbound HTTP requests that arrive on SSL(ATTLSAWARE) TCPIPSERVICEs must use secure connections so they are always treated as HTTPS.

SOCKETCLOSE({0|*hhmmss*})
This attribute is for USAGE (CLIENT).

SOCKETCLOSE specifies whether, and for how long, CICS keeps a client HTTP connection open after the CICS application finishes using it. After use, CICS checks the state of the connection and then places it in a pool in a dormant state. A dormant connection can be reused by the same application or by another application that connects to the same host and port.

0
CICS closes each client HTTP connection when the CICS application finishes using it. CICS does not place the connection in a pool for reuse.

hhmmss
When a CICS application finishes using its client HTTP connection, CICS checks the state of the connection and places it in a pool for reuse. A dormant connection that is not reused is discarded after the length of time that you specify here.

Connection pooling can provide performance benefits for the HTTP EP adapter for CICS event processing, or where multiple invocations of CICS web support applications make connection requests for the same host and port, or where a web services application makes multiple requests and responses. To activate connection pooling, your application programs must specify the URIMAP resource on the **INVOKE SERVICE** or **WEB OPEN** command. For more information about connection pooling, see Connection pooling for HTTP client performance.

STATUS({ENABLED|DISABLED})
This attribute is for all USAGE options.

STATUS specifies whether the URIMAP definition is to be installed in an enabled or disabled state. The default is enabled.

This attribute is ignored for URIMAP resources that are dynamically generated by a CICS bundle. The initial status of a URIMAP resource is derived from the initial status of the bundle that defines the resource.

TCPIPSERVICE(*name*)
This attribute is for USAGE(SERVER), USAGE(PIPELINE), and USAGE(ATOM).

TCPIPSERVICE specifies the 1- to 8-character name of a TCPIPSERVICE resource definition, with PROTOCOL(HTTP), that defines an inbound port to which this URIMAP definition relates. If this attribute is not specified, the URIMAP definition applies to a request on any inbound ports.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

When a URIMAP definition with HTTPS as the scheme matches a request that a web client is making, CICS checks that the inbound port used by the request is using SSL. If SSL is not specified for the port, the request is rejected with a 403 (Forbidden) status code. When the URIMAP definition applies to all inbound ports, this check ensures that a web client cannot use an unsecured port to access a secured resource. No check is carried out for a URIMAP definition that specifies HTTP as the scheme, so web clients can use either unsecured or secured (SSL) ports to access these resources.

You specify the security measures that are applied for each port in the TCPIPSERVICE resource definition. You can choose whether to use SSL, and, if you do use SSL, you need to choose the exact security measures that are applied, for example, the authentication method, the sending of certificates by client and server, and encryption of messages. Read Security for CICS web support for more information about the security features that you can use to keep your CICS web support facility safe.

TEMPLATENAME(*name*)

This attribute is for USAGE(SERVER), where a static response is to be provided.

TEMPLATENAME specifies the 1- to 48-character name of a CICS document template that forms the body of the static response that is sent to the HTTP request from the web client. It must be defined by using a DOCTEMPLATE resource definition, and the TEMPLATENAME attribute of that definition specifies the name that is used in the URIMAP definition. For more information, see CICS documents and document templates.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

If you specify TEMPLATENAME or HFSFILE, you must set ANALYZER to NO. The other attributes that relate only to application-generated responses (TRANSACTION, CONVERTER, and PROGRAM) must remain blank.

If you want to use path matching, include an asterisk as a wildcard character at the end of the name of the CICS document template and also at the end of the path that is specified by the PATH attribute. CICS takes the portion of each HTTP requests path that is covered by the wildcard character and substitutes it as the last part of the template name.

For example, you can create a URIMAP definition with the PATH attribute specified as:

```
findout/about/*
```

and the TEMPLATENAME attribute that is specified as:

```
templates.facts.*
```

The URIMAP definition is used to process an incoming HTTP request

```
http://www.example.com/findout/about/fish.html
```

CICS appends `fish.html` to the template name specified in the URIMAP definition in place of the asterisk, so that the template

```
templates.facts.fish.html
```

is used to form the static response.

Specifying an asterisk alone for the TEMPLATENAME attribute means that the selected template has the same name as the part of the URL that corresponds to the wildcard character in the PATH attribute.

When you specify the TEMPLATENAME attribute, if a query string is present on the URI, but is not used in the PATH attribute, CICS automatically passes the content of the query string into the named CICS document template as a symbol list. If you want to use the content of the query string in the document template, include appropriate variables in your document template to be substituted for the content of the query string.

TRANSACTION(*name*)
This attribute is for USAGE(SERVER), USAGE(PIPELINE) USAGE(ATOM), and USAGE(JVMSERVER).

TRANSACTION specifies the 1-4 character name of an alias transaction that is to be used to run the application, or to start the pipeline.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

The default alias transaction is different for each type of URIMAP:

- USAGE(SERVER) uses the CWBA transaction
- USAGE(PIPELINE) uses the CPIH transaction
- USAGE(ATOM) uses the CW2A transaction
- USAGE(JVMSERVER) uses the CJSA transaction

You can select a different transaction name for the purposes of security, monitoring and accounting, or transaction class limitation. Whatever name you choose for the alias transaction, it must always run the same program, which is determined by the USAGE attribute, and which overrides any program specified in the transaction definition:

- For USAGE(SERVER), the program is DFHWBA, which links to the application program named in the PROGRAM attribute of the URIMAP definition or named by the analyzer program.
- For USAGE(PIPELINE), the program is DFHPIDSH, which starts the pipeline that is named in the PIPELINE attribute and the web service that is named in the WEBSERVICE attribute, if specified.
- For USAGE(ATOM), the program is DFHW2A, the W2 domain alias program.
- For USAGE(JVMSERVER), the program is DFHSJTHP, which runs tasks in the JVM server for an application request.

For USAGE(SERVER) only, if the ANALYZER attribute is specified as YES, the TRANSACTION attribute is used as input to the analyzer program, but the analyzer program can override it. Alternatively, you can leave this attribute blank and let the analyzer program specify it. The analyzer is not used for other types of URIMAP.

URIMAP(*name*)
Specifies the name of this URIMAP definition. The name can be up to 8 characters in length. The attribute is specified in mixed case, and the case is preserved in the URIMAP definition.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

USAGE({SERVER|CLIENT|PIPELINE|ATOM|JVMSERVER})
Specifies whether this URIMAP definition is for CICS as an HTTP server (SERVER), CICS as an HTTP client (CLIENT), a web service (PIPELINE), an Atom feed (ATOM), or a JVM server (JVMSERVER). The USAGE attribute governs which other attributes in the URIMAP definition can be used.

- When you specify SERVER, you create a URIMAP definition for CICS as an HTTP server. This type of URIMAP definition is used to map the URI of an incoming HTTP request from a web client to CICS resources. An application-generated response or a static response can be provided. Requests that require an application-generated response might qualify for being processed by directly attached

user transactions, and bypassing the web attach task. For more information, see HTTP requests are processed by directly attached user transactions.

- When you specify CLIENT, you create a URIMAP definition for CICS as an HTTP client. This type of URIMAP definition is used when CICS makes a request for an HTTP resource on a server, so that you can avoid identifying the URI in your application program.
- When you specify PIPELINE, you create a URIMAP definition for a web service. This type of URIMAP definition is used for an inbound web service request; that is, a request by which a client invokes a web service in CICS. The URI of the incoming request is associated with WEBSERVICE and PIPELINE resources, which specify the processing that is to be performed on the message. Requests might qualify for being processed by directly attached user transactions, and bypassing the web attach task. For more information, see HTTP requests are processed by directly attached user transactions.
- When you specify ATOM, you create a URIMAP definition for an Atom feed. This type of URIMAP definition is used for an incoming request for data that CICS makes available as an Atom feed. The URIMAP definition maps the request URI to an ATOMSERVICE resource definition, which defines an Atom document. Requests might qualify for being processed by directly attached user transactions, and bypassing the web attach task. For more information, see HTTP requests are processed by directly attached user transactions.
- When you specify JVMSERVER, you create a URIMAP for a web application that is running in a JVM server. This type of URIMAP is used to map incoming application requests to a CICS transaction. If no URIMAP matches the URI of the incoming request, CICS uses the CJSA transaction.

USERID(*id*)

This attribute is for USAGE(SERVER), where an application-generated response is to be provided, USAGE(PIPELINE), USAGE(JVMSERVER), and USAGE(ATOM).

USERID specifies a 1- to 8-character default user ID that can be used by any client. For an application-generated response or a web service, the alias transaction is attached under this user ID.

For USAGE(JVMSERVER), the user ID is used to attach a transaction to run work in the Liberty profile server. The transaction can be specified in the URI map or use the default transaction CJSA. The user ID is not used if basic authentication is enabled; CICS expects a user ID and password in the HTTP header for the application request and rejects the request if authentication fails. If security is not enabled and no user ID is present in the HTTP header or in the URI map, the default user ID for the CICS region is used.

For other usage types, when authentication is required for the connection, so that CICS requests an authenticated user ID directly from the client, the default user ID that you specify in the URIMAP definition is not used. The authenticated user ID of the client is used instead, or if authentication fails, the request is rejected. Authentication procedures are specified by the AUTHENTICATE attribute of the TCPIPSERVICE definition for the connection.

For an application-generated response, if ANALYZER(YES) is specified, the USERID attribute is used as input to the analyzer program, but the analyzer program can override it. Alternatively, you can leave this attribute blank and let the analyzer program specify it. The analyzer is used only when USAGE(SERVER) is specified. A user ID specified by a client can also be changed by the analyzer program. If no user ID is specified by any of the methods that are mentioned, the default user ID for an application-generated response is the CICS default user ID.

For static responses, the USERID attribute does not apply. Resource security checking for static responses can be carried out by using the authenticated user ID of a client.

If surrogate user checking is enabled in the CICS region, with XUSER=YES specified as a system initialization parameter, CICS checks that the user ID used to install the URIMAP definition is authorized as a surrogate of the user ID specified for the USERID attribute. For more information about surrogate user checking, see Where surrogate user checking applies .

WEBSERVICE(*name*)

This attribute is for USAGE(PIPELINE). This attribute is deprecated for USAGE(JVMSERVER).

When a client makes an inbound web service request to CICS with the URI specified by this URIMAP definition, WEBSERVICE specifies the name of the web service. This name can be the 1- to 8-character name of a WEBSERVICE resource definition, or a name up to 32 characters in mixed case that represents a web service that is generated by the CICS web services assistant.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

A WEBSERVICE resource defines aspects of the runtime environment for a CICS application program that is deployed in a web services setting, where the mapping between application data structure and SOAP messages is generated by using the CICS tools.

This attribute is optional. However, if you do not specify a WEBSERVICE resource for USAGE(PIPELINE), the WEBSERVICE name must be resolved by a handler program in the pipeline, or you must use an alternative application or terminal handler in place of a WEBSERVICE resource.

Installing URIMAP resource definitions

This procedure uses the CEMT and CEDA transactions to install a URIMAP resource definition. If the URIMAP resource already exists, it has to be disabled before it can be reinstalled.

Procedure

1. If the URIMAP resource already exists, ensure that it is disabled.

   Use the following command:

   ```
   CEMT SET URIMAP(name) DISABLED
   ```

   While the URIMAP resource is disabled, if a web client makes an HTTP request that requires the resource, CICS issues error message DFHWB0763, and returns an HTTP 503 response (Service Unavailable) to the web client through a web error program. You can tailor this response by changing the web error program.
2. Install the URIMAP definition.

   Use the following command:

   ```
   CEDA INSTALL GROUP(groupname) URIMAP(name)
   ```

   When you install a URIMAP definition, CICS carries out the following security checks:

   - If the URIMAP definition specifies SCHEME(HTTPS), CICS checks at install time that SSL is active in the CICS region. This is indicated by the use of the KEYRING system initialization parameter to specify the key ring used by the CICS region. If SSL is not active in the CICS region, CICS issues message DFHAM4905, and the URIMAP definition is not installed.
   - If the URIMAP definition specifies the CIPHERS attribute, CICS validates the list of ciphers against the ciphers supported in the running system. If no valid ciphers are found in the list, CICS issues message DFHAM4918 and the URIMAP definition is not installed. However, if some but not all of the ciphers in the list are supported, CICS issues message DFHAM4917 and the URIMAP is installed with the reduced set of cipher codes.
   - If the URIMAP definition specifies the CERTIFICATE attribute, CICS validates the certificate against those specified in the key ring. If the specified certificate is not valid, then CICS issues messages DFHAM4889 and DFHAM4928, and the URIMAP definition is not installed.

     Tip: CICS validates the certificate against the information held in the key ring for the CICS region in the external security manager's database. When the CICS region carries out SSL handshakes, it uses information from the cache of certificates in the SSL environment for the CICS region, which is managed by z/OS System SSL. If you have added this certificate to the key ring, or renewed it, since the last build or rebuild of the SSL environment for the CICS region, issue the PERFORM

SSL REBUILD command for the CICS region. The command refreshes the cache of certificates and ensures that the correct information for this certificate is present in the cache.

3. Optional: When you have successfully installed the URIMAP definition, use CEMT to enable the resource.

   Perform this step only if the URIMAP resource is not already defined as ENABLED, and you want to make the resource available for web clients or web services. Use the following command:

   ```
   CEMT SET URIMAP(name) ENABLED
   ```

URIMAP resource definitions

A URIMAP resource definition matches the URIs of HTTP or web service requests.

WEBSERVICE resources

A WEBSERVICE resource defines aspects of the run time environment for a CICS application program deployed in a web services setting, where the mapping between application data structure and SOAP messages has been generated using the CICS web services assistant. Although CICS provides the usual resource definition mechanisms for WEBSERVICE resources, they are typically installed dynamically, using the output produced by the assistant.

The aspects of the run time environment that are defined by the WEBSERVICE resource are:

A pipeline
: Defines the set of message handlers that operate on web service requests and responses. The WEBSERVICE resource specifies a separate PIPELINE resource which, in turn, specifies the pipeline configuration file.

A web service binding file
: Contains information that is used at run time to perform the mapping between application data structures and SOAP messages. The web service binding file is generated by the CICS-supplied tools.

A web service description
: The web services description is used only when runtime validation of SOAP messages is required. Validation of each message is performed against its schema, which is imbedded within the web service description.

An inbound web service request (that is, a request by which a client invokes a web service in CICS) is associated with a WEBSERVICE resource by the URIMAP resource. The URIMAP identifies the WEBSERVICE resource that applies to the URI in the inbound message; the WEBSERVICE specifies the processing that is to be performed on the message.

Although CICS provides the usual resource definition mechanisms for creating WEBSERVICE resources, and installing them in your CICS region, you can instead use the scanning mechanism to dynamically install WEBSERVICE resources in your running CICS system. The advantages of this approach are that it reduces the amount of resource definition required, and that CICS can make direct use of information provided at development time.

To invoke the scanning mechanism, use the **PERFORM PIPELINE** command.

The name of a dynamically-installed WEBSERVICE is derived from the name of the web service binding file from which the WEBSERVICE definition is generated, and has a maximum length of 32 characters; the names of WEBSERVICE definitions installed from the CSD or with the **EXEC CICS CREATE WEBSERVICE** are limited to eight characters. For example, a Webservice binding file whose zFS name is `/samples/Webservices/WSDir/InquireSingle.wsbind` generates a WEBSERVICE definition named `InquireSingle`

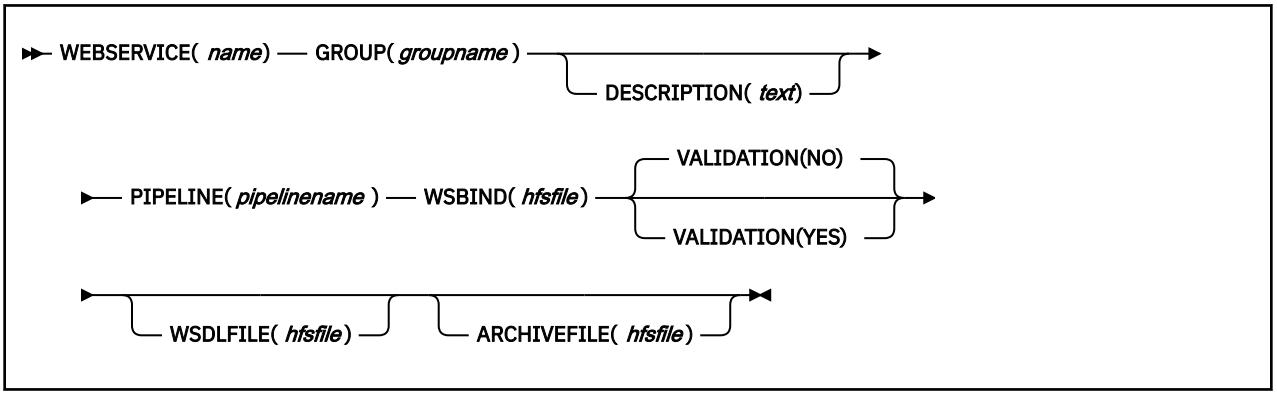
WEBSERVICE resources in CICS bundles

You can use a CICS bundle to create, edit, and install a WEBSERVICE resource definition. If you create a WEBSERVICE resource in this way, you must use the CICS bundle to manage the lifecycle of that resource, and you cannot manage the resource independently.

You can inquire on a WEBSERVICE resource that is dynamically generated by a CICS bundle, by using the **EXEC CICS INQUIRE WEBSERVICE** or **CEMT INQUIRE WEBSERVICE** command. However, you cannot issue a **DISCARD** command against a dynamically generated WEBSERVICE resource. You must issue the command against the BUNDLE resource, and CICS applies it to the WEBSERVICE resource. For more information about defining resources in CICS bundles, see Defining CICS bundles.

WEBSERVICE attributes

Describes the syntax and attributes of the WEBSERVICE resource.



WEBSERVICE(*name*)

Specifies the 1-8 character name of the WEBSERVICE resource.

Acceptable characters:

```
A-Z a-z 0-9 $ @ # . / - _ % & ? ! : | " = ¬ , ; < >
```

It is preferable not to use names that start with DFH because these characters are reserved for use by CICS. CICS can delete or discard them and replace them with CICS resources when you upgrade or provide new function or fixes.

ARCHIVEFILE(*hfsfile*)

Specifies the 1-255 character fully qualified or relative name of an archive that contains one or more WSDL files. The supported format for the archive is `.zip`.

- For an archive for a WEBSERVICE resource defined by online resource definition, the file path is either fully qualified, if prefixed with a forward slash (/), or is relative to the HOME directory of the CICS region user ID.
- For an archive for a WEBSERVICE resource defined in a CICS bundle, the file path is relative to the root directory of the CICS bundle. The zFS file must be packaged in the CICS bundle with the WEBSERVICE resource. For more information, see Referencing zFS artifacts in a bundle.

The value specified must be a valid name for a UNIX file:

- It must not contain embedded space characters.
- It must not contain consecutive instances of the / character.
- It is case-sensitive.

Acceptable characters:

```
A-Z a-z 0-9 . / _ # @ -
```

GROUP(*groupname*)

Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Any lowercase characters you enter are converted to uppercase.

The GROUP name can be up to eight characters in length. Lowercase characters are treated as uppercase characters.

DESCRIPTION(*text*)
You can provide a description of the resource that you are defining in this field. The description text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right parenthesis. If you use the CREATE command, for each single apostrophe in the text, code two apostrophes.

PIPELINE(*pipelinename*)
Specifies the 1-8 character name of the PIPELINE resource with which this WEBSERVICE resource is associated.

Acceptable characters:

```
A-Z 0-9 $ @ #
```

Unless you are using the CREATE command, any lowercase characters that you enter are converted to uppercase.

VALIDATION(NO|YES)
Specifies whether full validation of SOAP messages against the corresponding schema in the web service description should be performed at run time. Validating a SOAP message against its schema incurs considerable processing overhead, and you should normally specify VALIDATION(NO).

Full validation ensures that all SOAP messages that are sent and received are valid XML with respect to the XML schema. If VALIDATION(NO) is specified, checking is performed to ensure that the message contains well-formed XML, but there is no guarantee that the XML is valid.

WSBIND(*hfsfile*)
Specifies the 1-255 character fully-qualified or relative name of the web service binding file on z/OS UNIX.

- For a web service binding file for a WEBSERVICE resource defined by online resource definition, the file path is either fully qualified, if prefixed with a forward slash (/), or is relative to the HOME directory of the CICS region user ID.
- For a web service binding file for a WEBSERVICE defined in a CICS bundle, the file path is relative to the root directory of the CICS bundle. The zFS file must be packaged in the CICS bundle with the WEBSERVICE resource. For more information, see Referencing zFS artifacts in a bundle.

The value specified must be a valid name for a UNIX file:

- It must not contain embedded space characters.
- It must not contain consecutive instances of the / character.
- It is case-sensitive.

Acceptable characters:

```
A-Z a-z 0-9 . / _ # @ -
```

WSDLFILE(*hfsfile*)
Specifies the 1-255 character fully-qualified or relative name of the web service description (WSDL) file on z/OS UNIX. This file is used when full runtime validation is active.

- For a web service description file for a WEBSERVICE resource defined by online resource definition, the file path is either fully qualified, if prefixed with a forward slash (/), or is relative to the HOME directory of the CICS region user ID.
- For a web service description file for a WEBSERVICE resource defined in a CICS bundle, the file path is relative to the root directory of the CICS bundle. The zFS file must be packaged in the CICS bundle with the WEBSERVICE resource. For more information, see Referencing zFS artifacts in a bundle.

The value specified must be a valid name for a UNIX file:

- It must not contain embedded space characters.
- It must not contain consecutive instances of the / character.
- It is case-sensitive.

Acceptable characters:

```
A-Z a-z 0-9 . / _ # @ -
```

Installing WEBSERVICE resource definitions

Although CICS provides the usual resource definition mechanisms for creating WEBSERVICE resources, and installing them in your CICS region, there is an alternative strategy which you can use. You can use the scanning mechanism to install WEBSERVICE resources in your running CICS system.

About this task

There are three ways to install a WEBSERVICE resource definition.

Procedure

- Use the **PERFORM PIPELINE SCAN** command (using the CEMT transaction, or the system programming interface) to initiate a scan of the pickup directory for a PIPELINE resource.

  Use this method when you have added or updated a Web service binding file in the pickup directory of a PIPELINE that is already been installed. CICS scans the pickup directory and uses each Web service binding file that it finds there to dynamically install a WEBSERVICE resource.
- Install a PIPELINE resource.

  Use this method when the pickup directory of a PIPELINE resource contains a Web service binding file that you want to associate with the PIPELINE, and the PIPELINE has not been installed. When you install the PIPELINE, CICS scans the pickup directory and uses each Web service binding file that it finds there to install a WEBSERVICE resource.
- Install a WEBSERVICE resource from the CSD.

  Use this method if neither of the two other methods is appropriate - for example, when you want to use a PIPELINE definition with a Web service binding file that is not in the PIPELINE's pickup directory.

What to do next

If you have updated the Web service binding file that is referenced by a WEBSERVICE definition installed from the CSD, you must discard and reinstall the WEBSERVICE to pick up the new Web service binding file.

Web service resource definitions

A web service resource definition defines aspects of the runtime environment for a CICS application program deployed as a web service, where mapping between application data structure and SOAP messages has been generated using the CICS web services assistant.

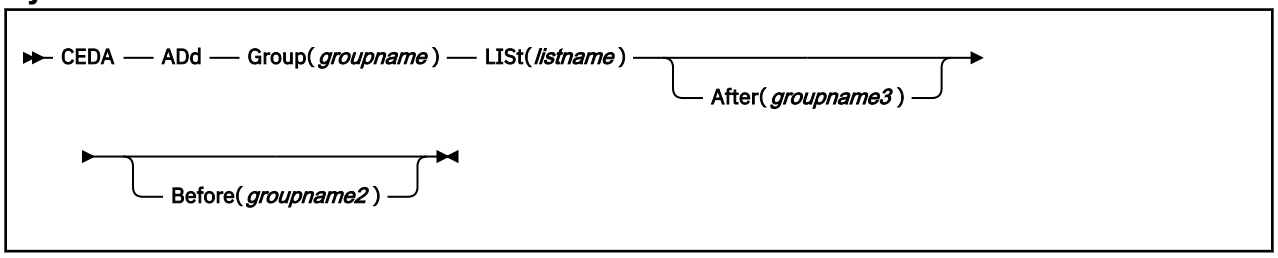
Chapter 2. Resource management transaction CEDA commands

The CEDA transaction has a number of commands for working with resource definitions on the CSD.

The CEDA ADD command

Use the **CEDA ADD** command to add a group to a list on the CSD.

Syntax



Description

You can use the ADD command from a DISPLAY screen.

Options

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

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

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

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

Examples

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

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

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

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

   LA01 now looks like this:

   GA001
   GA002

3. To add another group at the beginning of the list:

© Copyright IBM Corp. 1974, 2023

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

and another group between GA001 and GA002:

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

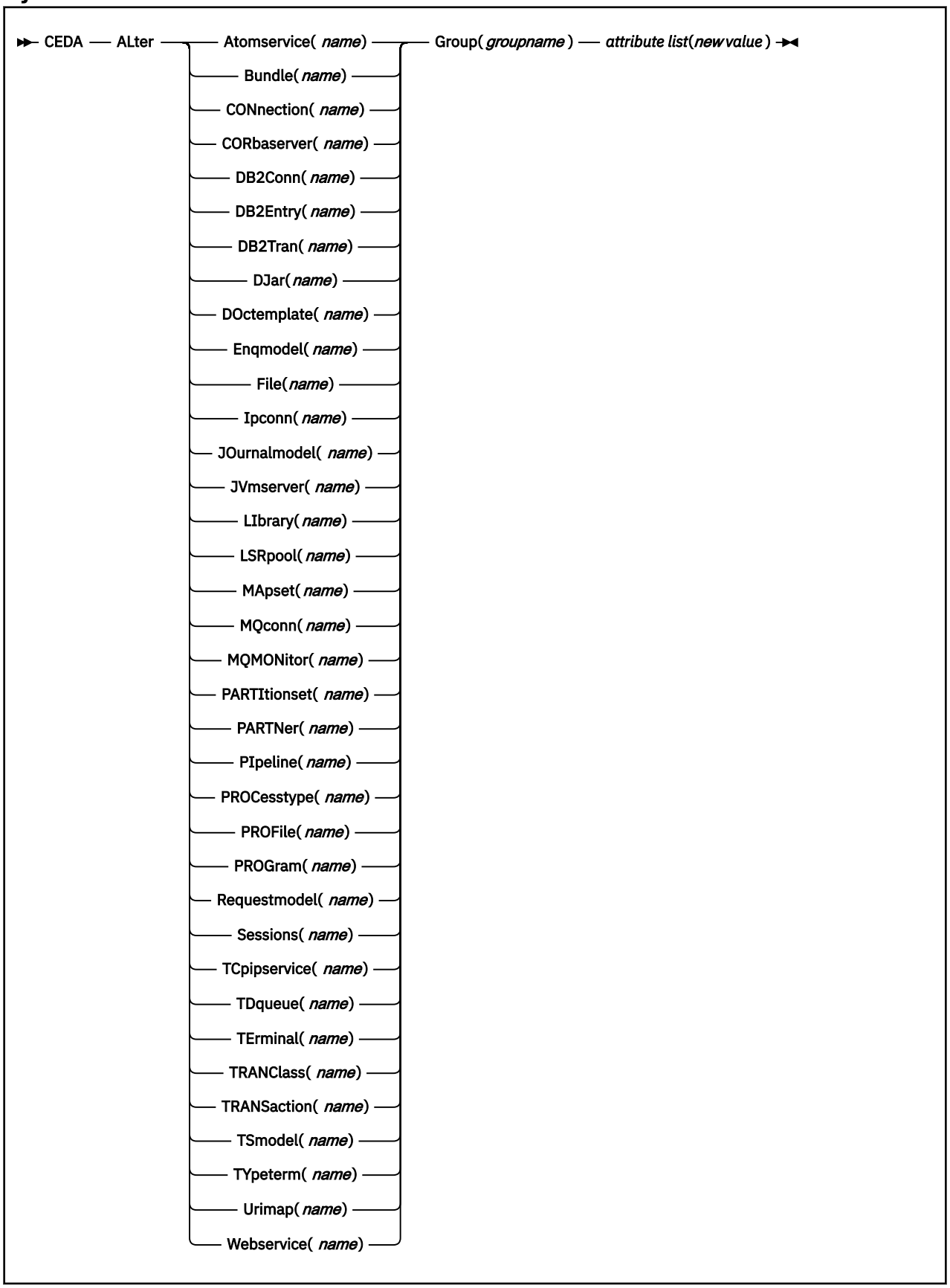
LA01 now looks like this:

GA003
GA001
GA004
GA002

The CEDA ALTER command

Use the **CEDA ALTER** command to change some or all of the attributes of an existing resource definition.

Syntax


CEDA
ALter
Atomservice(name)
Bundle(name)
CONnection(name)
CORbaserver(name)
DB2Conn(name)
DB2Entry(name)
DB2Tran(name)
DJar(name)
DOctemplate(name)
Enqmodel(name)
File(name)
Ipconn(name)
JOurnalmodel(name)
JVmserver(name)
LIbrary(name)
LSRpool(name)
MApset(name)
MQconn(name)
MQMONitor(name)
PARTItionset(name)
PARTNer(name)
PIpeline(name)
PROCesstype(name)
PROFile(name)
PROGram(name)
Requestmodel(name)
Sessions(name)
TCpipservice(name)
TDqueue(name)
TErminal(name)
TRANClass(name)
TRANSaction(name)
TSmodel(name)
TYpeterm(name)
Urimap(name)
Webservice(name)
Group(groupname)
attribute list(new value)

Description

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

You can specify null attribute values, for example:

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

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

- The command:

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

  behaves as if RLSACCESS was not specified. The RLSACCESS attribute has a CVDA default value which is ignored.
- The command:

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

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

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

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

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

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

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

Options

Attribute list
specifies the attributes to be altered.

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

***resource*(*name*)**
specifies the type and name of the resource whose attributes you want to alter.

Examples

- To make a program resident:

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

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

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

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

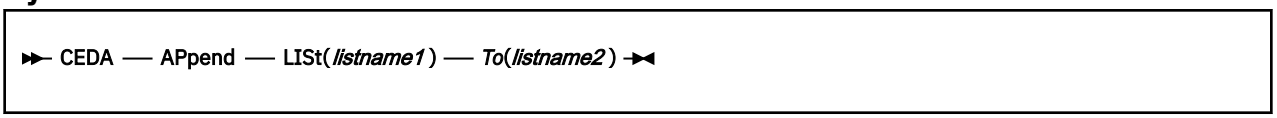
- To change the status of a whole group of programs:

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

The CEDA APPEND command

Use the **CEDA APPEND** command to append the groups in one list on the CSD to the end of another list.

Syntax



Options

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

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

Example

A list called LISTA contains the following groups:

GB001
GB002
GB003

A list called LISTB contains the following groups:

G001
G002
G003

Append LISTB to LISTA, like this:

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

After this, LISTA contains the following groups, in this order:

GB001
GB002
GB003
G001
G002
G003

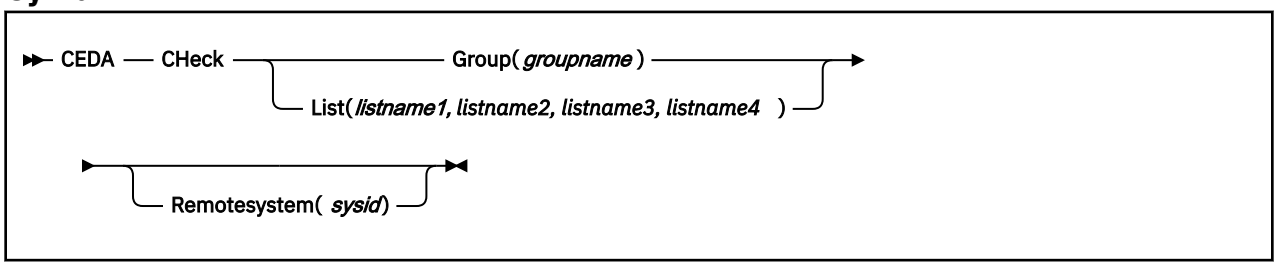
and LISTB still contains:

G001
G002
G003

The CEDA CHECK command

Use the **CEDA CHECK** command to check the consistency of a set of resource definitions on the CSD.

Syntax



Description

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

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

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

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

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

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

You can use the CHECK command from a DISPLAY panel.

How CEDA checks the definitions

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

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

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

Options

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

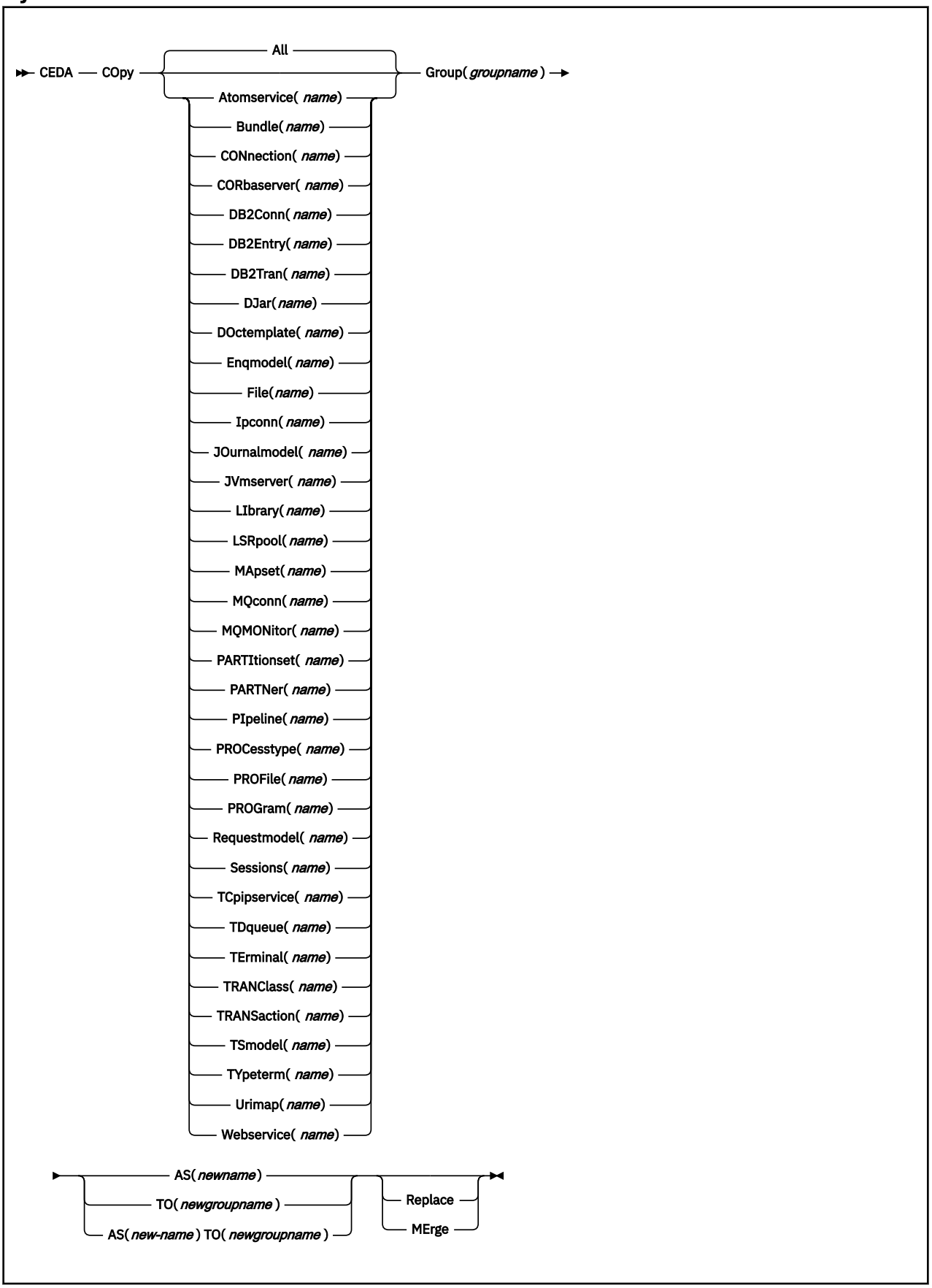
List(*listname1, listname2, etc.*)
specifies the list or lists you want to check. A generic list name is not accepted.

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

The CEDA COPY command

Use the **CEDA COPY** command to copy a resource definition, either within the same group or to a different group on the CSD.

Syntax


CEDA
COpy
All
Atomservice(name)
Bundle(name)
CONnection(name)
CORbaserver(name)
DB2Conn(name)
DB2Entry(name)
DB2Tran(name)
DJar(name)
DOctemplate(name)
Enqmodel(name)
File(name)
Ipconn(name)
JOurnalmodel(name)
JVmserver(name)
LIbrary(name)
LSRpool(name)
MApset(name)
MQconn(name)
MQMONitor(name)
PARTItionset(name)
PARTNer(name)
PIpeline(name)
PROCesstype(name)
PROFile(name)
PROGram(name)
Requestmodel(name)
Sessions(name)
TCpipservice(name)
TDqueue(name)
TErminal(name)
TRANClass(name)
TRANSaction(name)
TSmodel(name)
TYpeterm(name)
Urimap(name)
Webservice(name)
Group(groupname)
AS(newname)
TO(newgroupname)
AS(new-name) TO(newgroupname)
Replace
MErge

Description

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

Options

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

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

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

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

***resource*(*name*)**
specifies the type and name of the resource whose attributes you want to copy.

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

Examples

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

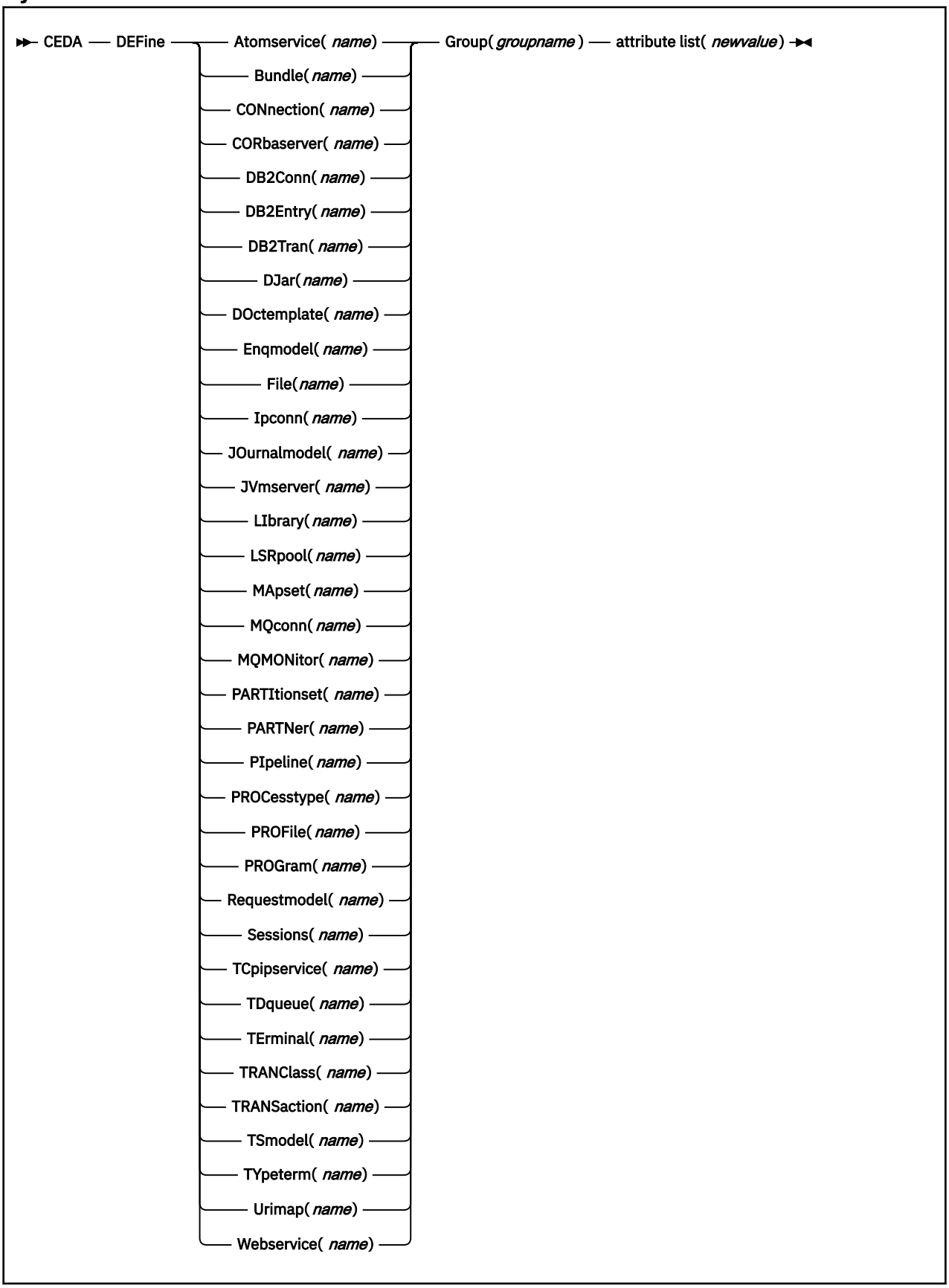
The following example copies group GA003 to group GA004, but if any duplicate definitions occur, preserves the group GA004 definitions.

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

The CEDA DEFINE command

Use the **CEDA DEFINE** command to create new resource definitions on the CSD.

Syntax


CEDA
DEFine
Atomservice(name)
Bundle(name)
CONnection(name)
CORbaserver(name)
DB2Conn(name)
DB2Entry(name)
DB2Tran(name)
DJar(name)
DOctemplate(name)
Enqmodel(name)
File(name)
Ipconn(name)
JOurnalmodel(name)
JVmserver(name)
LIbrary(name)
LSRpool(name)
MApset(name)
MQconn(name)
MQMONitor(name)
PARTItionset(name)
PARTNer(name)
PIpeline(name)
PROCesstype(name)
PROFile(name)
PROGram(name)
Requestmodel(name)
Sessions(name)
TCpipservice(name)
TDqueue(name)
TErminal(name)
TRANClass(name)
TRANSaction(name)
TSmodel(name)
TYpeterm(name)
Urimap(name)
Webservice(name)
Group(groupname)
attribute list(newvalue)

Description

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

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

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

When you have any resource definitions with the same name, make sure that you install the one you really want. See Duplicate resource definition names.

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

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

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

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

Beware of overtyping values on which other attribute values depend.

Options

Attribute list
Depends on the resource type being defined; some resources have attributes that must be included in the definition. Attributes that you do not specify are given default values.

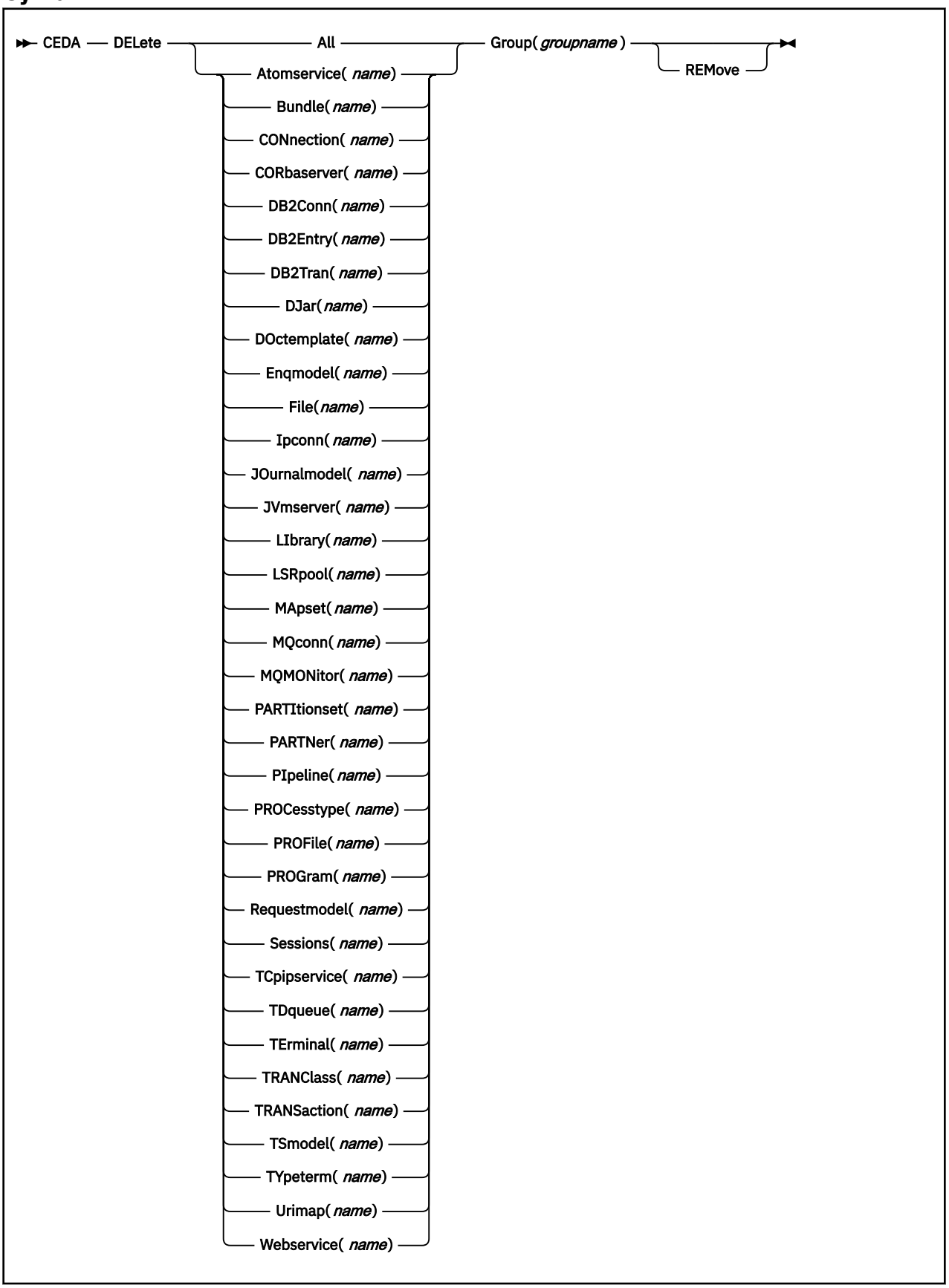
Group(*groupname*)
Specifies the name of the group containing the resource definition being defined. Do not use a generic group name. If you specify the name of a group that does not already exist, the group is created.

***resource*(*name*)**
Specifies the type and name of the resource that you want to define. Do not use a generic resource name.

The CEDA DELETE command

Use the **CEDA DELETE** command to delete one or more resource definitions from the CSD file.

Syntax


CEDA
DELete
All
Atomservice(name)
Bundle(name)
CONnection(name)
CORbaserver(name)
DB2Conn(name)
DB2Entry(name)
DB2Tran(name)
DJar(name)
DOctemplate(name)
Enqmodel(name)
File(name)
Ipconn(name)
JOurnalmodel(name)
JVmserver(name)
LIbrary(name)
LSRpool(name)
MApset(name)
MQconn(name)
MQMONitor(name)
PARTItionset(name)
PARTNer(name)
PIpeline(name)
PROCesstype(name)
PROFile(name)
PROGram(name)
Requestmodel(name)
Sessions(name)
TCpipservice(name)
TDqueue(name)
TErminal(name)
TRANClass(name)
TRANSaction(name)
TSmodel(name)
TYpeterm(name)
Urimap(name)
Webservice(name)
Group(groupname)
REMove

Description

Deleting a resource definition is different from removing a group from a list (see “The CEDA REMOVE command” on page 309). A deleted resource definition really does disappear from the CSD file.

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

This command does **not** affect definitions installed on the active CICS system. To remove installed definitions from the active CICS system, you can use either the CEMT DISCARD transaction or the EXEC CICS DISCARD command.

Options

All
specifies that all resources are to be deleted from the group.

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

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

***resource*(*name*)**
specifies the type and name of the resource you want to delete. You can use a generic resource name.

Examples

- To delete all resources in a group:

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

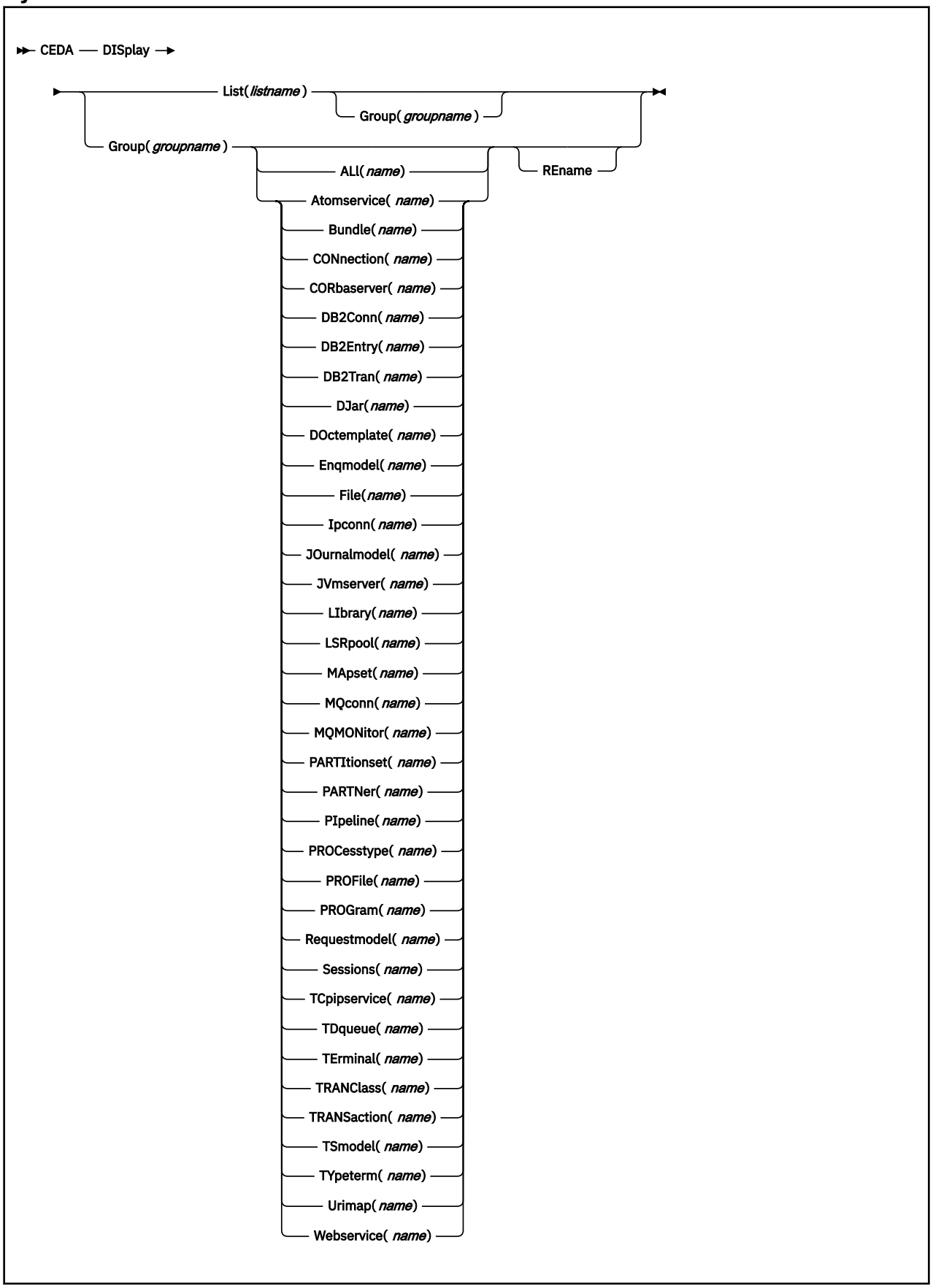
- To delete all programs in a group:

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

The CEDA DISPLAY command

Use the **CEDA DISPLAY** command to display one or more group names, list names, or resources.

Syntax


CEDA
DISplay
List(listname)
Group(groupname)
Group(groupname)
ALl(name)
REname
Atomservice(name)
Bundle(name)
CONnection(name)
CORbaserver(name)
DB2Conn(name)
DB2Entry(name)
DB2Tran(name)
DJar(name)
DOctemplate(name)
Enqmodel(name)
File(name)
Ipconn(name)
JOurnalmodel(name)
JVmserver(name)
LIbrary(name)
LSRpool(name)
MApset(name)
MQconn(name)
MQMONitor(name)
PARTItionset(name)
PARTNer(name)
PIpeline(name)
PROCesstype(name)
PROFile(name)
PROGram(name)
Requestmodel(name)
Sessions(name)
TCpipservice(name)
TDqueue(name)
TErminal(name)
TRANClass(name)
TRANSaction(name)
TSmodel(name)
TYpeterm(name)
Urimap(name)
Webservice(name)

Options

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

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

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

***resource*(*name*)**
specifies the type and name of the resource you want to display.

Examples

- To display all groups on the CSD file:

```
DISPLAY GROUP(*)
```

- To display all groups with seven-character names that begin with PROD:

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

- To display all the lists on the CSD file:

```
DISPLAY LIST(*)
```

- To display all lists with five-character names that begin with PROD:

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

The CEDA DISPLAY GROUP command

Use the **CEDA DISPLAY GROUP** command to display one or more group names on a full screen panel.

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

CHECK
DISPLAY ALL
EXPAND
INSTALL
LOCK
UNLOCK

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

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

To DISPLAY the group names, you must use a generic name.

The CEDA DISPLAY LIST command

Use the **CEDA DISPLAY LIST** command to display one or more list names on a full screen panel.

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

CHECK
DISPLAY GROUP
EXPAND
LOCK

UNLOCK

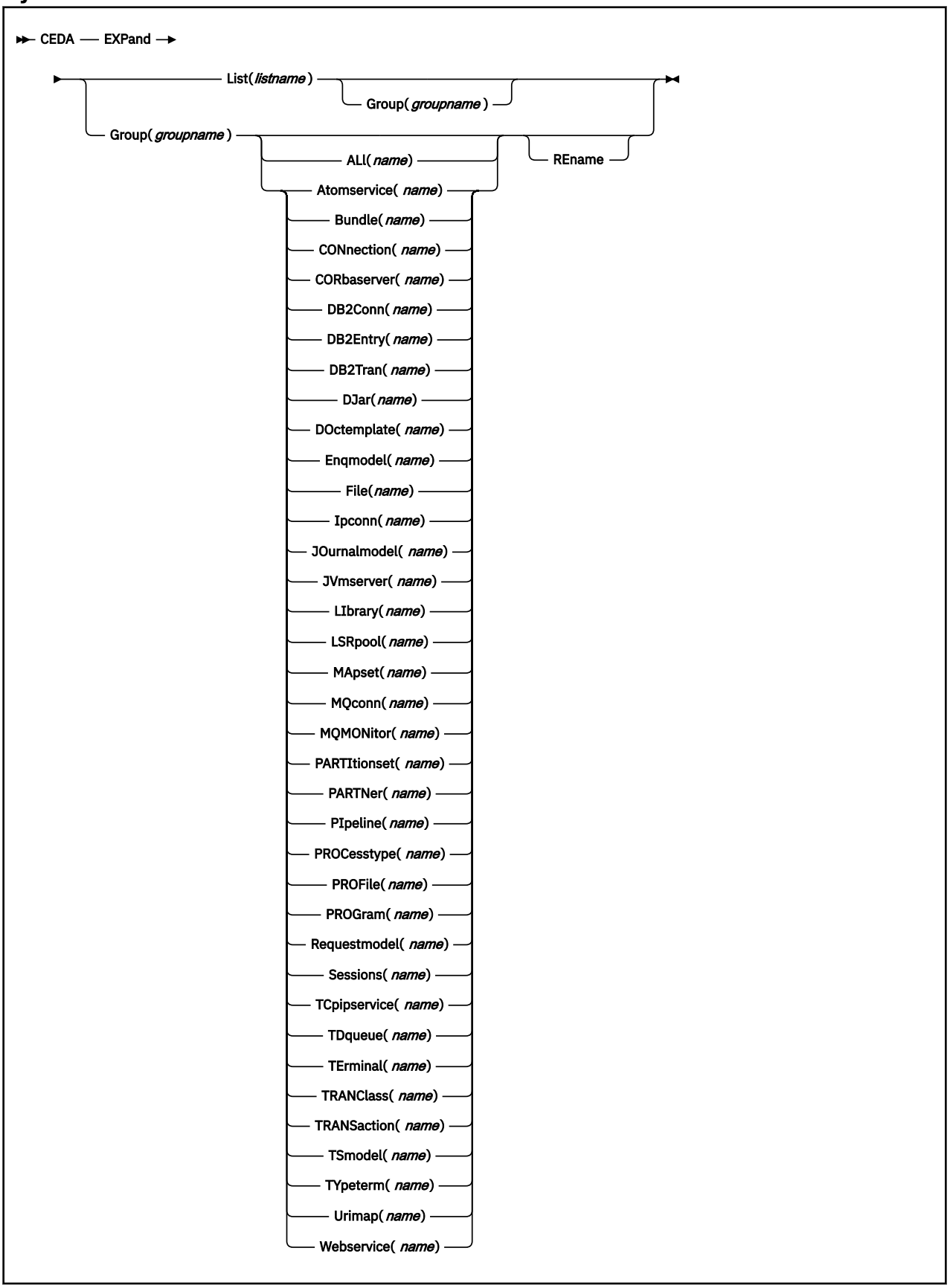
On this panel, all these commands can be abbreviated down to their initial letter. It is not necessary to type the list name. For the syntax of each command, see that command in this section.

To DISPLAY the list names, you must use a generic list name.

The CEDA EXPAND command

Use the **CEDA EXPAND** command to display the resource definitions that are contained in one or more groups or lists. You can enter other CEDA commands against the resource names that are displayed.

Syntax


CEDA
EXPand
List(listname)
Group(groupname)
Group(groupname)
ALl(name)
REname
Atomservice(name)
Bundle(name)
CONnection(name)
CORbaserver(name)
DB2Conn(name)
DB2Entry(name)
DB2Tran(name)
DJar(name)
DOctemplate(name)
Enqmodel(name)
File(name)
Ipconn(name)
JOurnalmodel(name)
JVmserver(name)
LIbrary(name)
LSRpool(name)
MApset(name)
MQconn(name)
MQMONitor(name)
PARTItionset(name)
PARTNer(name)
PIpeline(name)
PROCesstype(name)
PROFile(name)
PROGram(name)
Requestmodel(name)
Sessions(name)
TCpipservice(name)
TDqueue(name)
TErminal(name)
TRANClass(name)
TRANSaction(name)
TSmodel(name)
TYpeterm(name)
Urimap(name)
Webservice(name)

Options

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

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

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

***resource*(*name*)**
specifies the type and name of the resource you want to see within the expanded group.

The CEDA EXPAND GROUP command

Use the **EXPAND GROUP** command to display the resource definitions that are contained in one or more groups. You can enter other CEDA commands against the resource names that are displayed.

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

ALTER
COPY
DELETE
INSTALL
MOVE
RENAME
VIEW

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

You may EXPAND a generic group name. For example:

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

```
EXPAND GROUP(*)
```

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

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

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

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

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

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

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

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

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

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

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

The CEDA EXPAND LIST command

Use the **EXPAND LIST** command to display the groups that are contained in one or more lists. You can enter other CEDA commands against the group names that are displayed.

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

ADD
REMOVE

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

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

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

```
EXPAND LIST(*)
```

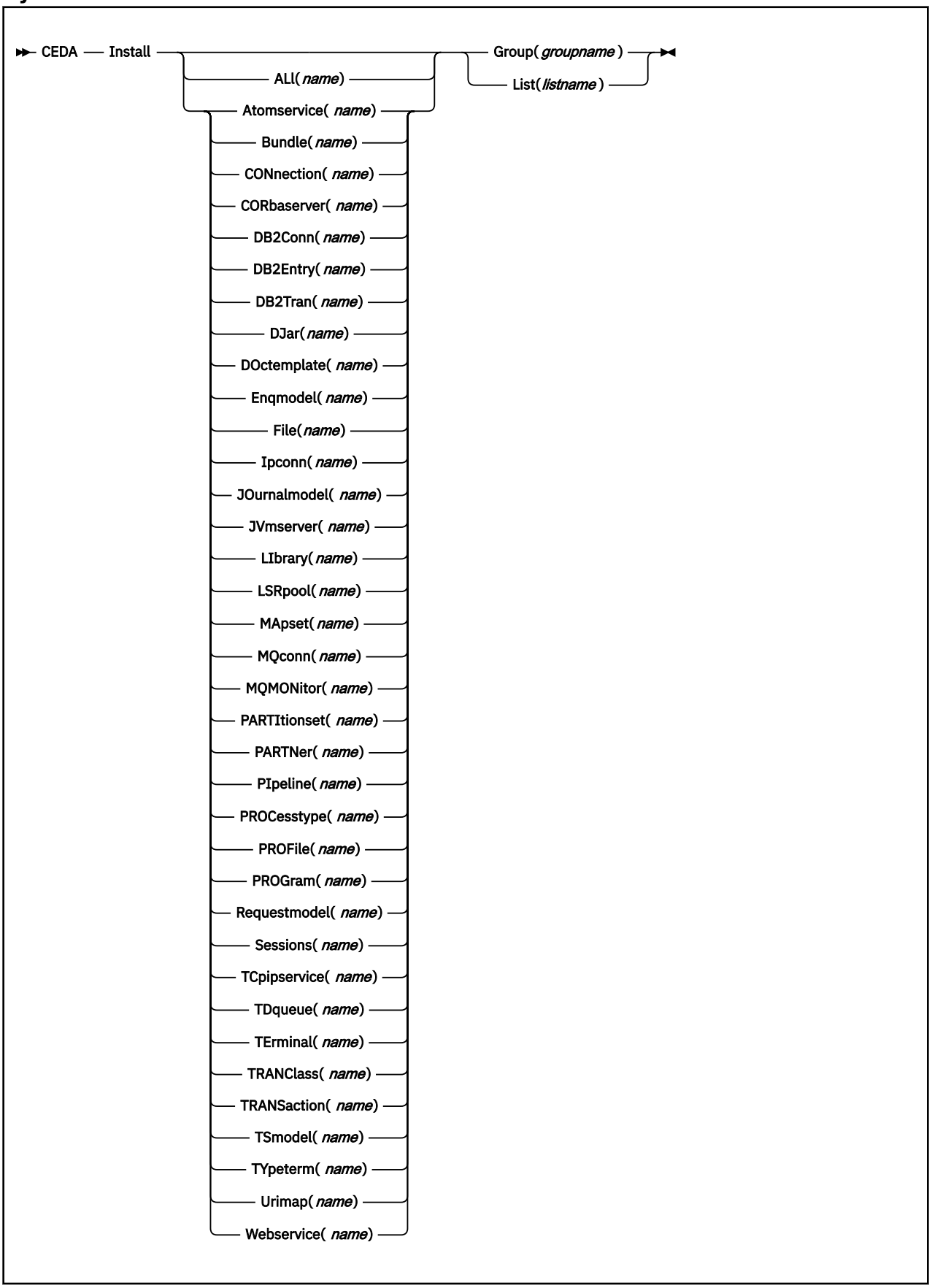
Show the groups in lists whose names begin with PROD and are five characters long:

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

The CEDA INSTALL command

Use the **CEDA INSTALL** command to make the resource definitions in the named group or group list available to the active CICS system.

Syntax


CEDA
Install
ALl(name)
Atomservice(name)
Bundle(name)
CONnection(name)
CORbaserver(name)
DB2Conn(name)
DB2Entry(name)
DB2Tran(name)
DJar(name)
DOctemplate(name)
Enqmodel(name)
File(name)
Ipconn(name)
JOurnalmodel(name)
JVmserver(name)
LIbrary(name)
LSRpool(name)
MApset(name)
MQconn(name)
MQMONitor(name)
PARTItionset(name)
PARTNer(name)
PIpeline(name)
PROCesstype(name)
PROFile(name)
PROGram(name)
Requestmodel(name)
Sessions(name)
TCpipservice(name)
TDqueue(name)
TErminal(name)
TRANClass(name)
TRANSaction(name)
TSmodel(name)
TYpeterm(name)
Urimap(name)
Webservice(name)
Group(groupname)
List(listname)

Description

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

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

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

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

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

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

If you want to use single-resource install for ENQMODEL, note that ENQMODEL definitions usually have the default status of ENABLED, so the order of installing ENQMODELs must follow the rules for enabling ENQMODEL definitions. This means that the ENQMODEL definitions that form nested generic ENQnames must be enabled in sequence, from the most specific to the least specific, for example: ABCD* then ABC* then AB*.

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

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

If the installation of one or more of the resources in the group or groups that are being installed fails because the resource is in use or for any other reason, the following actions occur:

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

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

You can run the CEDA INSTALL from a console, using the MVS MODIFY command.

Options

Group(*groupname*)
Specifies the name of the group to be installed, or the name of the group that contains the resource to be installed. A generic group name is not accepted.

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

***resource*(*name*)**
Specifies the type and name of the resource that you want to install.

The CEDA LOCK command

Use the **CEDA LOCK** command to restrict update and delete access to a single operator identifier.

Syntax

Description

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

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

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

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

A generic group or list name is not accepted.

Locking and unlocking resources

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

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

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

- CHECK
- COPY
- DISPLAY
- INSTALL
- VIEW

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

Options

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

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

Examples

- To lock a list L1:

```
LOCK LIST(L1)
```

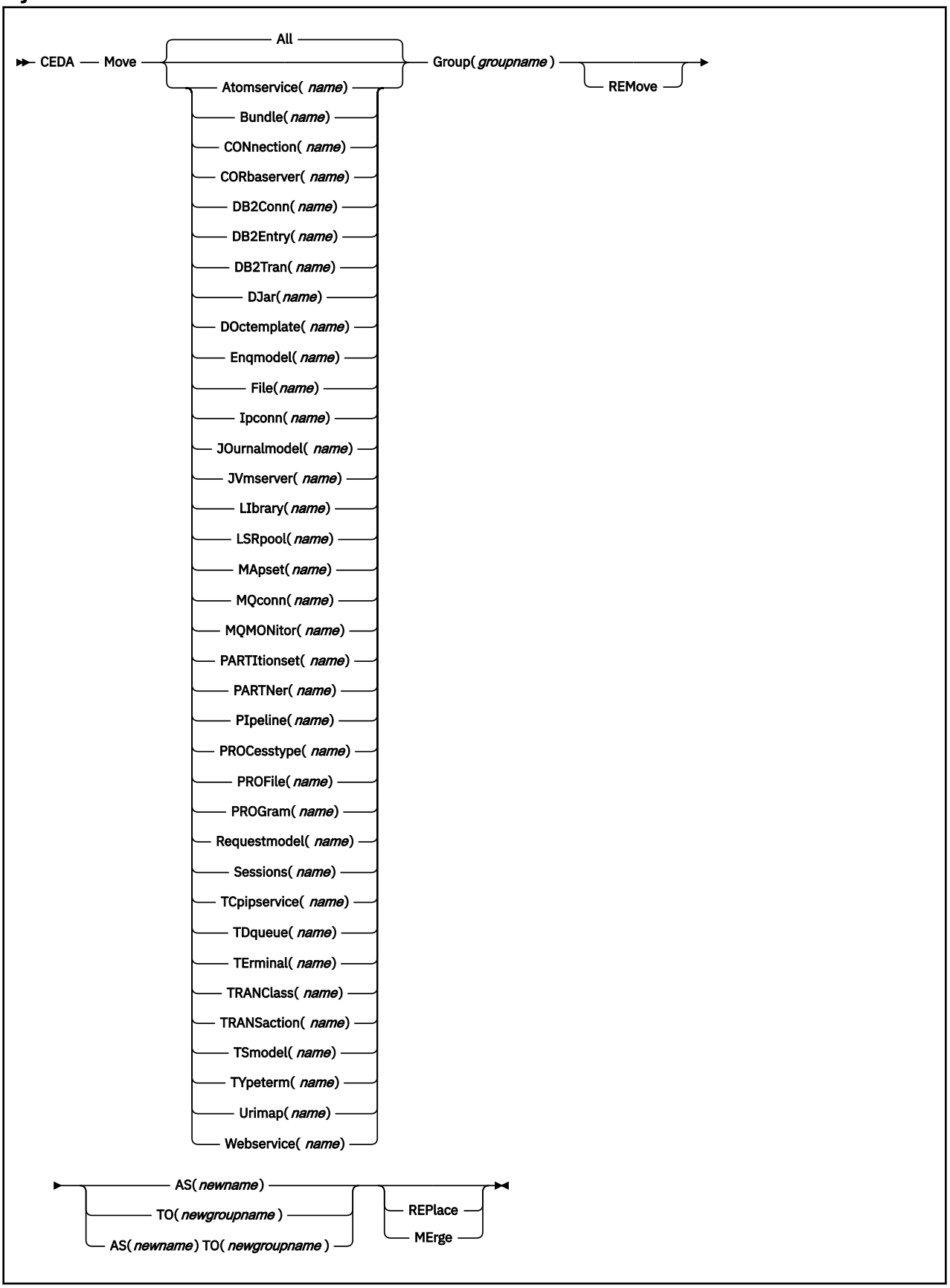
- To lock a group G1:

```
LOCK GROUP(G1)
```

The CEDA MOVE command

Use the **CEDA MOVE** command to move one or more resource definitions from one group to another.

Syntax


CEDA
Move
All
Atomservice(name)
Bundle(name)
CONnection(name)
CORbaserver(name)
DB2Conn(name)
DB2Entry(name)
DB2Tran(name)
DJar(name)
DOctemplate(name)
Enqmodel(name)
File(name)
Ipconn(name)
JOurnalmodel(name)
JVmserver(name)
LIbrary(name)
LSRpool(name)
MApset(name)
MQconn(name)
MQMONitor(name)
PARTItionset(name)
PARTNer(name)
PIpeline(name)
PROCesstype(name)
PROFile(name)
PROGram(name)
Requestmodel(name)
Sessions(name)
TCpipservice(name)
TDqueue(name)
TErminal(name)
TRANClass(name)
TRANSaction(name)
TSmodel(name)
TYpeterm(name)
Urimap(name)
Webservice(name)
Group(groupname)
REMove
AS(newname)
TO(newgroupname)
AS(newname) TO(newgroupname)
REPlace
MErge

Description

This command moves the resource definitions from the group named by the GROUP option to the group named by the TO option.

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

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

Options

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

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

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

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

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

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

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

Examples

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

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

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

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

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

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

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

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

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

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

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

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

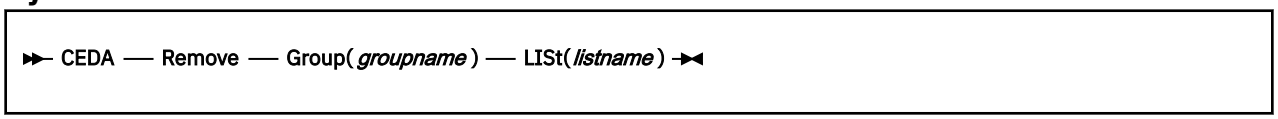
- To combine definitions from group X in with definitions in group Y, keeping the X version of any duplicates:

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

The CEDA REMOVE command

Use the **CEDA REMOVE** command to remove the name of a group from a list.

Syntax



Description

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

A generic list name is not accepted.

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

When a group is deleted, the user can request the group to be removed from all lists that had contained it. When the last group is removed from a list, the list is deleted.

Options

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

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

Examples

- A list LL02 contains the following groups:

  G001
  X001
  XG001
  G002
  G003
  X002
  G004

  To remove all groups beginning with G:

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

  This leaves:

  X001
  XG001

X002

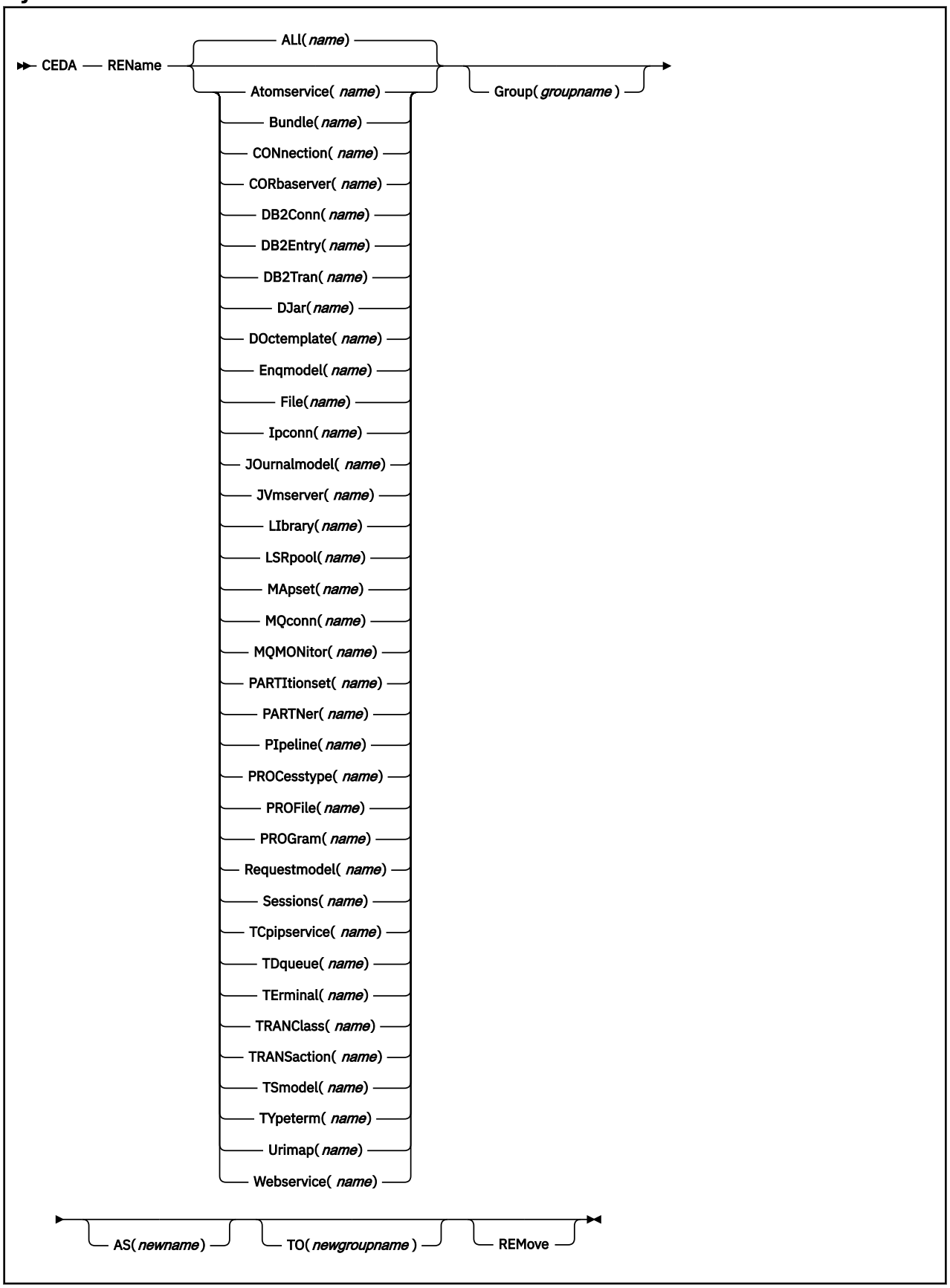
- To remove the list completely:

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

The CEDA RENAME command

Use the **CEDA RENAME** command to change the name of a resource on the CSD or move it to a different group.

Syntax


CEDA
REName
ALl(name)
Atomservice(name)
Bundle(name)
CONnection(name)
CORbaserver(name)
DB2Conn(name)
DB2Entry(name)
DB2Tran(name)
DJar(name)
DOctemplate(name)
Enqmodel(name)
File(name)
Ipconn(name)
JOurnalmodel(name)
JVmserver(name)
LIbrary(name)
LSRpool(name)
MApset(name)
MQconn(name)
MQMONitor(name)
PARTItionset(name)
PARTNer(name)
PIpeline(name)
PROCesstype(name)
PROFile(name)
PROGram(name)
Requestmodel(name)
Sessions(name)
TCpipservice(name)
TDqueue(name)
TErminal(name)
TRANClass(name)
TRANSaction(name)
TSmodel(name)
TYpeterm(name)
Urimap(name)
Webservice(name)
Group(groupname)
AS(newname)
TO(newgroupname)
REMove

Description

You can also rename a resource definition by using the DISPLAY command or the EXPAND command with the RENAME option. See "The CEDA DISPLAY command" on page 293 and "The CEDA EXPAND command" on page 297 for information about these commands.

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

Options

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

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

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

***resource*(*name*)**
specifies the type and name of the resource you want to move or rename. The default is ALL, which copies all the resource definitions in a group to another group. A generic resource definition name is not accepted.

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

Examples

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

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

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

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

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

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

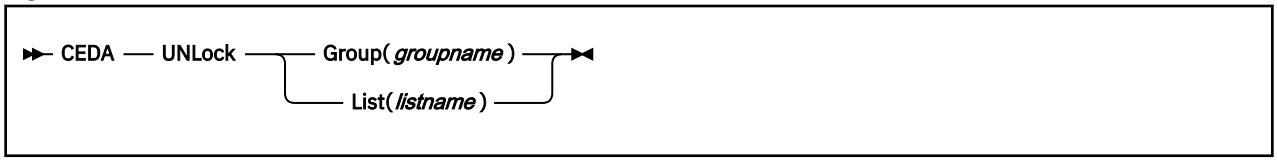
- You can move all the resource definitions of the same name from one group to another, using the TO option, at the same time as renaming them. For example:

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

The CEDA UNLOCK command

Use the **CEDA UNLOCK** command to remove the lock from a group or a list of definitions.

Syntax



Description

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

You can UNLOCK a nonexistent group or list.

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

Locking and unlocking resources

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

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

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

- CHECK
- COPY
- DISPLAY
- INSTALL
- VIEW

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

Options

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

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

Examples

- To unlock a group G1:

```
UNLOCK GROUP(G1)
```

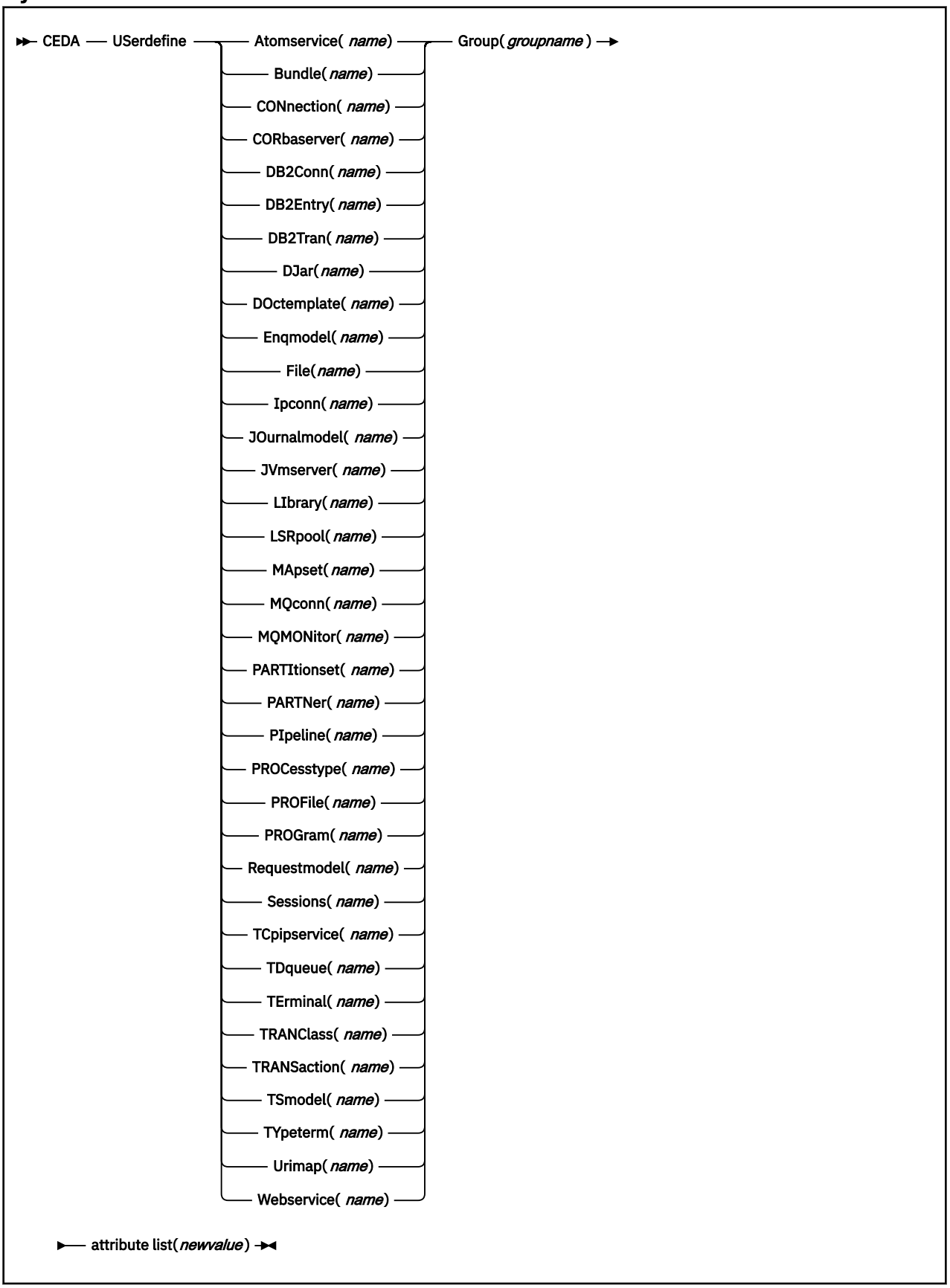
- To unlock a LIST L1:

```
UNLOCK LIST(L1)
```

The CEDA USERDEFINE command

Use the **CEDA USERDEFINE** command to create a new resource definition on the CSD with user-defined default attributes.

Syntax


CEDA
USerdefine
Atomservice(name)
Bundle(name)
CONnection(name)
CORbaserver(name)
DB2Conn(name)
DB2Entry(name)
DB2Tran(name)
DJar(name)
DOctemplate(name)
Enqmodel(name)
File(name)
Ipconn(name)
JOurnalmodel(name)
JVmserver(name)
LIbrary(name)
LSRpool(name)
MApset(name)
MQconn(name)
MQMONitor(name)
PARTItionset(name)
PARTNer(name)
PIpeline(name)
PROCesstype(name)
PROFile(name)
PROGram(name)
Requestmodel(name)
Sessions(name)
TCpipservice(name)
TDqueue(name)
TErminal(name)
TRANClass(name)
TRANSaction(name)
TSmodel(name)
TYpeterm(name)
Urimap(name)
Webservice(name)
Group(groupname)
attribute list(newvalue)

Description

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

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

Do this for each type of resource for which you want to set default values. Although every definition in the group has the same name (USER), each is unique because it defines a different resource type.

Options

Attribute list
The attribute list depends on the resource type being defined; some resources have attributes that must be included in the definition. Attributes that you do not specify are given default values.

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

Examples

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

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

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

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

```
  DEFINE PROGRAM(USER)
GROUP(USERDEF)            CICS RELEASE = 0620
  OVERTYPE TO MODIFY
   CEDA  DEFine
    PROGram        : USER
    Group          : USERDEF
    DEscription  ==>
    Language     ==>                    CObol | Assembler | Le370 | C | Pli
    RELoad       ==> No                 No | Yes
    RESident     ==> No                 No | Yes
    USAge        ==> Normal             Normal | Transient
    USElpacopy   ==> No                 No | Yes
    Status       ==> Enabled            Enabled | Disabled
    RSl            : 00                 0-24 | Public
    Cedf         ==> Yes                Yes | No
    DAtalocation ==> Any                Below | Any
 I New group USERDEF created
                                                SYSID=ABCD    APPLID=DBDCCICS
   DEFINE SUCCESSFUL                      TIME:  11.24.39   DATE:  97.359
 PF 1 HELP 2 COM 3 END           6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

2. Type in ASSEMBLER as the LANGUAGE option and press ENTER:

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



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

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

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

The following figure shows the panel resulting from this command.

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

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

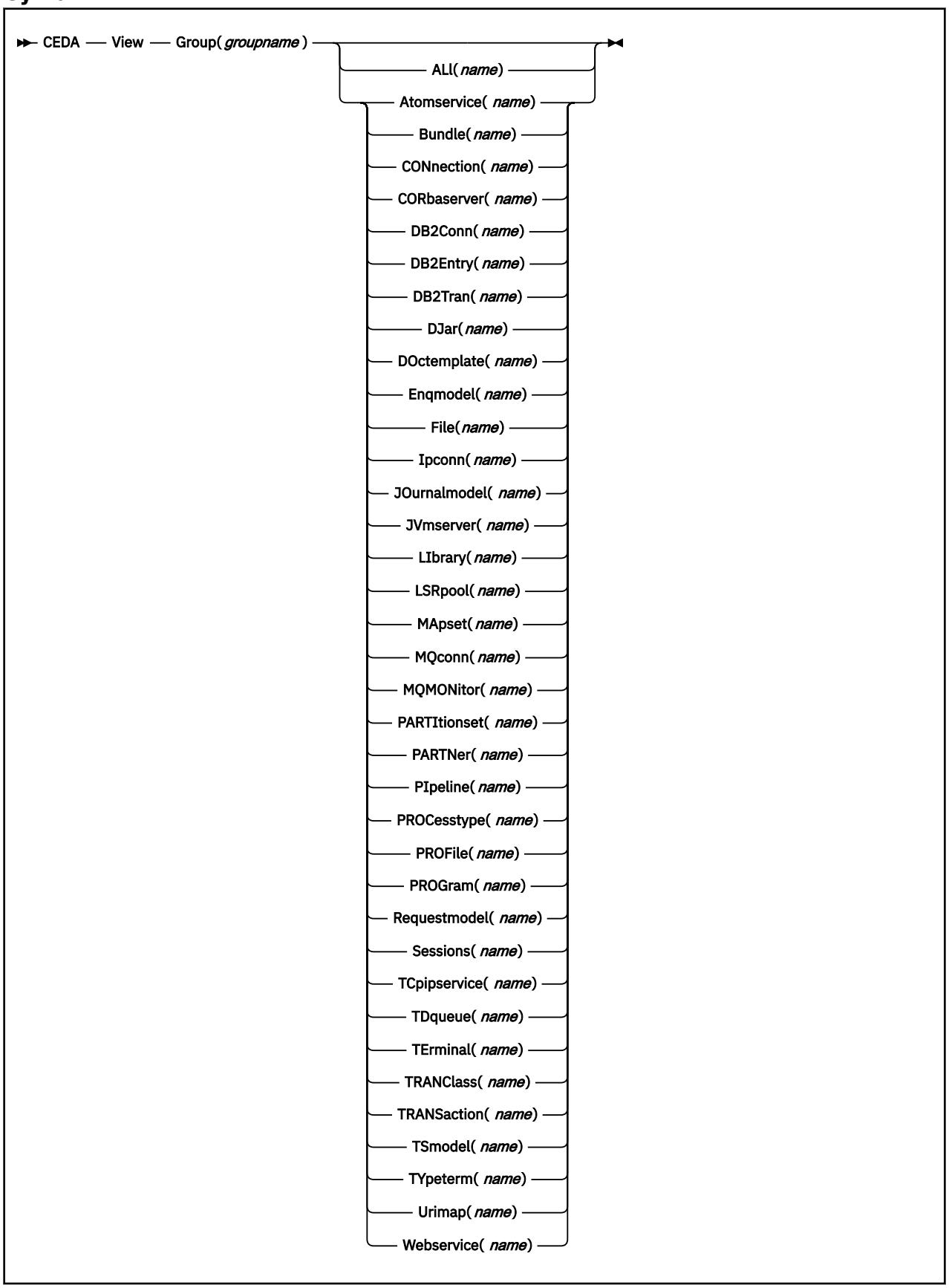
After you have set up your own defaults in a USER resource definition, anyone using the USERDEFINE command for that resource type gets those default values.

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

The CEDA VIEW command

Use the **CEDA VIEW** command to view the attributes of a resource definition on the CSD.

Syntax


CEDA
View
Group(groupname)
ALl(name)
Atomservice(name)
Bundle(name)
CONnection(name)
CORbaserver(name)
DB2Conn(name)
DB2Entry(name)
DB2Tran(name)
DJar(name)
DOctemplate(name)
Enqmodel(name)
File(name)
Ipconn(name)
JOurnalmodel(name)
JVmserver(name)
LIbrary(name)
LSRpool(name)
MApset(name)
MQconn(name)
MQMONitor(name)
PARTItionset(name)
PARTNer(name)
PIpeline(name)
PROCesstype(name)
PROFile(name)
PROGram(name)
Requestmodel(name)
Sessions(name)
TCpipservice(name)
TDqueue(name)
TErminal(name)
TRANClass(name)
TRANSaction(name)
TSmodel(name)
TYpeterm(name)
Urimap(name)
Webservice(name)

Description

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

Options

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

Examples

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

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

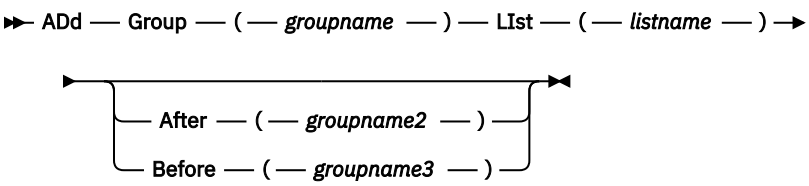
Chapter 3. Resource management utility DFHCSDUP commands

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

The DFHCSDUP ADD command

Add a group to a list.

ADD syntax



Options

After*(groupname2)*
specify AFTER to place the new group name after the existing group name. The group name is added at the end of the list if BEFORE or AFTER is not specified.

Before*(groupname3)*
specify BEFORE to place the new group name before the existing group name. The group name is added at the end of the list if BEFORE or AFTER is not specified.

Group*(groupname)*
specifies the name of the group to be added. The name must not already exist in the list. A generic group name is not accepted.

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

Examples

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

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

To add another group to list LA01, withot specifying where

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

LA01 now looks like this

- GA001
- GA002

To add another group at the beginning of the list

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

To add another group between GA001 and GA002

© Copyright IBM Corp. 1974, 2023

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

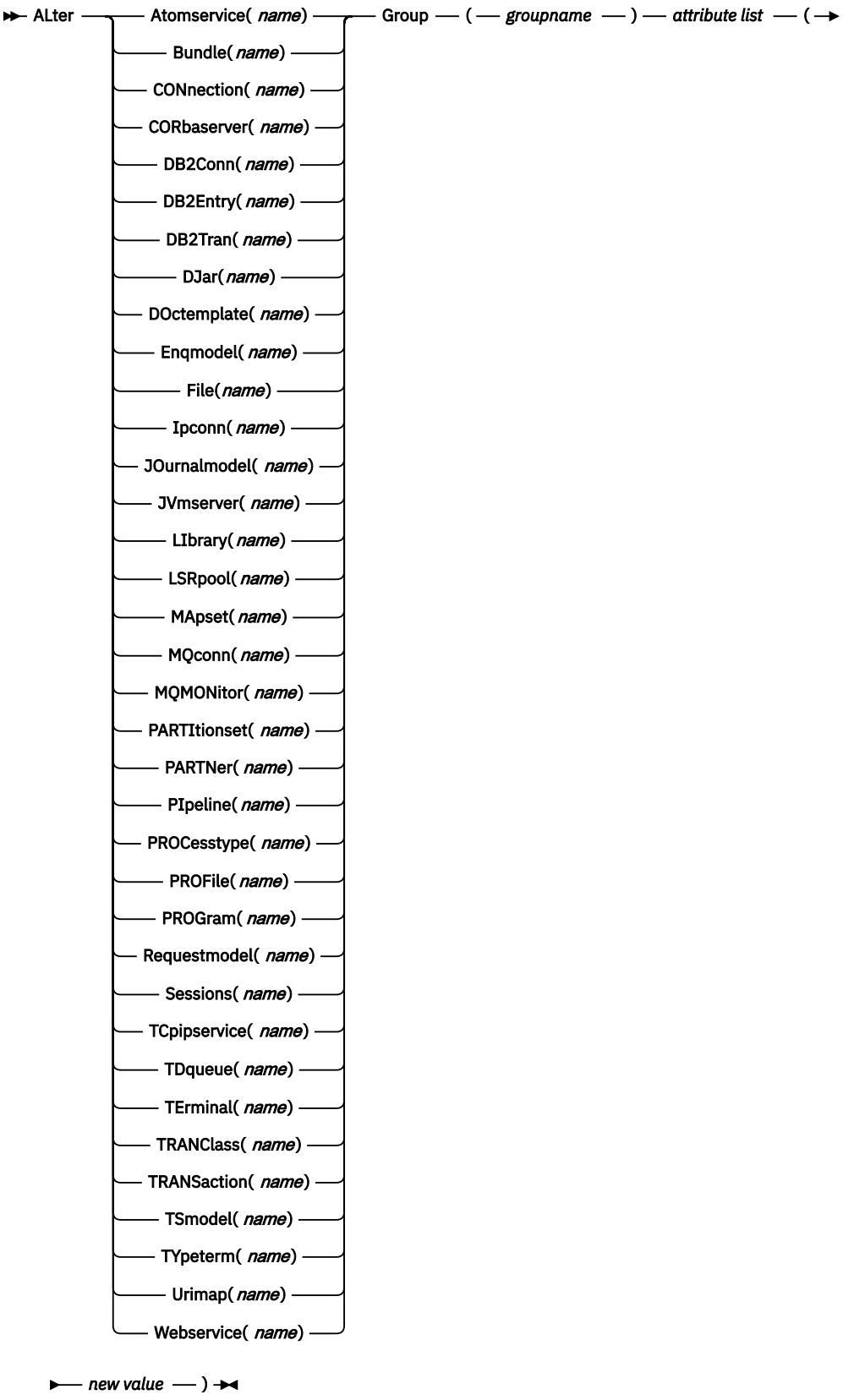
LA01 now looks like this

- GA003
- GA001
- GA004
- GA002

The DFHCSDUP ALTER command

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

ALTER syntax



Description

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

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

You can specify null attribute values, for example:

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

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

- The command:

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

  uses the default value of NO for RLSACCCESS and the description is blanked out.
- The command:

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

  uses the default value DFHCICSA for the PROFILE field.

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

Generic naming in the ALTER command

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

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

Options

Attribute list
specifies the attributes to be altered.

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

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

Examples

To make a program resident:

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

To make all programs in the group GENMOD resident:

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

The DFHCSDUP APPEND command

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

APPEND syntax

Description

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

Note: If you are appending from one CSD to another, you should be aware that this command does not copy the groups themselves; you should use a separate COPY command to do this.

Options

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

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

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

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

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

Examples

A list called LISTA contains the following groups:

- GB001
- GB002
- GB003

A list called LISTB contains the following groups:

- G001
- G002
- G003

Append LISTB to LISTA, like this:

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

After this, LISTA contains the following groups, in this order:

- GB001
- GB002

- GB003
- G001
- G002
- G003

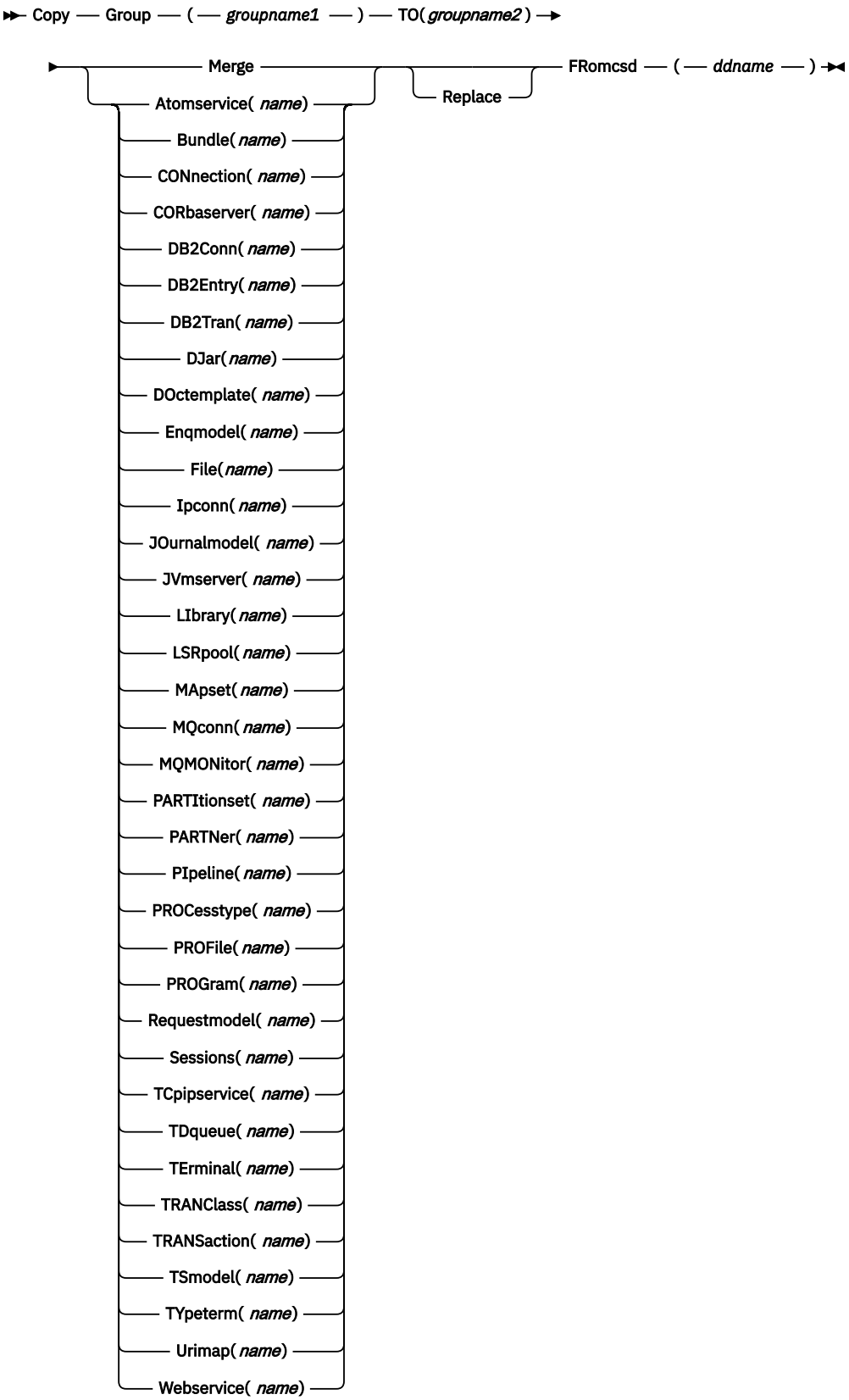
and LISTB still contains:

- G001
- G002
- G003

The DFHCSDUP COPY command

Copy resource definitions from one group to another group.

COPY syntax



Description

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

Using the DFHCSDUP COPY command, you can copy individual resource definitions, or all the resource definitions in the group. The resource definitions are copied to the group named on the TO parameter (*groupname2*) in the primary file. If this group already exists, the definitions from the source group (*groupname1*) are added to those already in the *groupname2* group. If the group specified on the TO parameter does not already exist, a new group of that name is created.

When you use the DFHCSDUP COPY command to copy all the resource definitions in a group into another group, and the groups contain duplicate resource names, you must specify either MERGE or REPLACE to indicate how duplicates should be handled. If you specify MERGE, duplicate definitions in the TO group are not replaced. If you specify REPLACE, the definitions being copied replace those in the TO group. If you do not specify MERGE or REPLACE, and duplicates are found, the copy operation fails.

When you use the DFHCSDUP COPY command to copy an individual resource definition into another group, by default, if a duplicate definition exists in the TO group it is not replaced. If you specify REPLACE, the definition being copied replaces that in the TO group.

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

Generic naming in the COPY command

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

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

You can use the asterisk (*) symbol to copy from generically named groups to other generically named groups or from generically named groups to a specific group, as shown in “Examples” on page 331.

The DFHCSDUP COPY command does not accept generic names for individual resource definitions that you specify using one of the resource options.

Options

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

Group*(groupname1)*
Specifies the name of the group to be copied. You can specify a generic name for the group by using an asterisk (*). The default action is to copy the whole group. Alternatively, you can specify one of the resource type options and name an individual resource definition of that type to be copied.

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

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

***resource*(*name*)**
Specifies the type and name of an individual resource definition whose attributes you want to copy. On the DFHCSDUP COPY command, you cannot use generic names for individual resource definitions.

To*(groupname2)*

Specifies the name of the group to which all the resource definitions, or the individual named resource definition, are copied. If you are copying from another CSD file, you can give this group the same name as the one you are copying from. You can specify a generic name for the group by using an asterisk (*).

Examples

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

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

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

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

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

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

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

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

The following example copies a single resource definition from the CICSC1 group to the CICSC2 group:

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

The DFHCSDUP DEFINE command

Create new resource definitions.

DEFINE syntax

Options

Attribute list
The attribute list depends on the resource type being defined; some resources have attributes that must be included in the definition. Attributes that you do not specify are given default values.

Group*(groupname)*
Specifies the name of the group that contains the resource definition that is created. Do not use a generic group name. If you specify the name of a group that does not already exist, the group is created.

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

Examples

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

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

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

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

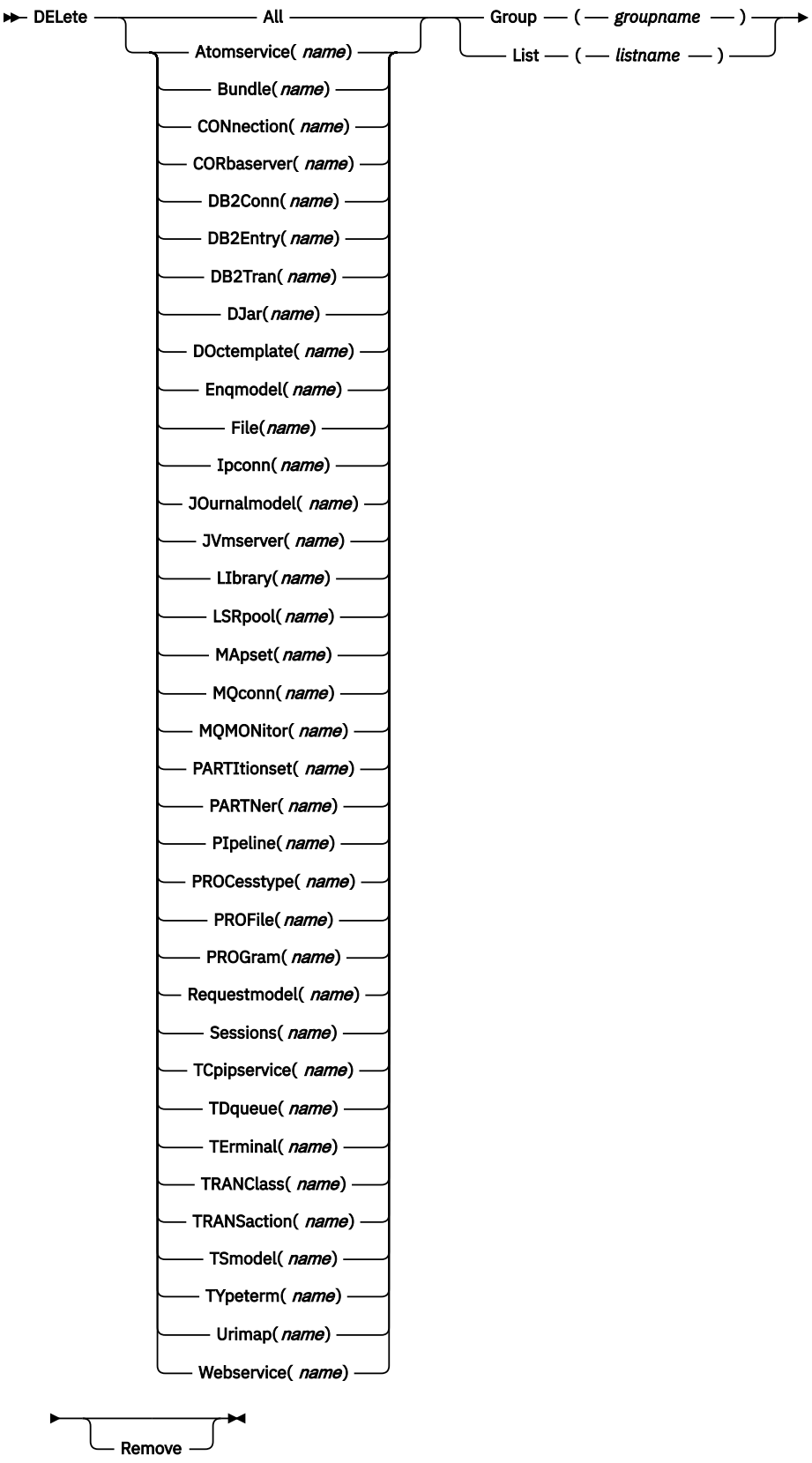
```
DEFINE TERMINAL(CON0)     GROUP(CONTERMS)
       CONSNAME(CONSJCL)  TYPETERM(DFHCONS)
       DESCRIPTION(MVS CONSOLE FOR ISSUING JCL COMMANDS)

DEFINE TERMINAL(CON1)     GROUP(CONTERMS)
       CONSNAME(CONSMAS)  TYPETERM(DFHCONS)
       DESCRIPTION(MVS MAIN CONSOLE)
```

The INITIALIZE command generates a TYPETERM definition, but not a TERMINAL definition, for a console. You must have at least one console defined to issue MVS **MODIFY** commands to CICS.

The DFHCSDUP DELETE command

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

DELETE syntax



Description

Deleting a resource definition is different from removing a group from a list (see "The DFHCSDUP REMOVE command" on page 341). A deleted resource definition really does disappear from the CSD file.

Note:

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

If a group is deleted, the group is not removed from all lists that contain it.

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

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

Options

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

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

Remove
If this is specified when the group is deleted, the group is removed from all lists that contained it unless UPGRADE commands are running.

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

This operand can be used only with the GROUP option.

Examples

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

- GB001
- GB002

Group GB001 contains the following resource definitions:

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

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

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

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

```
DELETE GROUP(GB002)
```

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

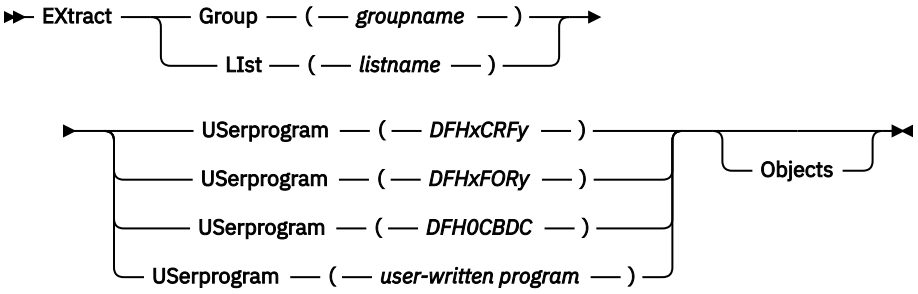
```
DELETE LIST(LISTA)
```

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

The DFHCSDUP EXTRACT command

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

EXTRACT syntax



Description

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

Note: For programming information about coding user programs for the EXTRACT command, see Invoking a user program from DFHCSDUP.

Options

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

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

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

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

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

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

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

Each user-written program must use a specific DD card:

- DFHxCRFy needs DD card CRFOUT
- DFHxFORy needs DD card FOROUT
- DFH0CBDC needs DD card CBDOUT

All other user programs are available in source form, in CICSTS54.CICS.SDFHSAMP, and the assembler versions are also available in pregenerated form in CICSTS54.CICS.SDFHLOAD.

Examples

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

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

The DFHCSDUP INITIALIZE command

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

INITIALIZE syntax

Description

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

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

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

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

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

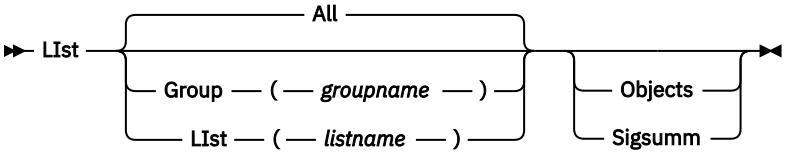
INITIALIZE also creates a control record at the start of the CSD file. This record contains fields identifying the CICS release and the current level of service applied to the CSD. It also has fields containing the date and time of creation of the CSD file, and the date and time the file was last updated.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

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

The DFHCSDUP LIST command

Produces listings of the current status of the CSD file.

LIST syntax



Description

The listings are sent to the SYSOUT data set, with the messages issued by the command processing. The contents of all the qualifying groups or lists are printed.

The listings contain output to help you confirm the release information for the CSD. The heading *CREATED BY RELEASE:* shows the release in which the CSD was created. If present, it may be followed by *UPGRADED TO RELEASE:* if the CSD has been upgraded, and, to what release.

If, maintenance has been applied to the CSD, then an additional heading appears; *LAST CSD MAINTENANCE UPGRADE USED filename AT PTF ptfnumber LEVEL*. For example, maintenance applied using a *DFHCSDUP UPGRADE USING (DFHCURDM)* command will indicate the PTF level of the *DFHCURDM* used.

Options

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

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

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

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

Sigsumm
Shows the definition signature for each of the resource definitions displayed.

Examples

The listings produced by the various commands are as follows:

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

  The LIST ALL command prints summaries of all the definitions of lists and groups that are on the CSD file.
- LIST ALL OBJECTS
  - Names of defined lists and groups
  - Summary of lists
  - Summary of groups
  - Objects in groups

  The LIST ALL OBJECTS command prints summaries of all the definitions of lists and groups that are on the CSD file, with the properties of the resources in all the groups.
- LIST GROUP(*groupname*)
  - Summary of groups

  - Group name can be generic

  The LIST GROUP command summarizes the names of all the resources in one or more groups. They are organized in each group into resource type categories; for example: map sets, and programs.

- LIST GROUP(*groupname*) OBJECTS
  - Summary of groups
  - Objects in groups
  - Group name can be generic

  With this command, you can tabulate the properties of the resources, organized according to resource type. The creation time for each resource is given, with all its attributes, as originally set up by using DEFINE and ALTER commands or by migrating it from a CICS table. The properties of transactions and profiles are arranged in the same subcategories that appear on the CEDA DEFINE screen.

- LIST GROUP(*groupname*) SIGSUMM
  - Group name can be generic

  Use this command to tabulate the definition signature of the resources, organized according to resource type.

- LIST LIST(*listname*)
  - Summary of lists
  - List name can be generic

  The contents of one or more group lists are tabulated. The groups are displayed in the same sequence as their position in the list. This order is set by the commands ADD and APPEND, which were used in the CEDA transaction to build the list.

- LIST LIST(*listname*) OBJECTS
  - Summary of lists
  - Objects of groups in list
  - Generic list name is not allowed

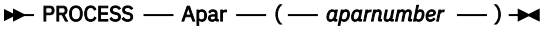
  Use this command to tabulate the properties of all the resources to be defined in a CICS system at startup time. They are identified by one or more list names specified in the GRPLIST=(*list1,list2,list3,list4*) system initialization parameter. The names of all the groups in the list appear in the summary of lists. Then, for each group that is contained in the list, the properties of the individual resources in the group are tabulated.

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

The DFHCSDUP PROCESS command

Apply maintenance to the CSD file for a specific APAR.

PROCESS syntax



Description

The PROCESS APAR command is used to apply maintenance to your CSD file for a specific APAR. Only use this command in accordance with the instructions in the associated PTF cover letter.

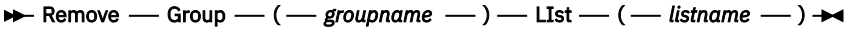
Options

Apar*(aparnumber)*
The number of the APAR providing the maintenance; for example, PROCESS APAR(PQ12417) is used to apply maintenance for APAR PQ12417.

The DFHCSDUP REMOVE command

Remove a group name from a list.

REMOVE syntax



Description

The group, and all its resource definitions, still exists on the CSD file.

Options

Group*(groupname)*
specifies the name of the group to be removed. Do not use a generic group name.

LIst*(listname)*
specifies the name of the list from which a group is to be removed. Do not use a generic list name. When the last group is removed from a list, the list no longer exists on the CSD file.

Examples

A list LL02 contains the following groups:

G001 G002 G003 G004

To remove group G003:

```
REMOVE GROUP(G003) LIST(LL02)
```

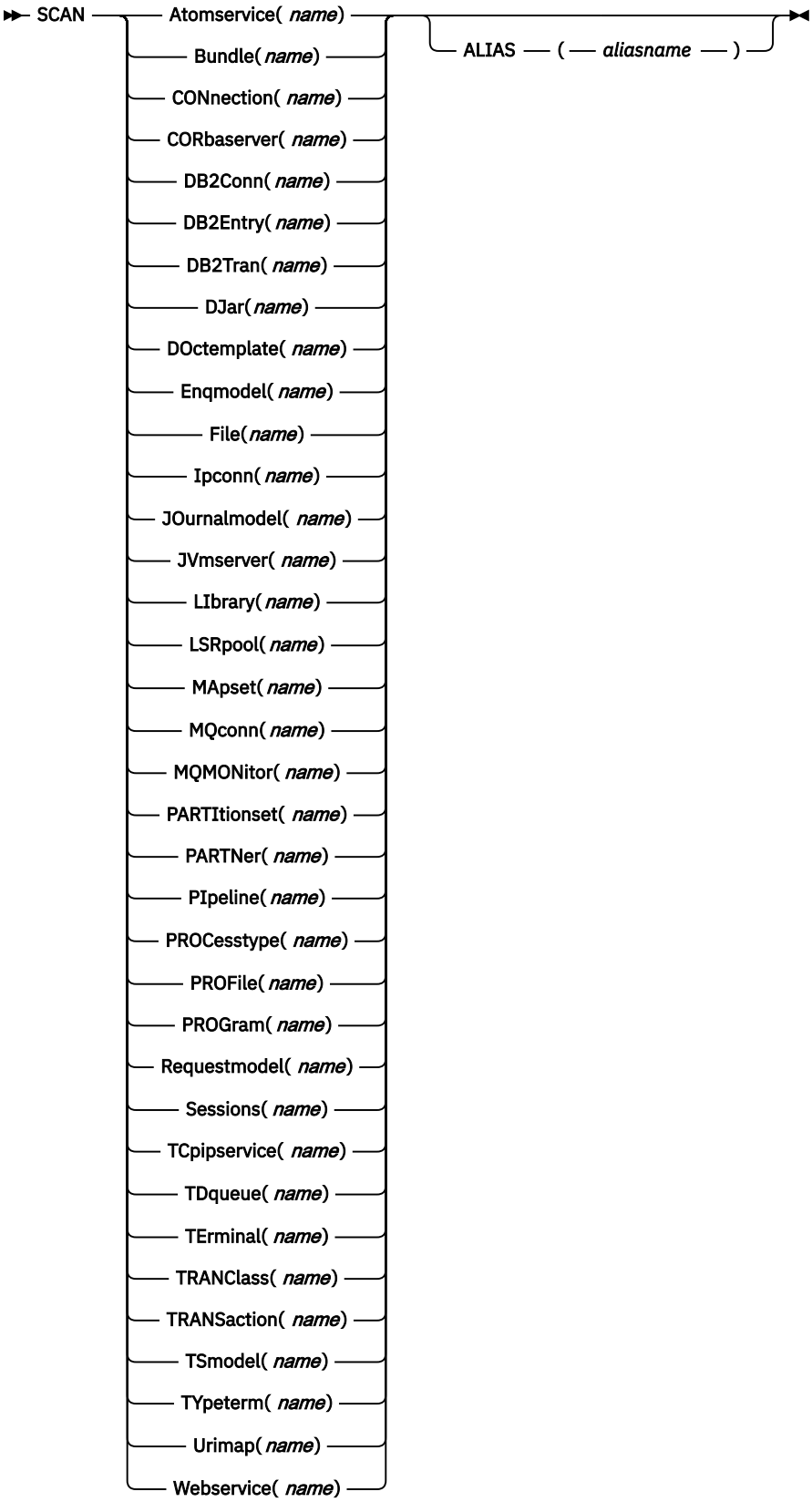
This leaves:

G001 G002 G004

The DFHCSDUP SCAN command

SCAN all the IBM-supplied groups and user-defined groups for a specified resource. The definition of the matched resource in an IBM supplied group is compared with the definition(s) of the corresponding matched resource in the user groups.

SCAN syntax


SCAN
Atomservice(name)
Bundle(name)
CONnection(name)
CORbaserver(name)
DB2Conn(name)
DB2Entry(name)
DB2Tran(name)
DJar(name)
DOctemplate(name)
Enqmodel(name)
File(name)
Ipconn(name)
JOurnalmodel(name)
JVmserver(name)
LIbrary(name)
LSRpool(name)
MApset(name)
MQconn(name)
MQMONitor(name)
PARTItionset(name)
PARTNer(name)
PIpeline(name)
PROCesstype(name)
PROFile(name)
PROGram(name)
Requestmodel(name)
Sessions(name)
TCpipservice(name)
TDqueue(name)
TErminal(name)
TRANClass(name)
TRANSaction(name)
TSmodel(name)
TYpeterm(name)
Urimap(name)
Webservice(name)
ALIAS
(
aliasname
)

Description

For information about the types of resource that you can specify on the SCAN command, and for a description of the attributes and default values of each resource type, see RDO resources.

The SCAN command searches all the IBM supplied groups in the CSD for a resource definition of a specified name and type. A message is issued with the results of the search. The user-defined groups are then searched for the same resource definition. The outcome of this can be one of the following:

- If an IBM-supplied group and one or more user-defined groups contain the resource definition, a comparison is made between the definition in the IBM-supplied group and the user group(s). A message is issued indicating whether the definition in the IBM supplied group matches the definition(s) in the user group(s).
- If the resource definition is not found in the user defined groups a message is issued.
- If the resource definition is not found in an IBM-supplied group but is found in one or more user defined groups a message is issued indicating the group(s) that contained it.

If *aliasname* is specified, the user groups are searched using *aliasname*.

Note:

1. The compatibility groups DFHCOMPx are not scanned as part of the IBM supplied groups but as user defined groups.
2. The DESCRIPTION attribute is not used in the comparison.

You can use the SCAN command to check for differences between IBM-supplied definitions that you have modified and the latest IBM-supplied versions after an upgrade.

Options

Alias*(aliasname)*
specifies the alias name of the resource type to be searched for in the user-defined groups.

This operand is optional.

Resource*(name)*
specifies the name of the resource type to be searched for in the IBM-supplied groups, and in the user-defined groups if *aliasname* is not specified. The resource option must always be the first operand of the SCAN command.

Examples

To search the CSD for transaction CEDA:

```
SCAN TRANSACTION(CEDA)
```

The result of this could look like:

```
DFH5130 I PRIMARY CSD OPENED;  DDNAME: DFHCSD
DFH5633 I TRANSACTION CEDA FOUND IN GROUP DFHSPI
DFH5631 I TRANSACTION CEDA FOUND IN GROUP A1
          MATCHES THE IBM SUPPLIED DEFINITION IN GROUP DFHSPI
DFH5631 I TRANSACTION CEDA FOUND IN GROUP A2
          MATCHES THE IBM SUPPLIED DEFINITION IN GROUP DFHSPI
DFH5632 I TRANSACTION CEDA FOUND IN GROUP DFHCOMP1
          DOES NOT MATCH THE IBM SUPPLIED DEFINITION
          IN GROUP DFHSPI
DFH5101 I SCAN COMMAND EXECUTED SUCCESSFULLY.
```

To search the CSD for transaction CEDA with an alias name of AEDA:

```
SCAN TRANSACTION(CEDA) ALIAS(AEDA)
```

The result of this could look like:

```
DFH5130 I PRIMARY CSD OPENED;  DDNAME: DFHCSD
DFH5633 I TRANSACTION CEDA FOUND IN GROUP DFHSPI
DFH5631 I TRANSACTION AEDA FOUND IN GROUP A3
          MATCHES THE IBM SUPPLIED DEFINITION IN GROUP DFHSPI
DFH5101 I SCAN COMMAND EXECUTED SUCCESSFULLY.
```

The DFHCSDUP SERVICE command

Carry out maintenance to your CSD file.

SERVICE syntax

Description

You might occasionally (between CICS releases) have to apply a service routine to carry out preventive or corrective maintenance to your CSD file.

You can use the SERVICE command to create a new copy of the CSD file, from the existing CSD file. All the definitions are preserved, with the corrections (if any) applied.

Options

FRomcsd*(ddname)*
specifies the ddname of the current CSD file, which for the purposes of the command is treated as the secondary CSD file.

LEvel*(nnn)*
Associated with your CSD file is a current service level, initially set to 000 when the file was initialized. Applying the service routine causes the service level to be incremented in steps of one, from a “current level” to a “target level”.

This operand specifies the target service level to which the CSD file is to be upgraded, and must be 1 higher than the current level of FROMCSD. Specify it as a 3-character integer; for example, LEVEL(001).

The DFHCSDUP UPGRADE command

Change the CICS-supplied resource definitions in a primary CSD file.

UPGRADE syntax

Description

Upgrades the IBM-supplied definitions in the CSD. Definitions are added to, modified in, or deleted from DFH-groups. Note that deleted definitions are added to compatibility groups with names of the form DFHCOMP*n*. This enables you to share the CSD with earlier releases of CICS after you have run the upgrade command.

Use the DFHCSDUP LIST command to check the status of the CSD file before or after the upgrade. The command reports the release in which a CSD was created, and the release to which it has been upgraded.

The upgrade command can also be used to apply any package of IBM-supplied resource definitions to the CSD file. For example, the definitions for the CICS sample programs and transactions can be transferred to the CSD file with the UPGRADE statement.

Options

Replace

Specify the REPLACE option when you need to rerun the UPGRADE command (for example, because of a previous issue).

USing*(filename)*

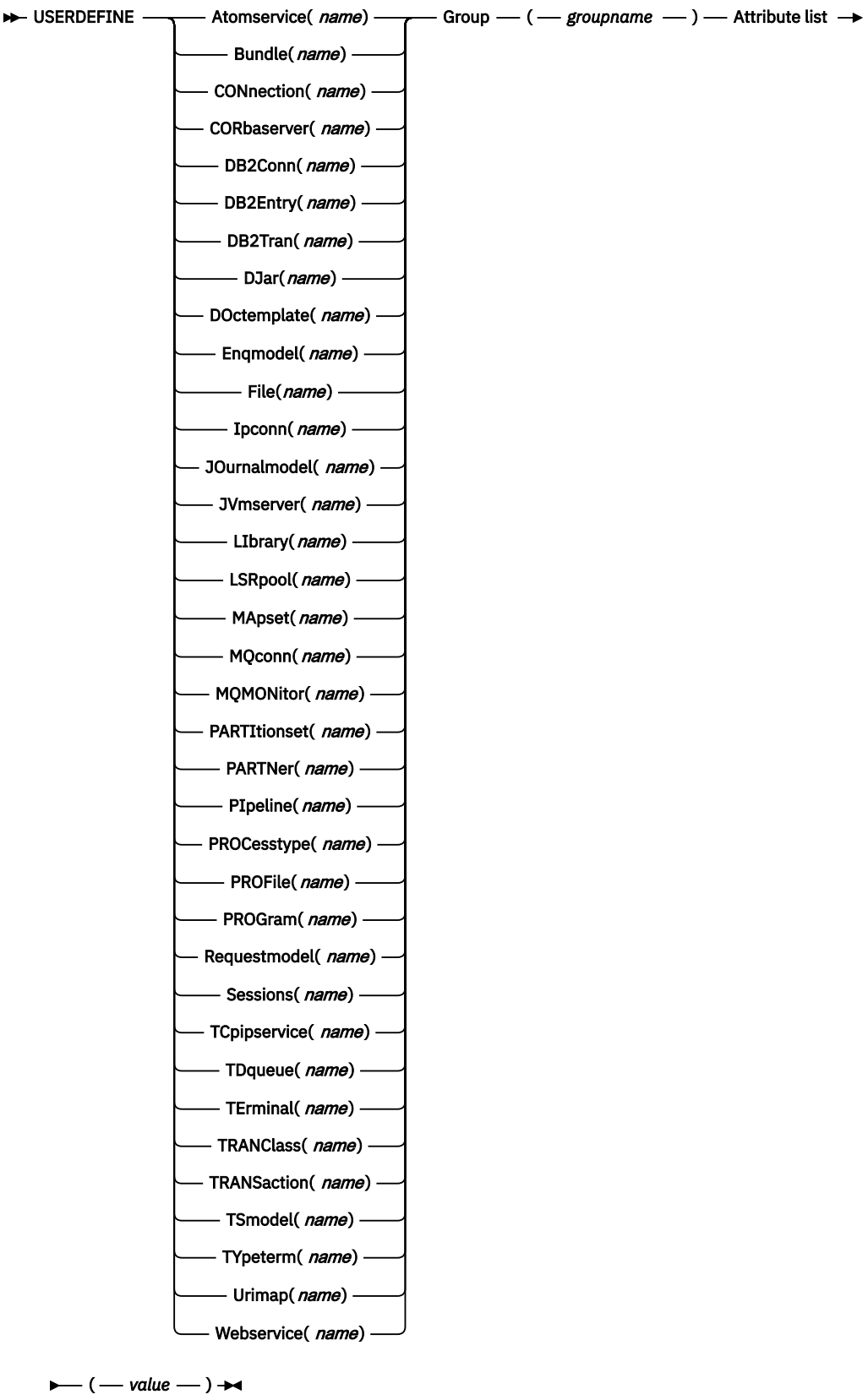
To install IBM features onto CICS, specify `UPGRADE USING(`*`filename`*`)`. For example, `UPGRADE USING(DFHRDJPN)` is used to place the double-byte character set feature definitions onto the CSD file.

To upgrade a CSD, do not specify the **USING** operand. All IBM-supplied definitions from **any** release are deleted and then the CSD file is initialized, so you do not need to say which release you came from.

The DFHCSDUP USERDEFINE command

Create new resource definitions using your own default values instead of the default values supplied by CICS.

USERDEFINE syntax



Description

The USERDEFINE command is an alternative to the DEFINE command. Instead of using the default values supplied by CICS, the USERDEFINE command uses your own default values to create a resource definition. Otherwise it operates in exactly the same way as the DEFINE command.

To set up your own default values for the USERDEFINE command, use the normal DEFINE command to create resource definitions named USER in a group named USERDEF:

- Create a resource definition named USER in the USERDEF group for each resource for which you want to provide default values. For example, if you want to provide default values for PROGRAM, TRANSACTION, and TCPIPSERVICE resource definitions, create the resource definitions PROGRAM(USER), TRANSACTION(USER), and TCPIPSERVICE(USER) in the USERDEF group. It does not matter that all the resource definitions in the USERDEF group are named USER; they are unique because they are different resource types. Any resource definitions in the USERDEF group that are not named USER are ignored by the USERDEFINE command.
- In each resource definition in the USERDEF group, specify the default values that are to be applied when you use the USERDEFINE command to create a resource of that type. For example, if you want Assembler to be the default language in PROGRAM resource definitions created with the USERDEFINE command, issue the following DEFINE command to create the resource definition:

```
DEFINE PROGRAM(USER) GROUP(USERDEF) LANGUAGE(ASSEMBLER)
```

- Each resource definition in the USERDEF group must be a complete, valid resource definition. For example, a transaction definition must name a program definition, even if you always supply a program name when you use the USERDEFINE command to define a transaction.
- You do not have to install the resource definitions in the USERDEF group.

When you have created resource definitions in the USERDEF group, you can use the USERDEFINE command to define those types of resources, and the default values that you set up are used in the resource definitions. For example, if you have created a PROGRAM resource definition in the USERDEF group that specifies LANGUAGE(ASSEMBLER), the following command creates a resource definition for program P2 in group GRP and specifies Assembler as the language:

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

Options

Attribute list*(value)*
The attribute list depends on the resource type that is being defined; some resources have attributes that must be included in the definition. For a description of the attributes and default values of each resource type, see RDO resources. Attributes that you do not specify are given default values.

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

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

The DFHCSDUP VERIFY command

Remove internal locks on groups and lists.

VERIFY syntax

Description

Use the VERIFY command only when the CSD file is not in use and no backout processing is pending on the CSD file; preferably use it only when no CICS systems that may use the CSD file are running. In particular, do not use the VERIFY command while CICS systems could be accessing the CSD file in RLS access mode.

VERIFY acts on the whole CSD file, and is for use in the extreme condition where internal lock records remain behind. These records are normally removed when a function that changes the CSD file is complete. However, this might not have happened if there was a system failure when the CEDA transaction was running, or if an offline utility failed to finish. The locks may prevent CEDA users from accessing certain groups and lists on the CSD file.

Note that VERIFY removes only the internal locks. It does not affect the normal user locks applied by the LOCK command in the CEDA transaction.

Chapter 4. DFHCSDUP: syntax and command rules

Use the following rules for the syntax and preparation of commands for the DFHCSDUP program

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

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

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

Leading blanks are ignored, and blanks between keywords and operands are permitted.

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

Blank records between commands are ignored.

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

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

An example of a valid sequence of commands is shown in Figure 6 on page 351. Other examples of commands are given in the command descriptions that follow.

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

Figure 6. Sample commands of the DFHCSDUP program

© Copyright IBM Corp. 1974, 2023

Chapter 5. Resource definition macros

Although you can define most CICS resources in a number of ways, for some resources you must use CICS macros.

CLT: command list table

The command list table (CLT) is used by the *extended recovery facility* (XRF) and contains a list of MVS system commands and messages to the operator, to be issued during takeover. Typically, the function of these commands is to tell alternate systems to take over from their active systems in the same MRO-connected configuration.

The command list table also contains the name of the alternate system, with the jobname of the active system that it is allowed to cancel. (See DFHCLT TYPE=LISTSTART FORALT operand.) This provides a security check against the wrong job being canceled, when the alternate system takes over.

In addition, the DFHCLT TYPE=INITIAL macro gives JES routing information, needed to send cancel commands to the appropriate MVS system. If you are using XRF, you **must** have a CLT: it is used only by the alternate CICS system.

For security reasons, link-edit the CLT into a library authorized using APF. For virtual storage constraint relief considerations, link-edit using a MODE control statement specifying AMODE(31),RMODE(ANY). The table should be link-edited as reentrant. The CLT is not loaded into the CICS nucleus.

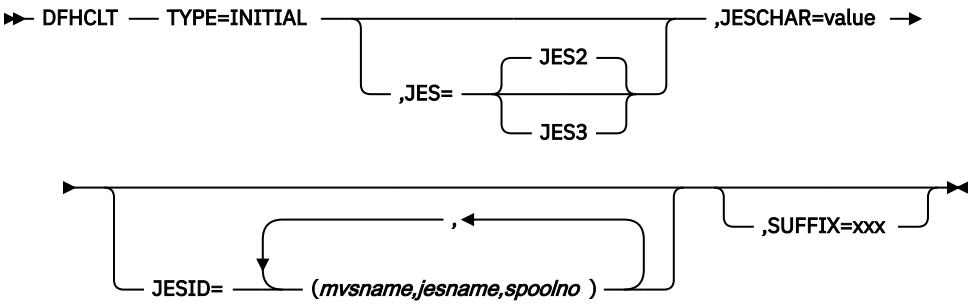
Your CLT can contain the following statements:

- DFHCLT TYPE=INITIAL
- DFHCLT TYPE=LISTSTART
- DFHCLT TYPE=COMMAND
- DFHCLT TYPE=WTO
- DFHCLT TYPE=LISTEND
- DFHCLT TYPE=FINAL (see TYPE=FINAL (end of table))

Note: Although the CLT may be shared by a number of alternate systems, take care that MVS is not given too many redundant commands during takeover. For example, in multiregion operation and using one CLT with commands for several regions, region 1 would send valid commands to other regions, but they would in turn send redundant commands to region 1 and to each other.

Control section: DFHCLT TYPE=INITIAL

The DFHCLT TYPE=INITIAL macro establishes the entry point and the beginning address of the CLT being defined.



For general information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see TYPE=INITIAL (control section).

© Copyright IBM Corp. 1974, 2023

JES={JES2vJES3}
Specifies the version of JES being used. If you have active and alternate CICS systems in different CPCs, you must use the same version of JES on both CPCs.

If you are using JES3, you need release 2.2.0 for full support of all CLT functions in a two-CPC environment. JES2 accepts commands issued by programs. For JES3, this feature was only introduced for release 2.2.0. With an earlier JES3 release, it is possible that a takeover in a two-CPC environment can only proceed after the operator has manually canceled the failing active CICS system. This should occur only when the active system does not realize that it is failing and continues to run.

JESCHAR=value
Specifies the one-character prefix to be used for commands to be passed to JES. If you omit this keyword:

- JESCHAR=$ is the default for JES=JES2
- JESCHAR=* is the default for JES=JES3

JESID=((mvsname,jesname,spoolno) [,(mvsname,jesname,spoolno),...])
Specifies the JES routing code that corresponds to the MVS name and JES name of an active CICS system. You must use this option if active and alternate CICS systems are in different CPCs.

You can specify several groups of *mvsname*, *jesname*, and *spoolno*, so that the CLT can be used to refer to more CPC/JES combinations.

mvsname
This is the SID, as specified in SYS1.PARMLIB member SMFPRM*xx*, for the CPC on which the active CICS system executes.

jesname
This is the JES2 or JES3 subsystem name for the JES under whose control the active system executes. It is defined in the MVS/ESA SCHEDULR sysgen macro and also in SYS1.PARMLIB member IEFSSN*xx*.

spoolno
For JES2, this is the multiaccess spool member number of the JES2 for the active CICS system. It is defined in the JES2 initialization parameter MASDEF SSID(*n*). For JES3, this is the processor name of the JES3 for the active CICS system. It is defined in the JES3 initialization parameter MAINPROC NAME=*name*. See z/OS JES2 Initialization and Tuning Guide or z/OS JES3 Initialization and Tuning Guide.

Specifying alternate systems: DFHCLT TYPE=LISTSTART

This macro defines the start of the set of commands and messages that the alternate CICS issues when it takes over from the active CICS. (There may be no commands or messages, but you still need a CLT, so that authorization checks can be made.)

FORALT=((applid1,jobname1)[,(applid2,jobname2),...])
Specifies pairs of alternate and active CICS systems.

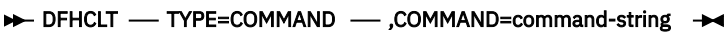
applid1
The name of the alternate CICS that issues the set of commands and messages when it takes over. This name must be the **specific APPLID**, defined in the APPLID system initialization parameter. It is used as an authorization check.

jobname1
The name of the active CICS system from which the alternate system is taking over. This name must be the **MVS JOBNAME** for the active CICS system. It is used as a security check, to ensure that the alternate system does not attempt to cancel any job other than one of that name.

You may extend this, using more pairs of *applid* and *jobname*, so that you can use one CLT for several alternate CICS systems.

Specifying takeover commands: DFHCLT TYPE=COMMAND

This macro allows you to specify the commands to be used by the alternate CICS system during takeover.



COMMAND=command-string
Defines a command that is passed to MVS for execution. CICS does not interpret this command.

The command that is issued in this way most frequently is `CEBT PERFORM TAKEOVER`.

In multiregion operation (MRO), where there is a simple hierarchy of **main** and **dependent** regions, a failing main region can issue this command to each of its dependent regions, if it is necessary that they also move to another CPC.

In a more complex multiregion operation, a failing main region can issue this to its **coordinator** region, and the coordinator can issue the same command to other main and dependent regions in the same hierarchy of regions. Hence, many MRO-connected regions can move together to another CPC, without operator intervention.

Here are some examples:

- A main region without a coordinator sends a command to a dependent region:

```
DFHCLT TYPE=COMMAND,
        COMMAND='MODIFY CICSDEP,CEBT PERFORM
        TAKEOVER'
```

- A main region sends a command to its coordinator region:

```
DFHCLT TYPE=COMMAND,
        COMMAND='MODIFY CICSCRD,CEBT PERFORM
        TAKEOVER'
```

- A coordinator region sends commands to main and dependent regions:

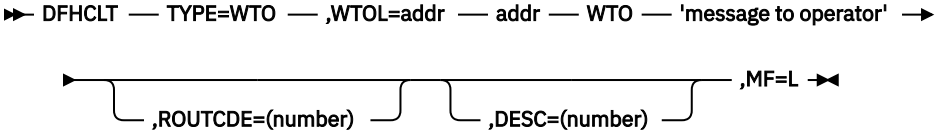
```
DFHCLT TYPE=COMMAND,
        COMMAND='MODIFY CICSMAS,CEBT PERFORM
        TAKEOVER'
DFHCLT TYPE=COMMAND,
        COMMAND='MODIFY CICSDEP,CEBT PERFORM
        TAKEOVER'
```

- You can also issue other commands to any other job running under MVS:

```
DFHCLT TYPE=COMMAND,
        COMMAND='MODIFY jobname,command
        string'
```

Messages to the operator: DFHCLT TYPE=WTO

These two instructions define a message that is written to the system operator.



WTOL=addr
Specifies the address of a list format WTO macro that defines the message and any associated route codes and descriptor codes.

The MF (macro format), ROUTCDE (routing code), and DESC (descriptor) operands of the WTO macro are described in z/OS MVS Programming: Authorized Assembler Services Reference (Volume 1).

An example is to send a request to the operator:

```
          DFHCLT TYPE=WTO,
                 WTOL=wtoad
wtoad     WTO    'switch local terminals, please',
                  MF=L
```

Closing the command list: DFHCLT TYPE=LISTEND

This instruction defines the end of the set of commands and messages issued by an alternate system when it takes over from an active system.

PDIR: DL/I directory

The PDIR is a directory of all the remote program specification blocks (PSBs) that are accessed by the CICS system.

To use remote DL/I, you must define the PDIR to CICS. For DBCTL, you must define the DL/I directories to DBCTL using IMS-supplied macros.

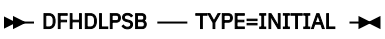
To create a program specification block directory (PDIR), code DFHDLPSB TYPE=INITIAL, TYPE=ENTRY, and TYPE=FINAL macros. The TYPE=FINAL (see TYPE=FINAL (end of table)) must be followed by:

```
END DFSIDIR0
```

Control section: DFHDLPSB TYPE=INITIAL

The TYPE=INITIAL macro establishes the control section (CSECT) for the table, and produces the necessary linkage editor control statements.

The DFHDLPSB TYPE=INITIAL macro has the following format and operands:

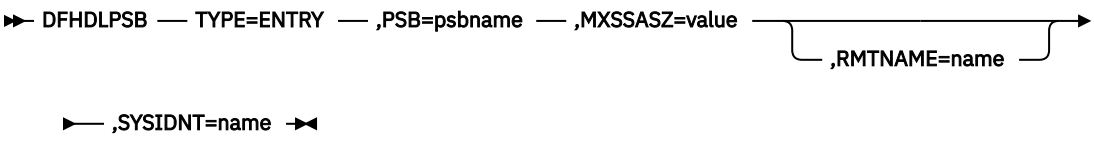


For general information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see TYPE=INITIAL (control section).

Program specification blocks: DFHDLPSB TYPE=ENTRY

The TYPE=ENTRY macro defines an entry to be generated in the PDIR.

The DFHDLPSB TYPE=ENTRY macro has the following format and operands:



TYPE=ENTRY
Indicates that an entry is to be generated in the PDIR. The maximum number of entries that can be included in the PDIR is 32760.

PSB=psbname
Specifies the name of the program specification block (PSB) accessed through the remote DL/I. The entry must specify the SYSIDNT and MXSSASZ (and optionally, RMTNAME) operands.

MXSSASZ=value
Specifies the maximum size in bytes of a segment search argument to be used for this PSB.

Note: An excessively large value for MXSSASZ affects performance badly, and may lead to a data stream being shipped which is too large for the connected CICS system.

RMTNAME=name
Indicates the name by which the PSB is known in the remote system or region. The default is the *psbname* specified in the PSB operand.

SYSIDNT=name
Indicates the 4-character alphanumeric name of the remote system or region for which the PSB is applicable. The name specified must be the name of the CONNECTION definition for the remote system.

If the SYSIDNT of the local system is specified, the request is routed to DBCTL and not function-shipped to a remote region. This allows the same PDIR to be used on both sides of the link if this is required. However, it is not necessary to have a PDIR when communicating with DBCTL.

FCT: file control table

The file control table (FCT) describes to CICS the Basic Direct Access Method (BDAM) user files that are processed by file management.

CICS user files correspond to physical data sets that must have been defined to MVS and allocated to the CICS system before they are used.

Note:

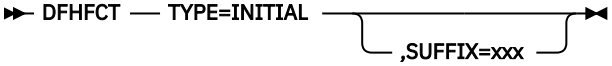
1. To define VSAM files, use a FILE resource.
2. Because CICS file management processes only VSAM and BDAM data sets, you define any sequential data sets as extrapartition destinations with a TDQUEUE resource.

The following macros specify file characteristics, and some of the characteristics of BDAM data sets referenced by the files:

- DFHFCT TYPE=INITIAL establishes the beginning of the FCT.
- DFHFCT TYPE=FILE defines the characteristics of a file, such as record characteristics and types of service allowed.
- DFHFCT TYPE=FINAL concludes the FCT. (See TYPE=FINAL (end of table).)

Control section: DFHFCT TYPE=INITIAL

The DFHFCT TYPE=INITIAL macro establishes the control sections into which the FCT is assembled, and must be coded as the first statement in the source used to assemble the FCT.

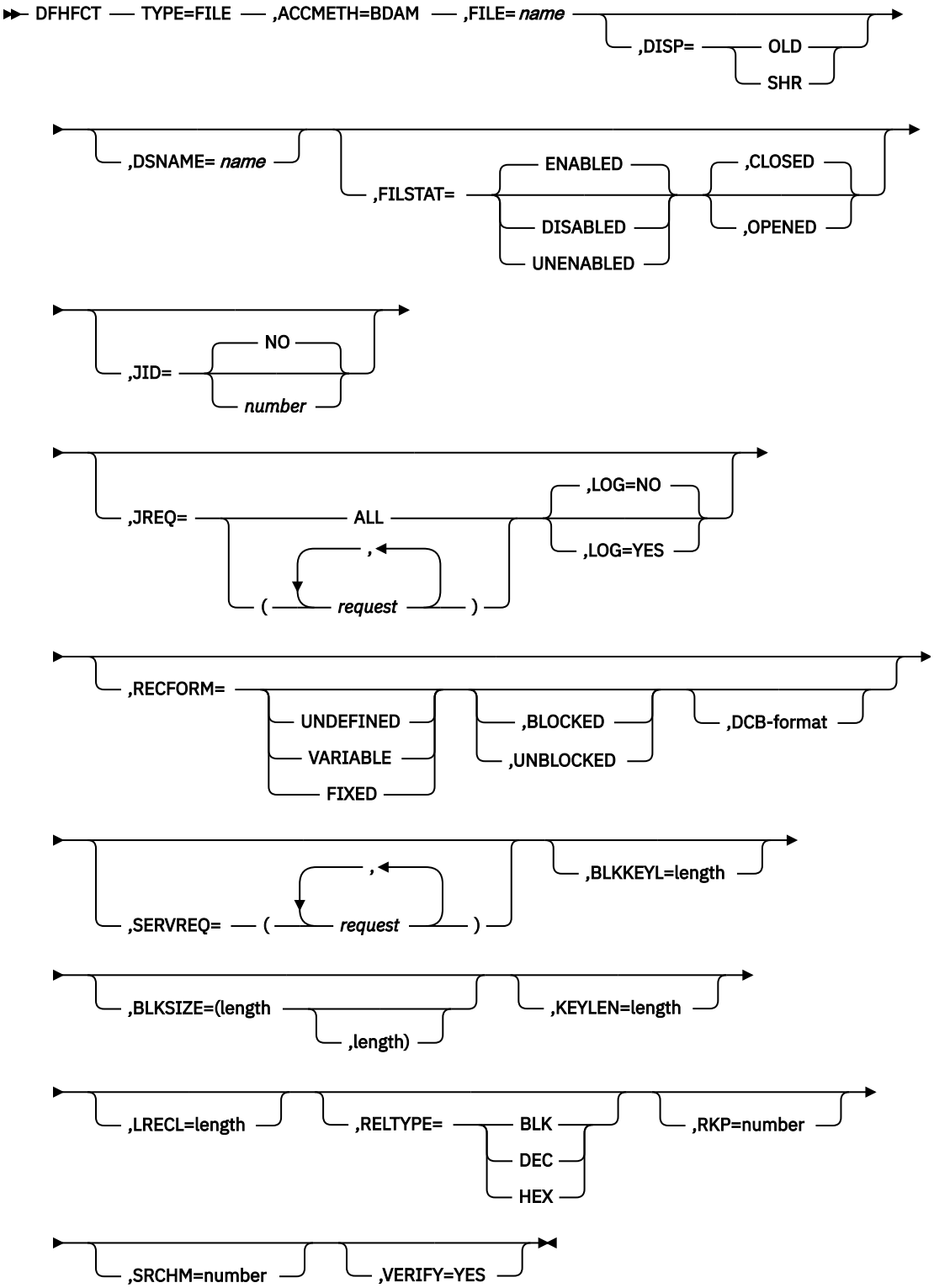


For general information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see TYPE=INITIAL (control section).

Local filesâ€”DFHFCT TYPE=FILE

The DFHFCT TYPE=FILE macro describes to CICS file control the physical and operational characteristics of a BDAM file.

This macro includes operands that provide information about the access method, record characteristics, and types of service allowed for the file. This information is used to generate control information used by CICS as well as a DCB .



TYPE=FILE
Indicates that this macro describes the characteristics of a file.

ACCMETH=BDAM
specifies the basic direct access method of the data set is allocated to the file.

BLKKEYL=length
Code this with a decimal value from 1 through 255, which represents the length in bytes of the physical key in the BDAM physical record. You must code this operand only for files that reference data sets with physical keys (that is, those with SERVREQ=KEY specified). If a data set contains blocked records, and deblocking is to be performed by using a logical key (that is, a key embedded within each logical record), the logical key length must be specified by using the KEYLEN operand.

If necessary, CICS can place a record under exclusive control by building an ENQ argument by concatenating the data set name, the block reference, and the physical key. An ENQ is issued using a maximum of 255 bytes of this argument. If the argument exceeds 255 bytes in length, the ENQ places a range of keys under exclusive control.

BLKSIZE=(length[,length])
Code this with the length of the block, in bytes. The way you calculate the BLKSIZE depends on the RECFORM. For UNDEFINED or VARIABLE blocks, the length must be the maximum block length. For FIXED length blocks, you calculate the BLKSIZE as follows:

`BLKSIZE = LRECL` for unblocked records.

`BLKSIZE = (LRECL x blocking factor)` for blocked records.

If you want to have a BLKSIZE value generated in the DCB, you must specify that value in the second parameter of the operand; for example, BLKSIZE=(250,250), where the first 250 relates to the FCT and the second 250 relates to the DCB. If the second parameter is not coded, the DCB is generated without a BLKSIZE value.

Note:

1. CICS assumes that the block size of the BDAM data set is the size you have specified on the first BLKSIZE parameter. If the value specified is smaller than the actual block size of the data set, you will probably get storage violations or other unpredictable results when using the file.
2. If you specify the second parameter (the DCB value) the value that you code must always be the actual block size. We recommend that you either omit the second parameter, or make it equal to the first parameter.

DISP={OLDâ¨SHR}
Code this to specify the disposition of the data set which will be allocated to this file. If no JCL statement exists for this file when it is opened, the open is preceded by a dynamic allocation of the file using this disposition. If a JCL statement does exist, it will take precedence over this disposition.

OLD
The disposition of the data set is set to OLD if dynamic allocation is performed.

SHR
The disposition of the data set is set to SHR if dynamic allocation is performed.

Note: You must specify the disposition of the data set, either with the DISP operand, or:

- In a JCL statement
- With CEMT SET
- With EXEC CICS SET

If you specify the disposition in a JCL statement, specify the data set name in the JCL statement as well.

DSNAME=name
Code from 1 to 44 characters to specify the JCL data set name (DSNAME) to be used for this file. If no JCL statement exists for this file when it is opened, the open will be preceded by a dynamic allocation of the file using this DSNAME. If a JCL statement does exist, it will take precedence over this DSNAME.

You must specify the data set name, either with the DSNAME operand, or:

- In a JCL statement
- With CEMT SET
- With EXEC CICS SET

If you specify the data set in a JCL statement, you **must** specify the disposition in the JCL statement as well.

Note: You define the CICS system definition (CSD) file by system initialization parameters, not in the FCT.

FILE=name
Code this with a 1-to 8-character symbolic name by which this FCT entry is to be identified. This name is known as the **file name** and is used by CICS or by CICS application programs to refer to the data set with which this FCT entry has been associated.

As well as identifying the FCT entry, this name is also used as the DDNAME when the associated data set is allocated to CICS. The allocation is achieved either by using JCL statements in the start-up job stream, or dynamically, by using the DSNAME and DISP values in the FCT.

Do not use file names that start with the character string DFH for your own files, because CICS reserves the right to use any file name beginning with DFH. In addition, using the character string FCT for a file name prefix can cause assembly errors.

FILSTAT=({ENABLEDâ¨DISABLEDâ¨UNENABLED},{OPENEDâ¨CLOSED})
Code this to specify the initial status of the file.

The first operand determines the initial enablement state of the file. It is used only during an initial or a cold start. (On a warm or emergency start, the file state is determined by the state at the time of the previous shutdown.)

The second operand specifies whether an attempt is made to open the file at the end of CICS initialization. It applies to initial, cold, warm, and emergency starts.

ENABLED
Normal processing is to be allowed against this file.

DISABLED
Any request against this file from an application program causes the DISABLED condition to be passed to the program.

UNENABLED
This option is valid only with the CLOSED option. It may be used to prevent the file being opened on first reference. An attempt to access the file in this state raises the NOTOPEN condition.

OPENED
The file is opened by an automatically initiated CICS transaction (CSFU) after CICS initialization. (On a warm or emergency start, a file remains UNENABLED, if that was its state at the time of the previous shutdown. CSFU ignores an OPENED option on an UNENABLED file, and leaves the file closed.)

CLOSED
The file is to remain closed until a request is made to open it by the main terminal function, by an EXEC CICS SET command, or by an implicit open.

For each combination of initial states, files are opened as follows:

(ENABLED,CLOSED)
The file is opened on first reference. This is the default.

(ENABLED,OPENED)
The file is opened by the automatically-initiated transaction CSFU after CICS initialization, unless a user application or main terminal function has opened it first.

(DISABLED,CLOSED)
The file is opened only by an explicit OPEN request (for example, from the main terminal transaction).

(DISABLED,OPENED)
The file is opened by the automatically-initiated transaction CSFU after CICS initialization, unless a user application or main terminal function has explicitly opened it first.

(UNENABLED,CLOSED)
The file is opened only by an explicit OPEN request. The file state after it has been opened is (ENABLED,OPENED).

Note: For performance reasons, the default CSD file entry for transaction CSFU is defined with DTIMOUT=10 (seconds). This can cause a transaction timeout abend if there is a delay in opening a file during CICS startup. See TRANSACTION attributes for an explanation of the DTIMOUT value.

JID={NOâ^¨number}
Code this if automatic journal activity is to take place for this FCT entry, and to identify the journal to be used to record the journaled data. The operations that cause data records to be journaled are specified in the JREQ parameter.

NO
No automatic journaling activity for this file is to take place.

number
The journal identifier to be used for automatic journaling. This may be any number in the range 01 through 99. The number is appended to the letters DFHJ to give a journal name of the form DFHJ*nn*, which maps to an MVS system logger general log stream.

Note: Automatic journaling can be specified if you want to record file activity for subsequent processing by yourself (for example, user-written data set I/O recovery). It must not be confused with automatic logging (specified with LOG=YES), which is required if CICS is to perform data set backout to remove in-flight task activity during emergency restart or dynamic transaction backout.

JREQ={ALLâ^¨(request[,request,...])}
Code this with the file operations that are to be automatically journaled, and whether the journaling operation is to be **synchronous** or **asynchronous** with file activity.

When a synchronous journal operation is executed for a READ request, control is not returned to the program that issued the file control request until the data read is written in the journal data set. When a synchronous journal operation is executed for a WRITE request, the output operation to the data set is not initiated until the data is written in the journal data set.

When an asynchronous journal operation is executed for a READ request, control can be returned as soon as the data read is moved to the journal I/O buffer. When an asynchronous journal operation is executed for a WRITE request, the output operation to the data set can be initiated as soon as the data is moved to the journal I/O buffer.

Synchronization defaults provide asynchronous operation for READs and synchronous operation for WRITEs.

If you have requested automatic journaling, the contents of the journal may not accurately reflect the actual changes to a data set, because the request is journaled before the response from the I/O operation is tested.

If this operand is omitted and JID is coded, JREQ defaults to JREQ=(WU,WN).

Here are the possible values for *request*:

ALL
Journal all file activity with READ asynchronous and WRITE synchronous.

ASY
Asynchronous journal operation for WRITE operations.

RO
Journal READ ONLY operations.

RU
Journal READ UPDATE operations.

SYN
Synchronous journal operation for READ operations.

WN
Journal WRITE NEW operations.

WU
Journal WRITE UPDATE operations.

KEYLEN=length
Code this with the length of the logical key for the deblocking of the BDAM data set to which this file refers.

The logical key for BDAM data sets is embedded and located through the use of the RKP operand. The length of the physical key is coded in the BLKKEYL operand, and can be different from the value specified for KEYLEN.

This operand must always be coded when logical keys are used in blocked BDAM data sets.

LOG={NOâ^¨YES}
This operand specifies the recovery attributes of the file. Specify LOG=YES if you want automatic logging. This enables backout (recovery) of incomplete changes to the data set referenced by this file, in the event of an emergency restart or transaction abend. Whenever a changeâ€”update, deletion, or additionâ€”is made to the data set, the â€œbeforeâ€ image is automatically recorded in the CICS system log. (Automatic logging should not be confused with automatic journaling.)

NO
Automatic logging is not to be performed.

YES
Automatic logging is to be performed.

When a request is made to change the contents of the data set referenced by the file, the record being updated, added, or deleted is enqueued upon, using the record identification together with the address of the CICS control block representing the base data set. This enqueue is maintained until the task terminates or the application issues a syncpoint request to signal the end of a logical unit of work. This ensures the integrity of the altered data.

Because the enqueues are thus maintained for a longer period of time, an enqueue lockout can occur if an application program that accesses this data set performs what is effectively more than one logical unit of work against it, without defining each separate logical unit of work to CICS by issuing a syncpoint request. Also, long-running tasks could tie up storage resources.

LRECL=length
Code this with the maximum length (in bytes) of the logical record. The value specified is also the length of records in a fixed length remote file. See the DFHFCT TYPE=REMOTE macro for further information on remote files.

RECFORM=([{UNDEFINEDâ^¨VARIABLEâ^¨FIXED}], [{BLOCKEDâ^¨UNBLOCKED}],[DCB format])
Code this to describe the format of physical records in the data set.

For BDAM data sets, **blocking** refers to CICS blocking, and has no meaning for BDAM. You must specify BLOCKED or UNBLOCKED for all data sets of FIXED or VARIABLE format.

BLOCKED
Specify this option when each physical record is to be viewed by CICS as a block consisting of more than one logical record.

DCB
Code this with the record format to be inserted in the DCB; for example, RECFORM=(FIXED,BLOCKED,FBS).

The DCB format sub-parameter of the RECFORM operand is the only way you can put record format information into the DCB when the FCT is assembled. The first two sub-parameters of the RECFORM operand do not generate information in the DCB.

FIXED
Records are fixed length.

UNBLOCKED
Specify this option when no CICS block structure is to be used. That is, when there is one CICS logical record for each BDAM physical record.

UNDEFINED
Records are of undefined length. (If you specify a data set as UNDEFINED, allow for an additional 8 bytes for the count field, when calculating the BLKSIZE.)

VARIABLE
Records are variable length.

RELTYPE={BLKâ¨DECâ¨HEX}
Code this if relative addressing is being used in the block reference portion of the record identification field of the BDAM data set referenced by this file. If the RELTYPE operand is omitted, absolute addressing is assumed (that is, MBBCCHHR).

BLK
Relative block addressing is being used.

DEC
The zoned decimal format is being used.

HEX
The hexadecimal relative track and record format is being used.

RKP=number
Code this with the starting position of the key field in the record relative to the beginning of the record. With variable-length records, this operand must include space for the 4-byte LLbb field at the beginning of each logical record. This operand must always be coded for data sets that have keys within each logical record, or when browsing.

SERVREQ=(request[,request],...)
Code this to define the types of service request that can be processed against the file. The parameters that can be included are as follows:

ADD
Records can be added to the file.

BROWSE
Records may be sequentially retrieved from the file.

KEY
Records can be retrieved from or added to the file. This parameter is mandatory if the data set referenced by the file is a keyed BDAM data set. It must not be coded for other files.

NOEXCTL
Records are not to be placed under exclusive control when a read for update is requested.

If you do not specify NOEXCTL, BDAM exclusive control is provided by default. This provides integrity in the system. For BDAM, you may specify LOG=YES with SERVREQ=NOEXCTL. This requests only a CICS enqueue and suppress the BDAM exclusive control, thus providing CICS integrity for the update only until a syncpoint.

Note: The CICS enqueue is at the record level within the CICS region, and lasts until a syncpoint, whereas the BDAM exclusive control operates on a physical block, is system-wide, and lasts only until the update is complete.

READ
Records in this file can be read. READ is assumed, if you specify BROWSE or UPDATE.

UPDATE
Records in this file can be changed.

SRCHM=number
Code this if multiple track search for keyed records is to be provided. This operand is applicable only to BDAM keyed data sets.

number
The number of tracks or blocks to be searched. The default is 0.

VERIFY=YES
Code this if you want to check the parity of disk records after they are written. If this operand is omitted, records are not verified after a write request.

Summary table

This section is intended to help you use the DFHFCT TYPE=FILE macro to define your files. Each TYPE=FILE instruction describes the characteristics of the file, and of the data set referenced by the file.

Table 12. DFHFCT TYPE=FILE instructions for BDAM files

	Blocked with key	**Blocked without key**	**Unblocked with key**	**Unblocked without key**
BLKKEYL	R		R	
SRCHM	O		O	
VERIFY	O	O	O	O
RELTYPE	R“1” on page 364	R“1” on page 364	R“1” on page 364	R“1” on page 364
LRECL	R	R	R	R
BLKSIZE	R“3” on page 364	R	R“2” on page 364	R
KEYLEN	R“5” on page 364			
RKP	R“4” on page 364		R“4” on page 364	
RECFORM	O	O	O	O
FILSTAT	O	O	O	O
SERVREQ	R“6” on page 364	O	R“6” on page 364	O

Notes:

R
Required

O
Optional

1. Required if relative type addressing is to be used.
2. If SERVREQ=BROWSE or SERVREQ=ADD, this value must be BLKSIZE + BLKKEYL for unblocked records.
3. If SERVREQ=BROWSE or SERVREQ=ADD, this value must be (LRECL x blocking factor) + BLKKEYL for blocked records.
4. Required if key exists within logical records.
5. Required if deblocking by key for BDAM.
6. SERVREQ=KEY is required.

DFHFCT example

An example of an FCT entry for a BDAM file.

Figure 7 on page 365 illustrates the coding required to create an FCT entry for a BDAM file.

```
DFHFCT TYPE=FILE,                                      *
       FILE=DAM83,                                     *
       ACCMETH=BDAM,                                   *
       SERVREQ=(READ,BROWSE,KEY),                      *
       BLKSIZE=172,                                    *
       RECFORM=(FIXED,BLOCKED),                        *
       LRECL=86,                                       *
       RELTYPE=HEX,                                    *
       KEYLEN=6,                                       *
       BLKKEYL=6,                                      *
       RKP=0,                                          *
       FILSTAT=(ENABLED,OPENED)
```

Figure 7. File control table example: BDAM file

MCT: monitoring control table

The monitoring control table (MCT) defines the user data fields in CICS monitoring performance class records and describes how they are manipulated at event monitoring points (EMPs). It also controls which system-defined performance class data fields are recorded.

See The monitoring control table (MCT) for details of the MCT and performance.

The MCT is required only in these circumstances:

- You call EMPs in your application programs.
- You need to exclude specific system-defined fields from being recorded.
- You want to use monitoring resource classes.
- You want to collect additional monitoring performance data for the resource managers used by your transaction.
- You want to deactivate data compression on monitoring records.

If no MCT is present in the CICS region, CICS assumes the following defaults:

- The performance, transaction resource, and exception monitoring classes are available.
- All CICS system-defined data fields are collected.
- Data compression is performed on monitoring records.

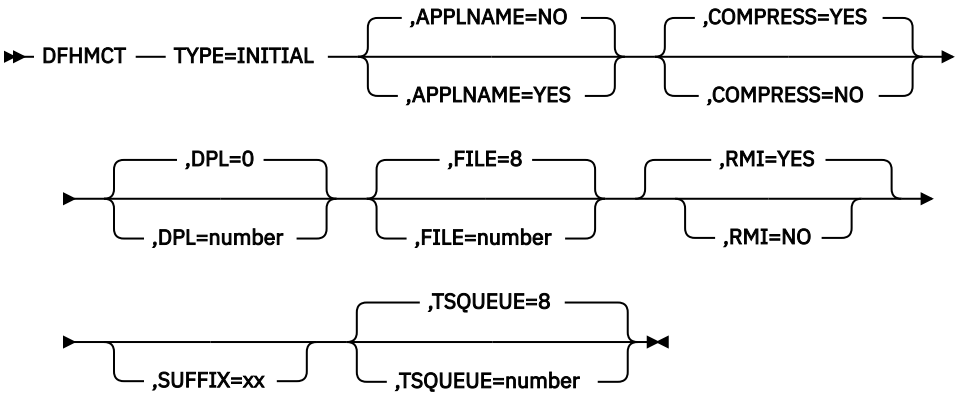
In your application programs, you can call EMPs using MONITOR.

The MCT consists of the following macro instructions:

- Control section, DFHMCT TYPE=INITIAL
- User event monitoring points, DFHMCT TYPE=EMP
- Control data recording, DFHMCT TYPE=RECORD
- End of monitoring control table, DFHMCT TYPE=FINAL (described in TYPE=FINAL (end of table))

Control section: DFHMCT TYPE=INITIAL

The control section name for the MCT is established by the DFHMCT TYPE=INITIAL macro. This macro also creates the necessary linkage editor control statements for subsequent link-editing.



For general information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see TYPE=INITIAL (control section).

Note: The combined length of all the transaction resource monitoring data must not exceed 32244 bytes.

APPLNAME={NO|YES}
This option specifies that you want to use the application naming support provided by CICS monitoring.

Application naming is an enabling function that allows application programs to call special CICS event monitoring points. Data collected at these CICS-generated EMPs can be used by any CICS monitoring software package.

NO
Application naming support is not enabled in the CICS region and the application naming event monitoring points, DFHAPPL.1, and DFHAPPL.2, are not generated.

YES
Application naming support is enabled in the CICS region. When you assemble the MCT, CICS generates the application naming event monitoring points (DFHAPPL.1 and DFHAPPL.2). Note the monitoring data moved at these EMPs by a command MONITOR that is calling these application naming EMPs is preserved until the end of the task, or until it is changed by another call of the EMPS by a subsequent **EXEC CICS MONITOR** command.

The application naming (DFHAPPL) EMPs are created by CICS as if defined with the TYPE=EMP macro parameters:

```
DFHMCT TYPE=EMP,CLASS=PERFORM,                                  X
               ID=(DFHAPPL.1),FIELD=(1,APPLNAME)                X
               PERFORM=(MOVE(0,4))
DFHMCT TYPE=EMP,CLASS=PERFORM,                                  X
               ID=(DFHAPPL.2),                                  X
               PERFORM=(MOVE(4,8))
```

For more information about how to use the application naming event monitoring points in your applications, see Application naming event monitoring points.

COMPRESS={YES|NO}
This option specifies whether you want data compression to be performed for the CICS SMF 110 monitoring records produced by the CICS monitoring facility.

YES
This default specifies that you want monitoring record data compression to be performed for the CICS SMF 110 monitoring records produced by the CICS monitoring facility. For information about data compression, see Data compression for monitoring records.

NO
Specifies that you do not want monitoring record data compression to be performed for the CICS SMF 110 monitoring records output by the CICS monitoring facility.

DPL={0|*number*}
This option specifies the maximum number of distributed program link (DPL) requests for which you want CICS to perform transaction resource monitoring. This option applies only if transaction resource monitoring is enabled, either by specifying MNRES=ON as a system initialization parameter, or by enabling it dynamically using the monitoring facility transaction CEMN or an EXEC CICS, or CEMT, SET MONITOR command.

CICS standard monitoring performance class data includes totals for *all* programs accessed by a transaction, including distributed program links. Transaction resource monitoring, alternatively, collects information about individual distributed program links, up to the number specified. The following data is collected:

- Program name
- System identifier (sysid) where the request is routed
- Number of distributed program link (DPL) requests

0
This default specifies that CICS is not to perform transaction resource monitoring for any distributed program links.

number
Specifies the maximum number of distributed program link requests, in the range 0 - 64, for which CICS is to perform transaction resource monitoring. CICS collects monitoring performance data at the resource level for each distributed program link request issued by a transaction, up to the maximum specified by *number*. If the transaction issues more distributed program link requests than the number specified, any requests over the maximum are ignored, but a flag is set to indicate that the transaction has exceeded the DPL limit.

If you specify DPL=0, specifying MNRES=YES either as a system initialization parameter or dynamically while CICS is running has no effect, and transaction resource monitoring data is not collected for distributed program links.

FILE={8|*number*}
This option specifies the maximum number of files for which you want CICS to perform transaction resource monitoring. This option applies only if transaction resource monitoring is enabled, either by specifying MNRES=ON as a system initialization parameter, or by enabling it dynamically using the monitoring facility transaction CEMN or an EXEC CICS, or CEMT, SET MONITOR command.

CICS standard monitoring performance class data includes totals for *all* files accessed by a transaction. Transaction resource monitoring, alternatively, collects information about individual files, up to the number specified. The following data is collected:

- File name
- Number and total time of file get requests
- Number and total time of file put requests
- Number and total time of file browse requests
- Number and total time of file add requests
- Number and total time of file delete requests
- Total number and total time of all requests against the file
- File access method request count
- File I/O wait time and number of waits
- RLS-mode file I/O wait time and number of waits
- Coupling facility data table (CFDT) I/O wait time and number of waits
- Exclusive control wait time and number of waits

- VSAM string wait time and number of waits

8
This default specifies that CICS is to perform transaction resource monitoring for a maximum of 8 files.

number
Specifies the maximum number of files, in the range 0 - 64, for which CICS is to perform transaction resource monitoring. CICS collects monitoring performance data at the resource level for each file accessed by a transaction, up to the maximum specified by *number*. If the transaction accesses more files than the number specified, any files over the maximum are ignored, but a flag is set to indicate that the transaction has exceeded the file limit.

If you specify FILE=0, specifying MNRES=ON either as a system initialization parameter or dynamically while CICS is running has no effect, and transaction resource monitoring data is not collected for files.

RMI={YES|NO}
This option specifies whether you want additional monitoring performance class data to be collected for the resource managers used by your transactions.

YES
This default specifies that you do want additional monitoring performance data to be collected for the resource managers used by your transactions.

For information about the data that is collected see Performance data in group DFHRMI.

NO
This specifies that you do not want monitoring performance data for the resource managers used by your transactions.

TSQUEUE={8|*number*}
This option specifies the maximum number of temporary storage queues for which you want CICS to perform transaction resource monitoring. This option applies only if transaction resource monitoring is enabled, either by specifying MNRES=ON as a system initialization parameter, or by enabling it dynamically using the monitoring facility transaction CEMN or an EXEC CICS, or CEMT, SET MONITOR command.

CICS standard monitoring performance class data includes totals for *all* temporary storage queues accessed by a transaction. Transaction resource monitoring, alternatively, collects information about individual temporary storage queues, up to the number specified.

number
Specifies the maximum number of temporary storage queues, in the range 0 - 64, for which CICS is to perform transaction resource monitoring. CICS collects monitoring performance data at the resource level for each temporary storage queue accessed by a transaction, up to the maximum specified by *number*. If the transaction accesses more temporary storage queues than the number specified, any temporary storage queues over the maximum are ignored, but a flag is set to indicate that the transaction has exceeded the temporary storage queue limit.

If you specify TSQUEUE=0, specifying MNRES=ON either as a system initialization parameter or dynamically while CICS is running has no effect, and transaction resource monitoring data is not collected for temporary storage queues. The following data is collected:

- Temporary storage queue name
- Number and total time of temporary storage queue get requests
- Number and total time of temporary storage queue get requests to shared temporary storage
- Number and total time of temporary storage queue put requests to auxiliary temporary storage
- Number and total time of temporary storage queue put requests to main temporary storage
- Number and total time of temporary storage queue put requests to shared temporary storage
- Total number and total time of all requests against the temporary storage queue
- Total length of all the items obtained from temporary storage

- Total length of all the items obtained from shared temporary storage
- Total length of all the items written to auxiliary temporary storage
- Total length of all the items written to main temporary storage
- Total length of all the items written to shared temporary storage
- Temporary storage I/O wait time and number of waits
- Shared temporary storage I/O wait time and number of waits

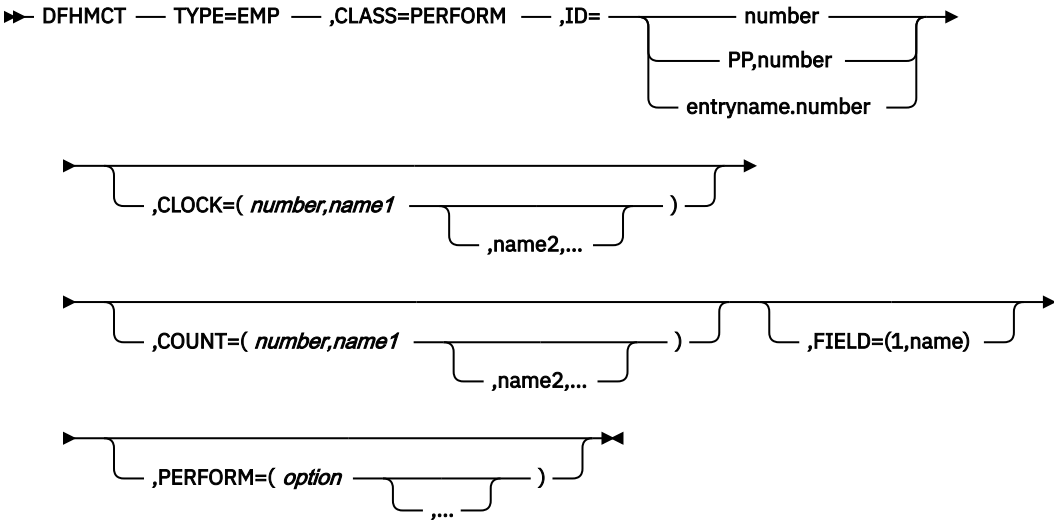
8

This default specifies that CICS is to perform transaction resource monitoring for a maximum of 8 temporary storage queues.

User event monitoring points: DFHMCT TYPE=EMP

You can use the DFHMCT TYPE=EMP macro to specify how the user data fields in performance class data records are added to or changed at each user event monitoring point. One TYPE=EMP macro must be coded for each user event monitoring point (EMP) at which user data is required.

The TYPE=EMP macro must be coded between the TYPE=INITIAL macro and the first TYPE=RECORD macro instruction.



TYPE=EMP
Indicates that this macro defines the user data to be collected at a user event monitoring point.

CLASS=PERFORM
Code this with the monitoring classes for which you want user data to be collected at this user EMP. The value PERFORM must be coded. The corresponding PERFORM operand must also be coded.

ID={*number*|(PP,*number*)|*entryname.number*}
Code this with the identifier of the user event monitoring point at which the user data defined in this macro is to be collected. If one of the forms *number* or (PP,*number*) is coded, a default entry name, USER, is provided.

number
A decimal integer in the range 1 through 255. Identification numbers between 1 and 199 are available for user EMPs. Numbers between 200 and 255 are reserved for IBM licensed program EMPs. Code these numbers if you want to collect user data at EMPs defined in the code of IBM licensed programs.

(PP,*number*)
An IBM licensed program EMP identification number. It is equivalent to specifying an ID value of 199 + number. The value of *number* is a decimal integer in the range 1 through 56.

entryname.number
Allows multiple use of *number*, a decimal integer in the range 1 through 255. Thus UNIQUE.3, DSN.3, and 3 are three different EMPs. A maximum of 98 entrynames can be specified against any particular number. Also, any count, clock, or byte-offset referred to by one of them is a different object from that referred to by any other.

In the following descriptions, any reference to a constant means a hexadecimal constant of up to eight hexadecimal digits; any shorter string is padded on the left with zeros. For example, to add or subtract decimal 14, the constant would be coded as 0000000E or just E (no quotation marks are required).

Any reference to the fields DATA1 and DATA2 means the two binary fullwords supplied by the user EMP coded in the application program. These are specified by the DATA1 and DATA2 operands of the EXEC CICS MONITOR command for defining user EMPs. Depending on the options coded, the DATA1 and DATA2 fields can be interpreted as numbers, masks for performing logical operations, or pointers to further information.

Any reference to a number means a decimal integer in the range defined in the description of the option.

CLOCK=(*number*,*name1*[,*name2*,...])
Assigns an informal name to one or more clocks. The informal name of any clock appears in its dictionary entry and is available to a postprocessor for use as, for example, a column heading.

The character string *name1* is assigned to the clock specified by *number* at MCT generation. If specified, *name2* is assigned to the clock *number* + 1. Similarly, any subsequent names are assigned to subsequent clocks. Any clock not named by this option receives the entry name value from the ID operand (the default is USER).

number must be in the range 1 through 256. The names specified must each be a character string up to eight characters long. If any string contains one or more blanks or commas, it must be enclosed in quotes.

COUNT=(number,*name1*[,*name2*,...])
Assigns an informal name to one or more count fields. The informal name of any count field appears in its dictionary entry and is available to a postprocessor for use as, for example, a column heading.

The character string *name1* is assigned to the count field specified by number at MCT generation. If specified, *name2* is assigned to the count field *number*+1. Similarly, any subsequent names are assigned to subsequent count fields. Any count fields not named by this option receive the entry name value from the ID operand (the default is USER).

number must be in the range 1 through 256. The names specified must each be a character string up to eight characters long. If any string contains one or more blanks or commas, it must be enclosed in quotes.

FIELD=(1,*name*)
Assigns an informal name to the user byte-string field. This appears in its dictionary entry and is available to a postprocessor for use as, for example, a column heading.

name must be a character string up to 8 characters long. If it contains one or more blanks or commas, it must be enclosed in quotes.

PERFORM=(*option*[,...])
Code this operand when CLASS=PERFORM is specified. It specifies that information is to be added to or changed in the user fields of the performance class data record at this EMP.

The user fields for each user EMP distinguished by a separate entry name in the ID operand can comprise:

1. Up to 256 counters
2. Up to 256 clocks, each made up of a 8-byte accumulator and 4-byte count
3. A byte string of up to 8192 bytes.

Note: If the combined sizes of the objects (clocks, counts, and fields) implied in the specified options exceed 16384 bytes, assembly-time errors occur. You can avoid this by using fewer objects, either by collecting less data, or by clustering references to clocks and counts to avoid implied, but unused, objects.

Note: When you define user data to be collected at a user event monitoring point, this extends the size of all CICS performance class monitoring records. Each CICS monitoring record is the same size as the largest record; bear this in mind when specifying user data fields.

Actions are performed on the user fields according to the options specified.

PERFORM can be abbreviated to PER. Valid options for the PERFORM operand are:

ADDCNT(*number*,{*constant*|DATA1|DATA2})
The value of the user count field specified by *number* is to be incremented by *constant* or by the value of the field DATA1 or DATA2. *number* is a decimal integer in the range 1 through 256.

EXCNT(*number*,{*constant*|DATA1|DATA2})
A logical exclusive OR operation is to be performed on the value of the user count field specified by *number*, using *constant* or the value of the field DATA1 or DATA2. *number* is a decimal integer in the range 1 through 256.

MLTCNT(*number1*,*number2*)
A series of adjacent user count fields are to be updated by adding the values contained in adjacent fullwords in an area addressed by the DATA1 field. To use this option, both the DATA1 and DATA2 fields must be passed from the user EMP.

The user count fields that are to be updated start at the field specified by *number1*. The number of user count fields that are updated is the smaller of the values of *number2* and the DATA2 field. If the DATA2 field is zero, the value of *number2* is used. The series of adjacent fullwords used to add into the user count fields starts at the address specified in the DATA1 field. Successive fullwords are added into successive user count fields.

number1 and *number2* are decimal integers in the range 1 through 256. The number of user counts generated is (*number1* + *number2* - 1). This value must also be in the range 1 through 256.

Note: Only one of the MLTCNT and MOVE options can be used in each DFHMCT TYPE=EMP macro.

MOVE(*number3*,*number4*)
A string of data is to be moved into the user byte-string field. To use this option, both the DATA1 and DATA2 fields must be passed from the user EMP.

The user byte-string field is updated starting at the offset specified by *number3*. The data to be moved starts at the address supplied in the DATA1 field. The maximum length of data that can be moved is given by *number4* (in bytes), and the actual length of data that is to be moved is given by the value of the DATA2 field. If the value of DATA2 is zero, the length of the data given by *number4* is moved.

number3 is a decimal integer in the range 0 to 8191, and *number4* is a decimal integer in the range 1 to 8192. The maximum length of the user character field is (*number3* + *number4*), and must be in the range 1 to 8192.

Note: Only one of the MLTCNT and MOVE options can be used in each DFHMCT TYPE=EMP macro instruction.

NACNT(*number*,{*constant*|DATA1|DATA2})
A logical AND operation is to be performed on the value of the user count field specified by *number*, using *constant* or the value of the field DATA1 or DATA2. *number* is a decimal integer in the range 1 through 256.

ORCNT(*number*,{*constant*|DATA1|DATA2})
A logical inclusive OR operation is to be performed on the value of the user count field specified by *number*, using *constant* or the value of the field DATA1 or DATA2. *number* is a decimal integer in the range 1 through 256.

PCLOCK(*number*)
The clock specified by number is to be stopped. The 4-byte count in the user clock field is flagged to indicate that the clock is now stopped. The accumulator is set to the sum of its contents before the previous SCLOCK and the elapsed period between that SCLOCK and this PCLOCK. *number* is a decimal integer in the range 1 through 256.

PCPUCLK(*number*)
This option performs the same function as PCLOCK, but uses the CPU-time of the CICS main task instead of elapsed time.

SCLOCK(*number*)
The clock specified by *number* is to be started. The value of the 4-byte count in the user clock field is incremented by 1 and flagged to show its running state. *number* is a decimal integer in the range 1 through 256.

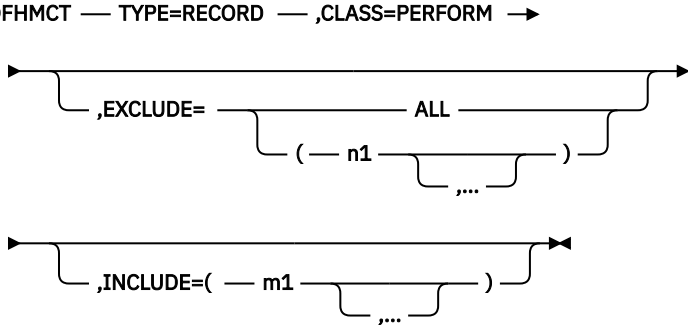
SCPUCLK(*number*)
This option performs the same function as SCLOCK, but uses the CPU-time of the CICS main task instead of elapsed time.

SUBCNT(*number*,{*constant*|DATA1|DATA2})
The value of the user count field specified by *number* is to be decremented by *constant* or by the value of the field DATA1 or DATA2. *number* is a decimal integer in the range 1 through 256.

DELIVER
Performance class data accumulated for this task up to this point is delivered to the monitoring buffers. Any running clocks are stopped. The performance class section of the monitoring area for this task is reset to X'00', except for the key fields (transid, termid) and any data stored as a result of invoking the DFHAPPL special EMPs. Any clocks that were stopped by this option are restarted from zero for the new measurement period. The high-water mark fields are reset to their current values.

Control data recording: DFHMCT TYPE=RECORD

The DFHMCT TYPE=RECORD macro identifies the performance class data fields that are selected for monitoring.



TYPE=RECORD
Indicates that monitoring data for selected performance class data fields will be recorded.

CLASS=PERFORM
Code this operand to record performance class data fields. You can abbreviate PERFORM to PER.

EXCLUDE={ALL|(*n1*[,...])}
Code this operand to prevent one or more CICS fields from being reported by the monitoring facility. By default, all documented performance class fields are reported.

The EXCLUDE operand is always processed before the INCLUDE operand, regardless of the order in which they are coded. (The INCLUDE operand is only relevant when the EXCLUDE operand is coded.)

ALL
This option prevents all fields that are eligible for exclusion from being reported. The following fields cannot be excluded:

- 1
- 2
- 4
- 5
- 6
- 89

You can use the INCLUDE operand at the same time as EXCLUDE=ALL if you want to include some fields but exclude the majority.

Table 13 on page 373 shows the fields that are eligible for exclusion. Each field has a group name associated with it, which identifies the group of fields to which it belongs. Each field also has its own numeric field identifier.

To exclude a group of fields you code the name of the group (a character string) as *n1*, for example, `EXCLUDE=(DFHTASK)`.

To exclude a single field you code the numeric identifier of the field as *n1*, for example, `EXCLUDE=(98,70)`.

Note: Do not code leading zeros on numeric identifiers. Do not code numeric identifiers of fields that are ineligible for exclusion.

You can code combinations of names and numeric identifiers, for example, `EXCLUDE=(DFHFILE,DFHTERM,112,64)`.

Table 13. Data groups and fields that can be excluded or included

Group Name	Field Id	Field Name	Description
DFHCBTS	200	PRCSNAME	CICS BTS process name
DFHCBTS	201	PRCSTYPE	CICS BTS process type
DFHCBTS	202	PRCSID	CICS BTS process ID
DFHCBTS	203	ACTVTYID	CICS BTS activity ID
DFHCBTS	204	ACTVTYNM	CICS BTS activity name
DFHCBTS	205	BARSYNCT	CICS BTS run process/activity synchronous count
DFHCBTS	206	BARASYCT	CICS BTS run process/activity asynchronous count
DFHCBTS	207	BALKPACT	CICS BTS link process/activity count
DFHCBTS	208	BADPROCT	CICS BTS define process count
DFHCBTS	209	BADACTCT	CICS BTS define activity count
DFHCBTS	210	BARSPACT	CICS BTS reset process/activity count
DFHCBTS	211	BASUPACT	CICS BTS suspend process/activity count
DFHCBTS	212	BARMPACT	CICS BTS resume process/activity count
DFHCBTS	213	BADCPACT	CICS BTS delete activity or cancel process/activity request count
DFHCBTS	214	BAACQPCT	CICS BTS acquire process/activity request count
DFHCBTS	215	BATOTPCT	CICS BTS total process/activity request count
DFHCBTS	216	BAPRDCCT	CICS BTS delete/get/put process container count
DFHCBTS	217	BAACDCCT	CICS BTS delete/get/put activity container count
DFHCBTS	218	BATOTCCT	CICS BTS total process/activity container request count

Table 13. Data groups and fields that can be excluded or included (continued)

Group Name	Field Id	Field Name	Description
DFHCBTS	219	BARATECT	CICS BTS retrieve reattach request count
DFHCBTS	220	BADFIECT	CICS BTS define input event request count
DFHCBTS	221	BATIAECT	CICS BTS timer associated event requests count
DFHCBTS	222	BATOTECT	CICS BTS total event related request count
DFHCHNL	321	PGTOTCCT	Number of CICS requests for channel containers
DFHCHNL	322	PGBRWCCT	Number of browse requests for channel containers
DFHCHNL	323	PGGETCCT	Number of GET CONTAINER and GET64 CONTAINER requests for channel containers
DFHCHNL	324	PGPUTCCT	Number of PUT CONTAINER and PUT64 CONTAINER requests for channel containers
DFHCHNL	325	PGMOVCCT	Number of MOVE CONTAINER requests for channel containers
DFHCHNL	326	PGGETCDL	Length of data in the containers of all GET CONTAINER CHANNEL and GET64 CONTAINER CHANNEL commands
DFHCHNL	327	PGPUTCDL	Length of data in the containers of all PUT CONTAINER CHANNEL and PUT64 CONTAINER CHANNEL commands
DFHCHNL	328	PGCRECCT	Number of containers created by MOVE, PUT CONTAINER, and PUT64 CONTAINER requests for channel containers
DFHCHNL	329	PGCSTHWM	Maximum amount (high-water mark) of container storage allocated
DFHCICS	25	CFCAPICT	CICS OO foundation class request count
DFHCICS	103	EXWTTIME	Transaction exception wait time
DFHCICS	112	RTYPE	Performance record type
DFHCICS	130	RSYSID	Transaction routing SYSID
DFHCICS	131	PERRECNT	Performance record count
DFHCICS	167	SRVCLASS	MVS Workload Manager Service Class name
DFHCICS	168	RPTCLASS	MVS Workload Manager Report Class name
DFHCICS	351	OADID	The adapter identifier added to the origin data by the adapter
DFHCICS	352	OADATA1	The data added to the origin data by the adapter
DFHCICS	353	OADATA2	The data added to the origin data by the adapter
DFHCICS	354	OSDATA3	The data added to the origin data by the adapter
DFHCICS	359	ONETWKID	The network identifier from which this work request transaction originated
DFHCICS	360	OAPPLID	APPLID of the CICS region in which this work request originated
DFHCICS	361	OSTART	Time at which the originating task was started
DFHCICS	362	OTRANNUM	Number of the originating task
DFHCICS	363	OTRAN	Transaction ID (TRANSID) of the originating task

Table 13. Data groups and fields that can be excluded or included (continued)

Group Name	Field Id	Field Name	Description
DFHCICS	364	OUSERID	Originating Userid-2 or Userid-1, depending on the originating task
DFHCICS	365	OUSERCOR	Originating user correlator
DFHCICS	366	OTCPSVCE	Name of the originating TCPIPSERVICE
DFHCICS	367	OPORTNUM	Port number used by the originating TCPIPSERVICE
DFHCICS	369	OCLIPORT	TCP/IP port number of the originating client
DFHCICS	370	OTRANFLG	Originating transaction flags
DFHCICS	371	OFCTYNME	Facility name of the originating transaction
DFHCICS	372	OCLIPADR	IP address of the originating client or Telnet client.
DFHCICS	373	PHNTWKID	The network identifier of the CICS system of an immediately previous task in another CICS system with which this task is associated
DFHCICS	374	PHAPPLID	The APPLID from previous hop data
DFHCICS	375	PHSTART	The start time of the immediately previous task in another CICS system with which this task is associated
DFHCICS	376	PHTRANNO	The task number of the immediately previous task in another CICS system with which this task is associated
DFHCICS	377	PHTRAN	The transaction identifier (TRANSID) of the immediately previous task in another CICS system with which this task is associated
DFHCICS	378	PHCOUNT	The number of times there has been a request from one CICS system to another CICS system to initiate a task with which this task is associated.
DFHCICS	402	EICTOTCT	EXEC CICS API and SPI request count
DFHCICS	405	TIASKTCT	ASKTIME request count
DFHCICS	406	TITOTCT	TIME request total count
DFHCICS	408	BFDGSTCT	BIF DIGEST request count
DFHCICS	409	BFTOTCT	BIF request total count
DFHCICS	415	ECSIGECT	SIGNAL EVENT request count
DFHCICS	416	ECEFOPCT	Event filter operations count
DFHCICS	417	ECEVNTCT	Events captured count
DFHCICS	418	ECSEVCCT	The number of synchronous emission events captured by the user task
DFHCICS	449	MPPRTXCD	The number of policy task rule thresholds that this task has exceeded
DFHCICS	464	NCGETCT	Number of GET COUNTER and DCOUNTER requests
DFHCICS	466	MPSRECT	The number of times that policy system rules have been evaluated for the task
DFHCICS	467	MPSRACT	The number of times that policy system rules that have evaluated true and have triggered either a message or an event

Table 13. Data groups and fields that can be excluded or included (continued)

Group Name	Field Id	Field Name	Description
DFHCICS	480	PTSTART	The start time of the immediately previous or parent task in the same CICS system with which the task is associated.
DFHCICS	481	PTTRANNO	The task number of the immediately previous or parent task in the same CICS system with which the task is associated
DFHCICS	482	PTTRAN	The transaction identifier (TRANSID) of the immediately previous or parent task in the same CICS system with which the task is associated
DFHCICS	483	PTCOUNT	The number of times there has been a request from one task to initiate another task in the same CICS system with which this task is associated, such as by a **RUN TRANSID ASYNCHRONOUS** or **START** command
DFHDATA	179	IMSREQCT	IMS (DBCTL) request count
DFHDATA	180	DB2REQCT	DB2 request count
DFHDATA	186	IMSWAIT	IMS (DBCTL) wait time
DFHDATA	187	DB2RDYQW	DB2 Readyq wait time
DFHDATA	188	DB2CONWT	DB2 Connection wait time
DFHDATA	395	WMQREQCT	MQ request count
DFHDATA	396	WMQGETWT	MQ GETWAIT wait time
DFHDATA	397	WMQASRBT	WebSphere MQ API SRB time
DFHDEST	41	TDGETCT	TD get count
DFHDEST	42	TDPUTCT	TD put count
DFHDEST	43	TDPURCT	TD purge count
DFHDEST	91	TDTOTCT	TD total count
DFHDEST	101	TDIOWTT	TD I/O wait time
DFHDEST	403	TDILWTT	TD intrapartition lock wait time
DFHDEST	404	TDELWTT	TD extrapartition lock wait time
DFHDOCH	223	DHDELCT	Document handler Delete count
DFHDOCH	226	DHCRECT	Document handler Create count
DFHDOCH	227	DHINSCT	Document handler Insert count
DFHDOCH	228	DHSETCT	Document handler Set count
DFHDOCH	229	DHRETCT	Document handler Retrieve count
DFHDOCH	230	DHTOTCT	Document handler Total count
DFHDOCH	240	DHTOTDCL	Document handler total created document length
DFHFEPI	150	SZALLOCT	FEPI allocate count
DFHFEPI	151	SZRCVCT	FEPI receive count
DFHFEPI	152	SZSENDCT	FEPI send count
DFHFEPI	153	SZSTRTCT	FEPI start count

Table 13. Data groups and fields that can be excluded or included (continued)

Group Name	Field Id	Field Name	Description
DFHFEPI	154	SZCHROUT	FEPI CHARS sent
DFHFEPI	155	SZCHRIN	FEPI CHARS received
DFHFEPI	156	SZWAIT	FEPI suspend time
DFHFEPI	157	SZALLCTO	FEPI allocate timeout count
DFHFEPI	158	SZRCVTO	FEPI receive timeout count
DFHFEPI	159	SZTOTCT	FEPI total count
DFHFILE	36	FCGETCT	FC get count
DFHFILE	37	FCPUTCT	FC put count
DFHFILE	38	FCBRWCT	FC browse count
DFHFILE	39	FCADDCT	FC add count
DFHFILE	40	FCDELCT	FC delete count
DFHFILE	63	FCIOWTT	FC I/O wait time
DFHFILE	70	FCAMCT	FC access-method count
DFHFILE	93	FCTOTCT	FC total count
DFHFILE	174	RLSWAIT	RLS FC I/O wait time
DFHFILE	175	RLSCPUT	RLS File request CPU (SRB) time
DFHFILE	176	CFDTWAIT	CFDT I/O wait time
DFHFILE	426	FCXCWTT	File control wait time for exclusive control of a VSAM control interval
DFHFILE	427	FCVSWTT	File control wait time for a VSAM string
DFHJOUR	10	JCIOWTT	Journal I/O wait time
DFHJOUR	58	JNLWRTCT	Journal write count
DFHJOUR	172	LOGWRTCT	Log stream write count
DFHMAPP	50	BMSMAPCT	BMS MAP count
DFHMAPP	51	BMSINCT	BMS IN count
DFHMAPP	52	BMSOUTCT	BMS OUT count
DFHMAPP	90	BMSTOTCT	BMS total count
DFHPROG	55	PCLINKCT	Program LINK count
DFHPROG	56	PCXCTLCT	Program XCTL count
DFHPROG	57	PCLOADCT	Program LOAD count
DFHPROG	71	PGMNAME	Program name
DFHPROG	72	PCLURMCT	Program LINK_URM count
DFHPROG	73	PCDPLCT	Program DPL count
DFHPROG	113	ABCODEO	Original abend code
DFHPROG	114	ABCODEC	Current abend code
DFHPROG	115	PCLOADTM	Program load time

Table 13. Data groups and fields that can be excluded or included (continued)

Group Name	Field Id	Field Name	Description
DFHPROG	286	PCDLCSDL	Length of data in the containers of all DPL requests with the CHANNEL option
DFHPROG	287	PCDLCRDL	Length of data in the containers of all DPL RETURN CHANNEL commands
DFHPROG	306	PCLNKCCT	Number of local LINK requests with CHANNEL option
DFHPROG	307	PCXCLCCT	Number of XCTL requests with CHANNEL option
DFHPROG	308	PCDPLCCT	Number of DPL requests with CHANNEL option
DFHPROG	309	PCRTNCCT	Number of remote RETURN requests with CHANNEL option
DFHPROG	310	PCRTNCLD	Length of data in the containers of all remote RETURN CHANNEL commands
DFHSOCK	241	SOIOWTT	Socket I/O wait time
DFHSOCK	242	SOBYENCT	Bytes encrypted for secure socket
DFHSOCK	243	SOBYDECT	Bytes decrypted for secure socket
DFHSOCK	245	TCPSRVCE	TCP/IP service name
DFHSOCK	246	PORTNUM	TCP/IP service port number
DFHSOCK	288	ISALLOCT	Number of allocate requests for IPCONNs
DFHSOCK	289	SOEXTRCT	Socket extract request count
DFHSOCK	290	SOCNPSCT	Create nonpersistent socket request count
DFHSOCK	291	SOCPSCT	Create persistent socket request count
DFHSOCK	292	SONPSHWM	Nonpersistent socket high-water mark
DFHSOCK	293	SOPSHWM	Persistent socket high-water mark
DFHSOCK	294	SORCVCT	Socket receive request count
DFHSOCK	295	SOCHRIN	Socket characters received
DFHSOCK	296	SOSENDCT	Socket send request count
DFHSOCK	297	SOCHROUT	Socket characters sent
DFHSOCK	298	SOTOTCT	Socket total request count
DFHSOCK	299	SOOIOWTT	MAXSOCKETS wait time
DFHSOCK	300	ISIOWTT	Elapsed time a user task waited for control at this end of an IPCONN
DFHSOCK	301	SOMSGIN1	Inbound socket receive request count
DFHSOCK	302	SOCHRIN1	Inbound socket characters received
DFHSOCK	303	SOMSGOU1	Inbound socket send request count
DFHSOCK	304	SOCHROU1	Inbound socket characters sent
DFHSOCK	305	ISIPICNM	Name of the IPCONN whose TCPIPSERVICE attached the user task
DFHSOCK	318	CLIPADDR	IP address of the originating client or Telnet client
DFHSOCK	319	ISALWTT	IPIC session allocation wait time

Table 13. Data groups and fields that can be excluded or included (continued)

Group Name	Field Id	Field Name	Description
DFHSOCK	320	SOCIPHER	Cipher suite code for SSL handshake for use on the inbound connection
DFHSOCK	330	CLIPPORT	Port number of the client or Telnet client
DFHSTOR	33	SCUSRHWM	User storage high-water mark (UDSA)
DFHSTOR	54	SCUGETCT	User storage get count (UDSA)
DFHSTOR	87	PCSTGHWM	Program storage high-water mark
DFHSTOR	95	SCUSRSTG	User storage occupancy (bytes-ms) (UDSA)
DFHSTOR	105	SCUGETCT	User storage get count above 16 MB (EUDSA)
DFHSTOR	106	SCUSRHWM	User storage high-water mark above 16 MB (EUDSA)
DFHSTOR	107	SCUSRSTG	User storage occupancy (bytes-ms) above 16 MB (EUDSA)
DFHSTOR	108	PC24BHWM	Program storage high-water mark below 16 MB
DFHSTOR	116	SC24CHWM	User storage high-water mark below 16 MB (CDSA)
DFHSTOR	117	SCCGETCT	User storage get count below 16 MB (CDSA)
DFHSTOR	118	SC24COCC	User storage occupancy (bytes-ms) below 16 MB (CDSA)
DFHSTOR	119	SC31CHWM	User storage high-water mark above 16 MB (ECDSA)
DFHSTOR	120	SCCGETCT	User storage get count above 16 MB (ECDSA)
DFHSTOR	121	SC31COCC	User storage occupancy (bytes-ms) above 16 MB (ECDSA)
DFHSTOR	122	PC31RHWM	Program storage high-water mark (ERDSA)
DFHSTOR	139	PC31AHWM	Program storage high-water mark above 16 MB
DFHSTOR	142	PC31CHWM	Program storage high-water mark (ECDSA)
DFHSTOR	143	PC24CHWM	Program storage high-water mark (CDSA)
DFHSTOR	144	SC24SGCT	Shared storage count of getmain requests below 16 MB (CDSA and SDSA)
DFHSTOR	145	SC24GSHR	Shared storage bytes allocated by using a getmain request below 16 MB (CDSA and SDSA)
DFHSTOR	146	SC24FSHR	Shared storage released by using a freemain request below 16 MB (CDSA and SDSA)
DFHSTOR	147	SC31SGCT	Shared storage count of getmain requests above 16 MB (ECDSA and ESDSA)
DFHSTOR	148	SC31GSHR	Shared storage bytes allocated by using a getmain request above 16 MB (ECDSA and ESDSA)
DFHSTOR	149	SC31FSHR	Shared storage bytes released by using a freemain request above 16 MB (ECDSA and ESDSA)
DFHSTOR	160	PC24SHWM	Program storage high-water mark (SDSA)
DFHSTOR	161	PC31SHWM	Program storage high-water mark (ESDSA)
DFHSTOR	162	PC24RHWM	Program storage high-water mark (RDSA)
DFHSTOR	441	SC64CGCT	User storage get count above the bar (GCDSA)

Table 13. Data groups and fields that can be excluded or included (continued)

Group Name	Field Id	Field Name	Description
DFHSTOR	442	SC64CHWM	User storage high-water mark above the bar (GCDSA)
DFHSTOR	443	SC64UGCT	User storage get count above the bar (GUDSA)
DFHSTOR	444	SC64UHWM	User storage high-water mark above the bar (GUDSA)
DFHSTOR	445	SC64SGCT	Shared storage get count above the bar (GCDSA or GUDSA)
DFHSTOR	446	SC64GSHR	Shared storage bytes allocated by using a GETMAIN request above the bar (GCDSA or GUDSA)
DFHSTOR	447	SC64FSHR	Shared storage bytes released by using a FREEMAIN request above the bar (GCDSA or GUDSA)
DFHSYNC	60	SPSYNCCT	Sync point count
DFHSYNC	173	SYNCTIME	Sync point elapsed time
DFHSYNC	177	SRVSYWTT	CFDT server sync point wait time
DFHSYNC	196	SYNCDLY	Sync point delay time
DFHSYNC	199	OTSINDWT	OTS indoubt wait time
DFHTASK	7	USRDISPT	User task dispatch time
DFHTASK	8	USRCPUT	User task CPU time
DFHTASK	14	SUSPTIME	User task suspend time
DFHTASK	31	TRANNUM	Task number
DFHTASK	59	ICPUINCT	IC put/initiate count
DFHTASK	64	TASKFLAG	Error flag field
DFHTASK	65	ICSTACCT	Number of local START requests with CHANNEL option
DFHTASK	66	ICTOTCT	IC total count
DFHTASK	82	TRNGRPID	Transaction group ID
DFHTASK	97	NETUOWPX	Network name of the originating terminal or system
DFHTASK	98	NETUOWSX	Unit-of-work ID on the originating system
DFHTASK	102	DISPWTT	User task wait-for-dispatch time
DFHTASK	109	TRANPRI	Transaction priority
DFHTASK	123	GNQDELAY	Task global ENQ delay time
DFHTASK	124	BRDGTRAN	3270 Bridge transaction ID
DFHTASK	125	DSPDELAY	First dispatch delay time
DFHTASK	126	TCLDELAY	First dispatch delay time due to TRANCLASS
DFHTASK	127	MXTDELAY	First dispatch delay due to MXT
DFHTASK	128	LMDELAY	Lock manager delay time
DFHTASK	129	ENQDELAY	Task ENQ delay time
DFHTASK	132	RMUOWID	Recovery manager unit-of-work ID
DFHTASK	163	FCTYNAME	Transaction facility name
DFHTASK	164	TRANFLAG	Transaction flags

Table 13. Data groups and fields that can be excluded or included (continued)

Group Name	Field Id	Field Name	Description
DFHTASK	166	TCLSNAME	Transaction class name
DFHTASK	170	RMITIME	Resource Manager Interface elapsed time
DFHTASK	171	RMISUSP	Resource Manager Interface suspend time
DFHTASK	181	WTEXWAIT	EXEC CICS WAIT EXTERNAL wait time
DFHTASK	182	WTCEWAIT	EXEC CICS WAITCICS and WAIT EVENT wait time
DFHTASK	183	ICDELAY	Interval Control delay time
DFHTASK	184	GVUPWAIT	Dispatch Wait wait time
DFHTASK	190	RRMSURID	RRMS/MVS unit-of-recovery ID (URID)
DFHTASK	191	RRMSWAIT	RRMS/MVS wait time
DFHTASK	192	RQRWAIT	Request receiver wait time
DFHTASK	193	RQPWAIT	Request processor wait time
DFHTASK	194	OTSTID	OTS Transaction ID (TID)
DFHTASK	195	RUNTRWTT	CICS BTS run process/activity synchronous wait time
DFHTASK	247	DSCHMDLY	Change-TCB mode delay time
DFHTASK	249	QRMODDLY	User task QR TCB wait-for-dispatch time
DFHTASK	250	MAXOTDLY	CICS L8 and L9 mode open TCB delay time
DFHTASK	251	TCBATTCT	CICS TCB attach count
DFHTASK	252	DSTCBHWM	User task peak open TCB count
DFHTASK	253	JVMTIME	CICS JVM elapsed time
DFHTASK	254	JVMSUSP	CICS JVM suspend time
DFHTASK	255	QRDISPT	User task QR TCB dispatch time
DFHTASK	256	QRCPUT	User task QR TCB CPU Time
DFHTASK	257	MSDISPT	User task MS TCB dispatch time
DFHTASK	258	MSCPUT	User task MS TCB CPU Time
DFHTASK	259	L8CPUT	User task L8 TCB CPU Time
DFHTASK	261	S8CPUT	User task S8 TCB CPU Time
DFHTASK	262	KY8DISPT	User task key 8 TCB dispatch time
DFHTASK	263	KY8CPUT	User task key 8 TCB CPU time
DFHTASK	264	KY9DISPT	User task key 9 TCB dispatch time
DFHTASK	265	KY9CPUT	User task key 9 TCB CPU time
DFHTASK	266	L9CPUT	User task L9 TCB CPU time
DFHTASK	268	DSTCBMWT	User task TCB mismatch wait time
DFHTASK	269	RODISPT	User task RO TCB dispatch time
DFHTASK	270	ROCPUT	User task RO TCB CPU time
DFHTASK	271	X8CPUT	User task X8 TCB CPU time
DFHTASK	272	X9CPUT	User task X9 TCB CPU time

Table 13. Data groups and fields that can be excluded or included (continued)

Group Name	Field Id	Field Name	Description
DFHTASK	273	JVMITIME	CICS JVM initialize elapsed time
DFHTASK	275	JVMRTIME	CICS JVM reset elapsed time
DFHTASK	279	DSMMSCWT	MVS storage constraint delay time
DFHTASK	281	MAXSTDLY	CICS SSL TCB delay time
DFHTASK	282	MAXXTDLY	CICS XP TCB delay time
DFHTASK	283	MAXTTDLY	CICS JVM server thread TCB delay time
DFHTASK	285	PTPWAIT	3270 bridge partner wait time
DFHTASK	345	ICSTACDL	Length of data in the containers of all locally-executed START CHANNEL requests
DFHTASK	346	ICSTRCCT	Number of interval control START CHANNEL requests to be executed on remote systems
DFHTASK	347	ICSTRCDL	Length of data in the containers of all remotely-executed START CHANNEL requests
DFHTASK	348	ROMODDLY	User task RO TCB wait-for-dispatch time
DFHTASK	349	SOMODDLY	User task SO TCB wait-for-dispatch time
DFHTASK	400	T8CPUT	User task T8 TCB CPU time
DFHTASK	401	JVMTHDWT	JVMSERVER thread wait time. This does not apply to Liberty JVM servers.
DFHTASK	429	DSAPTHWT	Dispatcher allocate pthread wait time
DFHTASK	430	CECMCHTP	CEC machine type
DFHTASK	431	CECMDLID	CEC model number
DFHTASK	432	LPARNAME	The name, in EBCDIC, of the logical partition (LPAR) on the processor where the CICS region is running
DFHTASK	433	MAXTASKS	MXT or MAXTASKS for CICS region at user task attach
DFHTASK	434	CURTASKS	Current tasks in CICS region at user task attach
DFHTASK	436	CPUTONCP	Task processor time on a standard processor
DFHTASK	437	OFFLCPUT	Task processor time eligible for offload to a specialty processor
DFHTASK	451	ACAPPLNM	The 64-character name of the application in the application context data
DFHTASK	452	ACPLATNM	The 64-character name of the platform in the application context data
DFHTASK	453	ACMAJVER	The major version of the application in the application context data, expressed as a 4-byte binary value
DFHTASK	454	ACMINVER	The minor version of the application in the application context data, expressed as a 4-byte binary value
DFHTASK	455	ACMICVER	The micro version of the application in the application context data, expressed as a 4-byte binary value
DFHTASK	456	ACOPERNM	The 64-character name of the operation in the application context data

Table 13. Data groups and fields that can be excluded or included (continued)

Group Name	Field Id	Field Name	Description
DFHTASK	470	ASTOTCT	The total number of EXEC CICS asynchronous API commands that have been issued by the user task. Includes **RUN TRANSID**, **FETCH CHILD**, and **FETCH ANY** commands.
DFHTASK	471	ASRUNCT	The number of **EXEC CICS RUN TRANSID** commands that have been issued by the user task
DFHTASK	472	ASFTCHCT	The number of **EXEC CICS FETCH CHILD** and **EXEC CICS FETCH ANY** commands that have been issued by the user task
DFHTASK	473	ASFREECT	The number of **EXEC CICS FREE CHILD** commands that have been issued by the user task
DFHTASK	475	ASFTCHWT	The elapsed time that the user task waited for a child task as a result of issuing an **EXEC CICS FETCH CHILD** or **EXEC CICS FETCH ANY** command which was not completed
DFHTASK	476	ASRNATWT	The elapsed time that the user task was delayed as a result of asynchronous child task limits managed by the asynchronous services domain
DFHTEMP	11	TSIOWTT	TS I/O wait time
DFHTEMP	44	TSGETCT	TS get count
DFHTEMP	46	TSPUTACT	TS put auxiliary count
DFHTEMP	47	TSPUTMCT	TS put main count
DFHTEMP	92	TSTOTCT	TS total count
DFHTEMP	178	TSSHWAIT	Shared TS I/O wait time
DFHTEMP	460	TSGETSCT	TS get shared count
DFHTEMP	461	TSPUTSCT	TS put shared count
DFHTERM	9	TCIOWTT	TC I/O wait time
DFHTERM	34	TCMSGIN1	TC principal facility input messages
DFHTERM	35	TCMSGOU1	TC principal facility output messages
DFHTERM	67	TCMSGIN2	TC alternate facility input messages
DFHTERM	68	TCMSGOU2	TC alternate facility output messages
DFHTERM	69	TCALLOCT	TC allocate count
DFHTERM	83	TCCHRIN1	TC principal facility CHARS input
DFHTERM	84	TCCHROU1	TC principal facility CHARS output
DFHTERM	85	TCCHRIN2	TC alternate facility CHARS input
DFHTERM	86	TCCHROU2	TC alternate facility CHARS output
DFHTERM	100	IRIOWTT	IR I/O wait time
DFHTERM	111	LUNAME	z/OS Communications Server terminal LU name
DFHTERM	133	LU61WTT	TC I/O wait time - LU6.1
DFHTERM	134	LU62WTT	TC I/O wait time - LU6.2

Table 13. Data groups and fields that can be excluded or included (continued)

Group Name	Field Id	Field Name	Description
DFHTERM	135	TCM62IN2	TC alternate facility input messages - LU6.2
DFHTERM	136	TCM62OU2	TC alternate facility output messages - LU6.2
DFHTERM	137	TCC62IN2	TC alternate facility CHARS input - LU6.2
DFHTERM	138	TCC62OU2	TC alternate facility CHARS output - LU6.2
DFHTERM	165	TERMINFO	Terminal information
DFHTERM	169	TERMCNM	Terminal session connection name
DFHTERM	197	NETID	Network qualified name network ID
DFHTERM	198	RLUNAME	Network qualified name network name
DFHTERM	343	TCALWTT	MRO, LU6.1, and LU6.2 session allocation wait time
DFHWEBB	224	WBREADCT	Web read request count
DFHWEBB	225	WBWRITCT	Web write request count
DFHWEBB	231	WBRCVCT	Web receive request count
DFHWEBB	232	WBCHRIN	Web characters received
DFHWEBB	233	WBSENDCT	Web send request count
DFHWEBB	234	WBCHROUT	Web characters sent
DFHWEBB	235	WBTOTCT	Web total request count
DFHWEBB	236	WBREPRCT	Web header or formfield read request count
DFHWEBB	237	WBREPWCT	Web header or formfield write request count
DFHWEBB	238	WBEXTRCT	Web extract request count
DFHWEBB	239	WBBRWCT	Web browse request count
DFHWEBB	331	WBREDOCT	Web header read request count, client
DFHWEBB	332	WBWRTOCT	Web header write request count, client
DFHWEBB	333	WBRCVIN1	Web client receive or converse request count
DFHWEBB	334	WBCHRIN1	Web characters received, client
DFHWEBB	335	WBSNDOU1	Web client send or converse request count
DFHWEBB	336	WBCHROU1	Web characters sent, client
DFHWEBB	337	WBPARSCT	Web client parse URL request count
DFHWEBB	338	WBRWOCT	Web client browse request count
DFHWEBB	340	WBIWBSCT	EXEC CICS INVOKE SERVICE or WEBSERVICE request count
DFHWEBB	341	WBREPRDL	Web header or formfield read byte count
DFHWEBB	342	WBREPWDL	Web header or formfield write byte count
DFHWEBB	380	WBURIMNM	URIMAP resource name
DFHWEBB	381	WBPIPLNM	PIPELINE resource name
DFHWEBB	382	WBATMSNM	ATOMSERVICE resource name
DFHWEBB	383	WBSVCENM	WEBSERVICE resource name

Table 13. Data groups and fields that can be excluded or included (continued)

Group Name	Field Id	Field Name	Description
DFHWEBB	384	WBSVOPNM	WEBSERVICE operation name
DFHWEBB	385	WBPROGNM	PROGRAM resource name
DFHWEBB	386	WBSFCRCT	SOAPFAULT create request count
DFHWEBB	387	WBSFTOTCT	Total SOAPFAULT request count
DFHWEBB	388	WBISSFCT	EXEC CICS INVOKE SERVICE SOAP faults
DFHWEBB	390	WBSREQBL	SOAP request body length
DFHWEBB	392	WBSRSPBL	SOAP response body length
DFHWEBB	412	MLXSSTDL	Total incoming converted document length
DFHWEBB	413	MLXMLTCT	EXEC CICS TRANSFORM request count
DFHWEBB	420	WSACBLCT	WS-Addressing context build request count
DFHWEBB	421	WSACGTCT	WS-Addressing context get request count
DFHWEBB	422	WSAEPCCT	WS-Addressing endpoint reference (EPR) create request count
DFHWEBB	423	WSATOTCT	WS-Addressing total request count
DFHWEBB	424	WBJSNRQL	For JSON webservice applications, the JSON message request length.
DFHWEBB	425	WBJSNRPL	For JSON webservice applications, the JSON message response length.

INCLUDE=(*m1*[,...])

Code this operand to enable one or more CICS fields to be reported by the monitoring facility. By default, all documented performance class fields are reported, so this operand is relevant only if you code the EXCLUDE operand in the same macro.

The fields that are eligible to be coded for inclusion on this operand are the same as those that are eligible for exclusion. (See the description of the EXCLUDE operand.) Each field has a numeric field identifier associated with it. To include a field, you code *m1* as the numeric identifier of the field. You can code multiple numeric identifiers.

Note: Do not code leading zeros on numeric identifiers. Do not code numeric identifiers of fields that are ineligible for exclusion, and that are therefore included by default.

The EXCLUDE operand is always honored first. The INCLUDE operand, if coded, then overrides some of its effects. For example, the following code secures the collection and reporting of file control PUTs and the total number of file control requests, but excludes the file control browse count and other file control fields:

```
EXCLUDE=DFHFILE,
INCLUDE=(37,93)
```

If you want to exclude the majority of fields, but include a few, you can use the operands in the way shown in the following example:

```
EXCLUDE=ALL,
INCLUDE=(DFHTERM,97,98)
```

This is more convenient than coding individually all the fields you want to exclude.

CICSTS54.SDFHSAMP provides the following sample monitoring control tables:

- For terminal-owning region (TOR): DFHMCTT$

- For application-owning region (AOR): DFHMCTA$
- For an application-owning region (AOR) with DBCTL: DFHMCTD$
- For file-owning region (FOR): DFHMCTF$

These samples show how to use the EXCLUDE and INCLUDE operands to reduce the size of the performance class record in order to reduce the volume of data written by CICS to SMF.

DFHMCT examples

These examples show the use of the DFHMCT TYPE=INITIAL, DFHMCT TYPE=RECORD and DFHMCT TYPE=EMP macros to create the control section for the monitoring control table, specify user event monitoring points, and exclude system-defined fields.

Figure 8 on page 386 demonstrates the coding for the control statements in the DFHMCT TYPE=INITIAL macro. In the example, the suffix C2 is specified, so the name of this monitoring control table is DFHMCTC2. The other parameters are set to their default values.

```
DFHMCT   TYPE=INITIAL,SUFFIX=C2,COMPRESS=YES,       *
        APPLNAME=NO,RMI=NO,FILE=8,DPL=0,TSQUEUE=8
DFHMCT   TYPE=FINAL
         END
```

Figure 8. Example: DFHMCT TYPE=INITIAL

Figure 9 on page 386 demonstrates the coding to create a monitoring control table (MCT) for two user event monitoring points (EMPs).

```
DFHMCT   TYPE=INITIAL,SUFFIX=MF,COMPRESS=YES
DFHMCT   TYPE=EMP,                                  *
         ID=180,                                    *
         CLASS=PERFORM,                             *
         PERFORM=(SCLOCK(1),ADDCNT(2,1))
DFHMCT   TYPE=EMP,                                  *
         ID=181,                                    *
         CLASS=PERFORM,                             *
         PERFORM=PCLOCK(1)
DFHMCT   TYPE=FINAL
         END
```

Figure 9. Example: DFHMCT TYPE=EMP, user event monitoring points

Figure 10 on page 387 demonstrates the coding to create a monitoring control table (MCT) that excludes specific data fields from the performance data record. If you do not have any applications that use certain CICS functions, you can safely exclude the performance data groups and fields relating to those functions. The example shows DFHMCT TYPE=RECORD macros to exclude fields for these functions:

Front End Programming Interface (FEPI)

```
DFHMCT   TYPE=RECORD,CLASS=PERFORM,                 *
         EXCLUDE=(DFHFEPI)
```

Business transaction services (BTS)

```
DFHMCT   TYPE=RECORD,CLASS=PERFORM,                 *
         EXCLUDE=(DFHCBTS,195,196)
```

Support for Java applications

```
DFHMCT   TYPE=RECORD,CLASS=PERFORM,                 *
         EXCLUDE=(253,254,273,275)
```

```
DFHMCT   TYPE=INITIAL,SUFFIX=CB,COMPRESS=YES,       *
        APPLNAME=NO,RMI=NO,FILE=16,TSQUEUE=12
DFHMCT   TYPE=RECORD,CLASS=PERFORM,                 *
        EXCLUDE=(DFHFEPI)
DFHMCT   TYPE=RECORD,CLASS=PERFORM,                 *
        EXCLUDE=(DFHCBTS,195,196)
DFHMCT   TYPE=RECORD,CLASS=PERFORM,                 *
        EXCLUDE=(253,254,273,275)
DFHMCT   TYPE=FINAL
         END
```

Figure 10. Example: DFHMCT TYPE=RECORD, excluding fields

PLT: program list table

A program list table (PLT) specifies programs that you want to run during CICS startup and shutdown, and groups of programs that you want to enable and disable together.

You may want to generate several PLTs for one or more of the following reasons:

- To specify a list of programs that you want to be executed in the second and third initialization stages of CICS startup. For more detail about the initialization stages, see Using post-initialization (PLTPI) programs. For programming information about restrictions on using programs in the initialization stages, see Writing initialization programs. The selected list should be specified at initialization time by the PLTPI=*xx* system initialization parameter, where *xx* is the suffix of the PLT that contains the required list of programs.

  For convenience, the list of programs selected for execution during initialization is referred to as the 'PLTPI' list.
- To specify a list of programs that you want to be executed during the first and second quiesce stages of controlled shutdown. The selected list should be specified at initialization time by the PLTSD=*xx* system initialization parameter, where *xx* is the suffix of the PLT that contains the required list of programs.

  The PLT specified in the PLTSD system initialization parameter can be overridden at shutdown time by the PLT option in the CEMT PERFORM SHUTDOWN command.

  The shutdown PLT is normally loaded as CICS is being shutdown. However, it is possible to use the same PLT for both initialization and shutdown, and under these circumstances the PLT is loaded during initialization and CICS does not need to reload it during shutdown. If this is the case and the PLT is updated while CICS is operational, a CEMT SET PROGRAM NEWCOPY command must be issued for the PLT to ensure that the updated version is used when CICS is shutdown.

  For convenience, the list of programs selected for execution during shutdown is referred to as the 'PLTSD' list.
- To specify a list of programs that you want to have enabled or disabled as a group by a main terminal ENABLE or DISABLE command. This use of PLTs means that a main terminal operator can enable or disable a set of programs with just one command, instead of using a separate command for each program.

Any number of PLTs can be generated for the purposes described here, provided that:

1. Each PLT has a unique suffix
2. Each program named in a PLT either has a program resource definition entry in the CSD file, or is capable of being autoinstalled (that is, the appropriate system initialization parameters have been specified for program autoinstall).

   Note: PLTs should not be defined as programs in the CSD.

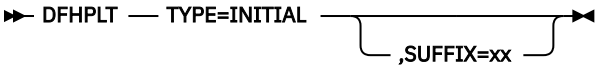
First-phase PLT programs must be placed in the DFHRPL concatenated data sets, but second-phase PLT programs can be placed in a dynamic LIBRARY concatenation. However, CICS scans the LPA for phase 1 PLTPI programs if they are not already installed.

The following macros are available to define the PLT entries:

- Control section: DFHPLT TYPE=INITIAL
- Entries in program list table: DFHPLT TYPE=ENTRY
- End of Program List Table: DFHPLT TYPE=FINAL (see TYPE=FINAL (end of table))

Control section: DFHPLT TYPE=INITIAL

The DFHPLT TYPE=INITIAL macro establishes the entry point and start address of the PLT being defined.



For general information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see TYPE=INITIAL (control section).

TYPE=INITIAL
Generates the PLT control section.

Note that the CSD file must define a program entry for each PLT generated.

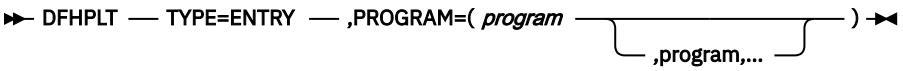
SUFFIX=xx
Code this with the suffix character(s) that uniquely identify this particular table.

Note: The PLT suffix is referenced by:

- CEMT {INQUIREvSET} PROGRAM CLASS(*suffix*)
- CEMT or EXEC CICS PERFORM SHUTDOWN PLT(*suffix*)
- System initialization parameters PLTPI and PLTSD keywords.

Entries in program list table: DFHPLT TYPE=ENTRY

Use the DFHPLT TYPE=ENTRY macro to specify an entry in the program list table (PLT).

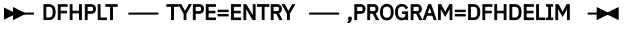


TYPE=ENTRY
Indicates that one or more program names are to be listed in this table.

Note: As shown here, a TYPE=ENTRY macro is also needed to specify the PROGRAM=DFHDELIM entry.

PROGRAM=program
Code this with a program name of up to eight characters. Each program must either have a definition in the CSD file or must be capable of being autoinstalled (that is, the appropriate system initialization parameters must be specified for program autoinstall). Undefined programs before the DFHDELIM statement are system autoinstalled.

For PLTPI and PLTSD lists, only initial programs should be named; other programs that are linked to by initial programs should not be listed (but must be defined or be capable of being autoinstalled). For programming information about restrictions on using PLT programs during initialization, see Writing initialization programs.



PROGRAM=DFHDELIM
Code this to delimit the programs to run in the first or second passes of PLTPI or PLTSD. The DFHDELIM entry is not a program: it serves as a delimiter only.

Note that:

- Programs listed before the PROGRAM=DFHDELIM entry in a PLTPI are executed during the second stage of initialization. These are to enable user exit programs needed during recovery. Define the user exit programs in the CSD file, otherwise CICS might not be able to access them after CICS initialization is complete, for example in **EXEC CICS DISABLE** commands. However, note that the properties defined by RDO have no effect during the second stage of initialization.
- Programs listed after the PROGRAM=DFHDELIM entry in a PLTPI are executed during the third stage of initialization. If these programs are used to enable user exits, the user exit programs must also be defined in the CSD file or must be capable of being autoinstalled.
- Programs listed before the PROGRAM=DFHDELIM entry in a PLTSD are executed during the first quiesce stage of shutdown.
- Programs listed after the PROGRAM=DFHDELIM entry in a PLTSD are executed during the second quiesce stage of shutdown.

Second stage initialization and second stage quiesce PLT programs do not require program resource definitions. If they are not defined, they are system autoinstalled (irrespective of the program autoinstall system initialization parameters). This means that the autoinstall exit is not called to allow the definition to be modified. The programs are defined with the following attributes:

```
LANGUAGE(ASSEMBLER)  STATUS(ENABLED)  CEDF(NO)
DATALOCATION(BELOW)  EXECKEY(CICS)
EXECUTIONSET(FULLAPI)
```

As a result, system autoinstalled programs have a default CONCURRENCY setting of QUASIRENT, and a default API setting of CICSAPI.

- For those threadsafe PLT programs that are defined with the OPENAPI value for the API attribute, or are C or C++ programs compiled with the XPLINK compiler option, provide an appropriate resource definition. Alternatively, for Language Environment conforming programs, use the CICSVAR runtime option to set the appropriate CONCURRENCY and API values. See Defining runtime options for Language Environment.

Third stage initialization and first stage quiesce PLT programs can be defined using program autoinstall, depending upon the program autoinstall system initialization parameters. If program autoinstall is not used, these programs must have program resource definitions in the CSD file.

DFHPLT example

The coding required to generate a PLT can be clarified with an example.

Note: The contents of PLT files must be written observing the same rules as apply to Assembler coding. Failure to follow these rules might produce unexpected results. For more information, see HLASM Language Reference.

Figure 11 on page 390 and Figure 12 on page 390 illustrate the coding required to generate a PLT.

```
*
* LIST OF PROGRAMS TO BE EXECUTED SEQUENTIALLY DURING SYSTEM
* INITIALIZATION.
* REQUIRED SYSTEM INITIALIZATION PARAMETER: PLTPI=I1
*
   DFHPLT TYPE=INITIAL,SUFFIX=I1
*
*  The following programs are run in the first pass of PLTPI
*
   DFHPLT TYPE=ENTRY,PROGRAM=TRAQA  EXECUTED DURING 2ND INIT. PHASE
   DFHPLT TYPE=ENTRY,PROGRAM=TRAQB  (PROGRAMS SHOULD ALSO BE DEFINED
   DFHPLT TYPE=ENTRY,PROGRAM=TRAQC  BY RDO)
*
   DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
*
*
*  The following programs are run in the second pass of PLTPI
*
   DFHPLT TYPE=ENTRY,PROGRAM=TRASA  EXECUTED DURING 3RD INIT. PHASE
   DFHPLT TYPE=ENTRY,PROGRAM=TRASB  (PROGRAMS MUST ALSO BE DEFINED
   DFHPLT TYPE=ENTRY,PROGRAM=TRASC  BY RDO)
   DFHPLT TYPE=FINAL
*
   END
```

Figure 11. PLTPI program list table: example

```
*
*
* LIST OF PROGRAMS TO BE EXECUTED SEQUENTIALLY DURING SYSTEM
* TERMINATION
* REQUIRED SYSTEM INITIALIZATION PARAMETER: PLTSD=T1
*
   DFHPLT TYPE=INITIAL,SUFFIX=T1
*
*  The following programs are run in the 1st pass of PLTSD
*
*
   DFHPLT TYPE=ENTRY,PROGRAM=TRARA  EXECUTED DURING 1st QUIESCE PHASE
   DFHPLT TYPE=ENTRY,PROGRAM=TRARB  (PROGRAMS MUST ALSO BE DEFINED
   DFHPLT TYPE=ENTRY,PROGRAM=TRARC  BY RDO)
*
   DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
*
*
*  The following programs are run in the 2nd pass of PLTSD
*
   DFHPLT TYPE=ENTRY,PROGRAM=TRAFA  EXECUTED DURING 2nd QUIESCE PHASE
   DFHPLT TYPE=ENTRY,PROGRAM=TRAFB  (PROGRAMS MUST ALSO BE DEFINED
*                                   BY RDO)
   DFHPLT TYPE=FINAL
*
   END
```

Figure 12. PLTSD program list table: example

RST: recoverable service table

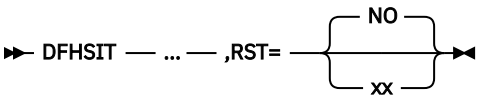
Note on terminology: DBCTL refers to the CICS-IMS/ESA® DBCTL (database control) interface.

The recoverable service table (RST) is used for CICS DBCTL XRF (extended recovery facility) support. It contains a description of the DBCTL configuration. On detection of a DBCTL failure, the **active** CICS system uses the RST together with the MVS subsystem VERIFY to determine the existence of a suitable alternative DBCTL subsystem. The **alternate** CICS system uses the RST to check for the presence of a DBCTL subsystem. For security reasons, the RST should be link-edited into a library authorized using APF. The RST is not loaded as part of the CICS nucleus.

You can code the following macros in a recoverable service table:

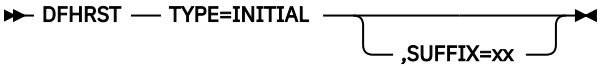
- DFHRST TYPE=INITIAL establishes the control section.
- DFHRST TYPE=RSE specifies the start of a recoverable service element (RSE). An RSE consists of a (nonempty) set of identifiers of equivalent DBCTL subsystems, the CICS applids associated with these DBCTL subsystems, and the application identifiers of the CICS systems which use the DBCTL subsystems.
- DFHRST TYPE=SUBSYS specifies one of the DBCTL subsystems in an RSE.
- DFHRST TYPE=FINAL concludes the RST (see TYPE=FINAL (end of table)).

The RST to be used by the system must be defined in the system initialization table by the following system initialization parameter:



Control section: DFHRST TYPE=INITIAL

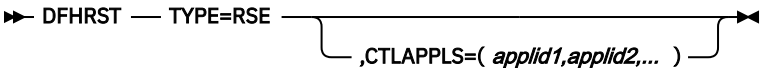
The DFHRST TYPE=INITIAL macro establishes the recoverable service table control section.



For general information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see TYPE=INITIAL (control section).

Recoverable service elements: DFHRST TYPE=RSE

The DFHRST TYPE=RSE macro contains information about a recoverable service element and the CICS systems expected to connect to the DBCTL subsystems within the recoverable service element.



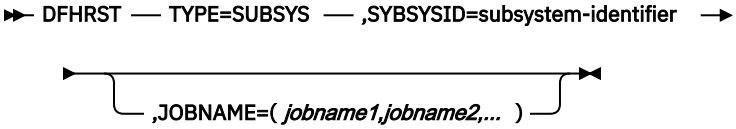
TYPE=RSE
Indicates the start of a set of identifiers of equivalent DBCTL systems. Equivalent DBCTL subsystems have the same database (DB) name.

CTLAPPLS=(applid1,applid2,....)
Specifies the application identifiers of the CICS systems that are authorized to restart the DBCTL subsystems in the RSE.

DBCTL subsystems: DFHRST TYPE=SUBSYS

The DFHRST TYPE=SUBSYS macro contains information about a specific DBCTL subsystem within an RSE.

For each subsystem in an RSE there must be a DFHRST TYPE=SUBSYS macro.



TYPE=SUBSYS
Defines one of the DBCTL subsystems in an RSE.

SUBSYSID=subsystem-identifier
Code this with the 4-character name of the DBCTL subsystem-identifier. This identifier must be unique within the RST.

JOBNAME=(jobname1,jobname2,....)
Code this with the **MVS JOBNAME(S)** associated with the DBCTL subsystem identified in the SUBSYSID parameter statement. This is a form of security check. If CICS needs to cancel the DBCTL subsystem, it is authorized to do so only if the appropriate MVS jobname associated with the DBCTL subsystem is one of those specified by the **JOBNAME** parameter.

DFHRST example

The coding required to generate an RST can be clarified with an example.

Figure 13 on page 392 illustrates the coding required to generate an RST.

```
DFHRST   TYPE=INITIAL
         ,SUFFIX=K1
DFHRST   TYPE=RSE
         ,CTLAPPLS=(applid1,applid2,applid3)
DFHRST   TYPE=SUBSYS
         ,SUBSYSID=CTL1
         ,JOBNAME=(job1,job2,job3,job4)
DFHRST   TYPE=SUBSYS
         ,SUBSYSID=CTL2
         ,JOBNAME=(job5,job6,job7,job8)
DFHRST   TYPE=FINAL
END
```

Figure 13. Recoverable service table: example

SRT: system recovery table

The system recovery table (SRT) contains a list of codes for abends that CICS intercepts. After it intercepts one, CICS attempts to remain operational by causing the offending task to abend.

You can modify the default recovery action by writing your own recovery program. You do this by means of the XSRAB global user exit point within the System Recovery Program (SRP). (See System recovery program exit XSRAB for programming information about the XSRAB exit.)

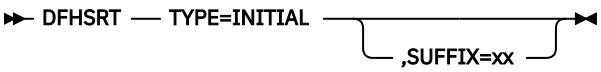
Note that recovery is attempted if a user task (but not a system task) is in control at the time the abend occurs.

The following macros may be coded in a system recovery table:

- DFHSRT TYPE=INITIAL establishes the control section
- DFHSRT TYPE=SYSTEM|USER specifies the abend codes that are to be handled
- DFHSRT TYPE=FINAL concludes the SRT (see TYPE=FINAL (end of table))

Control section: DFHSRT TYPE=INITIAL

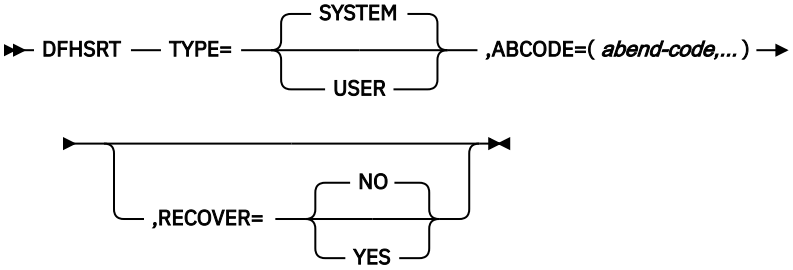
The DFHSRT TYPE=INITIAL macro generates the system recovery table control section.



For general information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see TYPE=INITIAL (control section).

Abend codes: DFHSRT TYPE=SYSTEM|USER

The DFHSRT TYPE=SYSTEM and TYPE=USER macros specify system and user abend codes that are to be intercepted by CICS.



TYPE={SYSTEM|USER}
Indicates the type of abend code to be intercepted.

SYSTEM
The abend code is an operating system abend code corresponding to an MVS S*xxx* abend code.

USER
The abend code is a user (including CICS) abend code corresponding to an MVS U*nnnn* abend code.

ABCODE=(abend-code,...)
Code this with the abend code (or codes) to be intercepted. If you specify a single abend code, parentheses are not required.

If you code TYPE=SYSTEM, this abend code must be three hexadecimal digits (*xxx*) representing the MVS system abend code S*xxx*.

If you code TYPE=USER, this abend code must be a decimal number (*nnnn*) representing the user part of the MVS abend code U*nnnn*. This is usually the same number as the CICS message that is issued before CICS tries to terminate abnormally.

RECOVER={YES|NO}
specifies whether codes are to be added to or removed from the SRT.

YES
Code this to add the specified codes to the SRT.

NO
Code this to remove the specified codes from the SRT.

Note:

1. CICS intercepts the following abend codes automatically and tries to recover:

```
001,002,013,020,025,026,030,032,033,034,035,
036,037,03A,03B,03D,052,053,067,0D3,0D4,0D5,
0D6,0D7,0D8,0E0,0F3,100,113,137,202,213,214,
237,283,285,313,314,337,400,413,437,513,514,
613,614,637,713,714,737,813,837,913,A13,A14,
B13,B14,B37,D23,D37,E37
```

Abend code 0F3 covers various machine check conditions. It also covers the Alternate Processor Retry condition that can occur only when running on a multiprocessor. CICS-supplied recovery code attempts to recover from instruction-failure machine checks on the assumption that they are not permanent. It also attempts to recover from Alternate Processor Retry conditions.

CICS will try to recover from these standard abend codes if you code the system recovery table (SRT) as:

```
DFHSRT  TYPE=INITIAL
DFHSRT  TYPE=FINAL
END
```

There is no need to list the standard codes individually.

2. If you want CICS to handle other errors, you can code the SRT as follows:

```
DFHSRT  TYPE=INITIAL
DFHSRT  TYPE=SYSTEM,or USER,
        ABCODE=(user or system codes),
        RECOVER=YES
DFHSRT  TYPE=FINAL
END
```

3. If you do not want CICS to try to recover after one or more of the standard abend codes occurs, specify the code(s) with **RECOVER=NO**, or without the **RECOVER** parameter.
4. CICS tries to recover if a user task (but not a system task) is in control at the time the abend occurs.

DFHSRT example

The coding required to generate an SRT can be clarified with an example.

Figure 14 on page 394 illustrates the coding required to generate a SRT.

```
DFHSRT TYPE=INITIAL,             *
       SUFFIX=K1
DFHSRT TYPE=SYSTEM,              *
       ABCODE=777,               *
       RECOVER=YES
DFHSRT TYPE=USER,
       ABCODE=(888,999),         *
       RECOVER=YES
DFHSRT TYPE=USER,                *
       ABCODE=020
DFHSRT TYPE=FINAL
END
```

Figure 14. System recovery table: example

TCT: terminal control table

Use the DFHTCT macro to define SNA logical units (LUs) that support logical device codes and sequential devices attached by BSAM.

A CICS system can communicate with terminals, sequential devices, logical units, and other systems. The TCT defines each of the devices in the configuration. Each TCT entry defines the optional and variable features of the device to CICS, and specifies the optional and variable features of CICS to be used.

CICS uses a telecommunication access method to communicate with terminals. This can be the z/OS Communications Server or (for sequential devices) BSAM; you can use one or both of these in your system.

An terminal can be a telecommunication device; for example, an IBM 3279 Color Display Station, or a subsystem such as an IBM 4700 Finance Communication System. Terminals can be local (channel-attached) or remote (link-attached).

You can use a sequential device to simulate a CICS terminal. You can define a card reader or punch, line printer, direct access storage device (disk drive), or magnetic tape drive as a sequential device.

A logical unit (LU) is a port through which a user of an SNA network gains access to the network facilities.

A system can be, for example, another CICS system, an IBM 8815 Scanmaster, an IBM Displaywriter, or an APPC/PC. Intercommunication with CICS systems can be:

- Between different processors (**intersystem communication** or **ISC**), using the LUTYPE 6.1 or LUTYPE 6.2 protocols, or using an intermediate system as an **indirect link**. (Intercommunication with non-CICS systems also uses ISC.)

- Within the same processor (**multiregion operation** or **MRO**), using interregion communication (IRC). (You can also use LUTYPE 6.2 ISC within the same processor.)

DFHTCT macro types

The DFHTCT macros you code depend on the device you are defining, and on the access method you are using.

You always start with one of these:

`DFHTCT TYPE=INITIAL,...` (See "Control section: DFHTCT TYPE=INITIAL" on page 396.)

There is a special macro to use when you assemble the TCT to migrate RDO-eligible definitions to the CSD file:

`DFHTCT TYPE=GROUP,...` (See "Migrating TCT definitions: DFHTCT TYPE=GROUP" on page 397.)

You can define your devices in any order you want. Each device needs one or more macros, and these sometimes have to be in a particular order. You are told when this is the case. The following table shows the macros you need for each type of device or system.

Table 14. DFHTCT macro types

Device or system	DFHTCT macros
Logical device codes.	`DFHTCT TYPE=LDC,...` `DFHTCT TYPE=LDCLIST,...`

Sequential devices.	`DFHTCT TYPE=SDSCI,...`
	`DFHTCT TYPE=LINE,...`
	`DFHTCT TYPE=TERMINAL,...`

Remote non-SNA LUs, for transaction routing.	`DFHTCT TYPE=REMOTE,...`
	or:
	`DFHTCT TYPE=REGION,...`
	`DFHTCT TYPE=TERMINAL,...`

At the very end of your macros you code:

```
DFHTCT TYPE=FINAL
END
```

This macro is described in TYPE=FINAL (end of table).

Notes:

SYSIDNT and TRMIDNT operands

CICS accepts both uppercase and lowercase characters for SYSIDNT and TRMIDNT, but the lowercase characters are not checked for duplication. Assembling a TCT containing lowercase SYSIDNT or TRMIDNT results in an MNOTE. If you want duplicate checking, use only uppercase characters for SYSIDNT and TRMIDNT.

Assembling the TCT

The assembly and link-edit of a TCT leads to the creation of two separate load modules. Assembly of a suffixed TCT (source name DFHTCT*xx*) produces a single text file. However, when this is link-edited into a load library, two members are created:

1. DFHTCT*xx*, which contains the non-RDO-eligible definitions in control block format
2. DFHRDTxx, which contains the RDO-eligible (SNA LU and system) definitions in RDO command format

This happens, **whether or not you intend to use RDO**. You need to be aware of the existence of these two tables if you copy or move assembled TCT tables between load libraries.

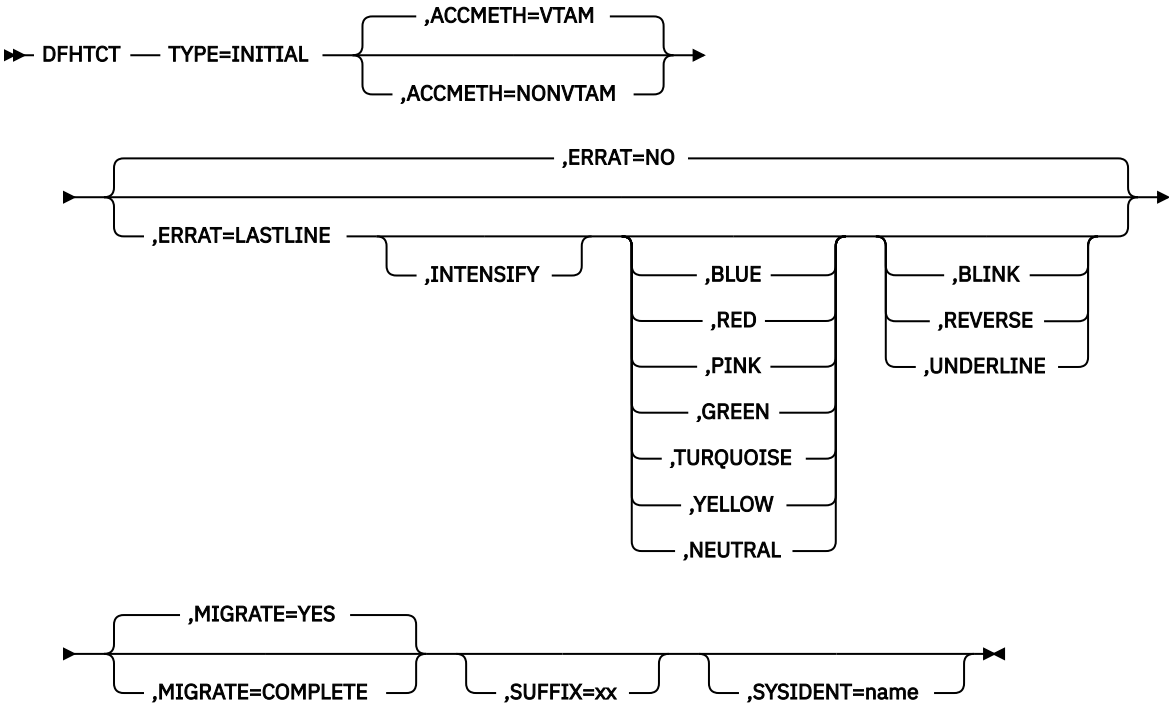
If you reassemble the TCT after starting CICS, any changes are picked up at a warm or emergency start.

Control section: DFHTCT TYPE=INITIAL

You code one DFHTCT TYPE=INITIAL macro before all the macros that define your resources.

The TYPE=INITIAL macro has two purposes:

1. To establish the area of storage into which the TCT is assembled.
2. To specify information that applies to the whole TCT, or to the individual non-z/OS Communications Server entries (and any z/OS Communications Server-LDC definitions).



Note: SYSIDNT is Intercommunication only

Note: For general information about TYPE=INITIAL macros, see TYPE=INITIAL (control section).

ACCMETH=([VTAM,]NONVTAM)
This specifies the access methods required in the running CICS system. VTAM is now z/OS Communications Server.

VTAM
Specify this if you are using the z/OS Communications Server as an access method.

NONVTAM
Specify this if you are using telecommunications access methods other than the z/OS Communications Server. These access methods include BSAM (for sequential devices).

Note: The default is to assume **both** VTAM and NONVTAM.

ERRATT={NOv([LASTLINE][, INTENSIFY] [,{BLUEvREDvPINKvGREENvTURQUOISEvYELLOWv NEUTRAL}][,{BLINKvREVERSEv UNDERLINE}])}
Indicates the way error messages are displayed on all 3270 display screens. You can either let it default to NO, or specify any combination of LASTLINE, INTENSIFY, one of the colors, and one of the highlights. Specifying INTENSIFY, one of the colors, or one of the highlights forces LASTLINE.

Any attributes that are not valid for a particular device are ignored.

NO
Any error message is displayed at the current cursor position and without any additional attributes.

LASTLINE
Any error message is displayed starting at the beginning of the line nearest the bottom of the screen, such that the message fits on the screen.

Attention: If messages are received in quick succession, they overlay one another. The earlier messages may disappear before the operator has read them.

INTENSIFY
Error messages are intensified, and placed on the last line of the screen.

BLUE|RED|PINK|GREEN|TURQUOISE|YELLOW| NEUTRAL
Error messages are shown in the color specified, and placed on the last line of the screen.

BLINK|REVERSE|UNDERLINE
Error messages are highlighted, and placed on the last line of the screen.

MIGRATE={YES|COMPLETE}
This operand controls the building of TCT entries for z/OS Communications Server devices that are **eligible** for resource definition online (RDO). The only way RDO-eligible resources may be moved to the CSD file from the macro source is to use DFHCSDUP, as described under YES.

YES
YES indicates that you want to generate the necessary data to migrate your RDO-eligible resources. The records generated from the macro source are designed to be used as input to the DFHCSDUP utility program. An MNOTE attention message is issued for each RDO-eligible resource.

Note: From CICS TS for z/OS, Version 4.1, the DFHCSDUP MIGRATE command is not supported. Use an earlier release of CICS to migrate your tables to the CSD.

COMPLETE
Use of COMPLETE means that TCT entries are not generated from the macro source for any RDO-eligible devices. For each one, the assembly produces an MNOTE. This means that you can keep your TCT macro source code after you have migrated your definitions.

If you continue to assemble a TCT for resources that are not eligible for RDO, continue to use MIGRATE=COMPLETE.

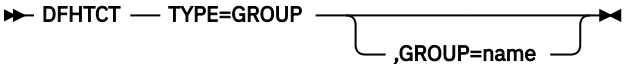
SYSIDNT={CICS|name}
This 1- to 4-character name is a private name that the CICS system uses to identify itself. If you use DFHTCT TYPE=REGION macros to define remote terminals, you must code this operand. It is used to determine whether a remote or a local TCT terminal entry is generated from each TYPE=TERMINAL macro following the TYPE=REGION macro. If the SYSIDNT on the TYPE=REGION macro is the same as the SYSIDNT on the TYPE=INITIAL macro, a local definition is generated; otherwise a remote definition is generated.

The value you code for this operand is used to define the name of the local region during assembly of the TCT only. You must also define the name of the local region, for execution purposes, using the SYSIDNT system initialization parameter.

Migrating TCT definitions: DFHTCT TYPE=GROUP

Use this macro to name the groups into which TCT definitions are put when you migrate to resource definition online.

This macro can appear as many times as required and at any point in the macro source. Each time it appears, it defines the CSD file group into which subsequent definitions are put until the next DFHTCT TYPE=GROUP macro occurs.



GROUP=name
Code this with the name of the group to which subsequent definitions are migrated. The name can be up to eight alphanumeric characters, but must not begin with DFH. The default name is TCT*xx*, where *xx* is the value coded for SUFFIX in the DFHTCT TYPE=INITIAL macro. If an error is found, the existing group name continues.

If a group with the name you specify does not already exist, it will be created. If it does exist, subsequent definitions are added to it.

DFHTCT logical device codes: z/OS Communications Server non-3270

Certain types of logical unit may be used to gain access to more than one resource within a subsystem. For example, a card punch device may be attached to a 3770 logical unit: the CICS application program can direct punch output, through BMS, via the 3770 to the card punch device. The facility provided by CICS to permit communication to devices within logical units of this type is the *logical device code* (LDC).

Certain types of logical unit may be used to gain access to more than one resource within a subsystem. For example, a card punch device may be attached to a 3770 logical unit: the CICS application program can direct punch output, through BMS, via the 3770 to the card punch device. The facility provided by CICS to permit communication to devices within logical units of this type is the **logical device code** (LDC).

Although these are z/OS Communications Server units, they require macro definition, unlike other z/OS Communications Server devices.

The logical units that support LDCs are:

3601 logical unit
3770 batch logical unit
3770 batch data interchange logical unit
3790 batch data interchange logical unit
LUTYPE 4 logical unit

To reference such a device in a CICS application program, or in the CMSG transaction for message switching, you specify an LDC mnemonic which CICS translates into a numeric LDC value. When CICS sends an output data stream to the logical unit, it includes the LDC value in the function management header (FMH). When the logical unit receives the data stream, it uses the LDC value to determine which component is to receive the output, or to perform some standard action.

Each LDC mnemonic to be referenced must be defined in the TCT, optionally with its associated LDC value and certain device characteristics for use by BMS functions. Such LDC information is contained in either the **system LDC table**, or in an **extended local LDC list**. You code the following DFHTCT macros to specify the system LDC table or an extended local LDC list:

- Code DFHTCT TYPE=LDC macro(s) to generate entries in the system LDC table. You may generate certain default LDC entries provided by CICS. For example,

  ```
  DFHTCT TYPE=LDC,LDC=SYSTEM
  ```

  generates the following entries in the system LDC table:

Table 15. System LDC table entries

LDC mnemonic	LDC value	Device	Pagesize (row,column)
DS	1	3604 Keyboard Display	6,40
JP	2	3610 Document Printer	1,80
PB	3	Passbook and Document Printer	1,40
LP	4	3618 Administrative Line Printer	50,80
MS	5	3604 Magnetic Stripe Encoder	1,40

Table 15. System LDC table entries (continued)			
LDC mnemonic	**LDC value**	**Device**	**Pagesize (row,column)**
CO	0	Console medium or default print data set group	
R1	32	Card input medium	1,80
H1	32	Card output medium	1,80
P1	48	Print medium or print data set group	50,80
W1	128	Word processing medium 1	50,80
W2	144	Word processing medium 2	50,80
W3	160	Word processing medium 3	50,80
W4	192	Word processing medium 4	50,80

You may also define LDCs specifically to add LDC entries to the system LDC table. For example,

```
   DFHTCT TYPE=LDC,
          LDC=XX,
          DVC=BLUPRT,
          PGESIZE=(12,80),
          PGESTAT=PAGE
   DFHTCT TYPE=LDC,
          LDC=YY,
          DVC=BLUPCH,
          PGESIZE=(1,80),
          PGESTAT=AUTOPAGE
```

adds the following entries to the system LDC table:

Table 16. System LDC table entries defined by LDCs				
LDC mnemonic	**LDC value**	**Device**	**Pagesize (row,column)**	**PGESTAT**
XX	48	Batch LU printer	12,80	PAGE
YY	32	Batch LU card output	1,80	AUTOPAGE

- Instead of the system LDC table, you may code the following series of DFHTCT TYPE=LDC macros to create an extended local LDC list. Default entries may also be generated. For example,

```
LDC1  DFHTCT TYPE=LDC,LOCAL=INITIAL
*  the next line generates default CO,R1,H1,P1 LDCs
   DFHTCT TYPE=LDC,LDC=BCHLU
   DFHTCT TYPE=LDC,DVC=BLUPRT,LDC=AA,
          PGESIZE=(6,30)
   DFHTCT TYPE=LDC,DVC=BLUPCH,LDC=BB,
          PGESIZE=(1,80)
   DFHTCT TYPE=LDC,DVC=BLUCON,LDC=CC,
          PGESIZE=(1,132)
   DFHTCT TYPE=LDC,LOCAL=FINAL
```

generates an extended local LDC list named LDC1 containing the following entries:

Table 17. Extended local LDC list entries			
LDC mnemonic	**LDC value**	**Device**	**Pagesize (row,column)**
CO	0	Console medium or default printer data set group	
R1	32	Card input medium	1,80
H1	32	Card output medium	1,80

Table 17. Extended local LDC list entries (continued)			
LDC mnemonic	**LDC value**	**Device**	**Pagesize (row,column)**
P1	48	Print medium or print data set group	50,80
AA	48	BLUPRT batch LU printer	6,30
BB	32	BLUPCH batch LU card output	1,80
CC	0	BLUCON batch LU console printer	1,132

When you are defining a logical unit in the TCT, you can specify its LDCs in either of two ways:

1. Code a DFHTCT TYPE=LDCLIST macro to define a local list of LDC mnemonics (and optionally their LDC values); for example,

   ```
   LDC2 DFHTCT TYPE=LDCLIST,
               LDC=(DS,JP,PB=5,LP,MS)
   ```

   In the DFHTCT TYPE=TERMINAL macro defining the logical unit, you specify in the LDC operand the name of the local list as defined by the DFHTCT TYPE=LDCLIST macro. For example:

   ```
        DFHTCT TYPE=TERMINAL,
               TRMTYPE=3600,
               LDC=LDC2, ...
   ```

   has associated the LDCs DS, JP, PB, LP, and MS with the 3601 logical unit that you are defining. The LDC values either may be specified in the local list, or are obtained from the system LDC table. If BMS uses these LDC mnemonics, their page size and page status must also be available from the system LDC table.

   Note: A local list defined by a DFHTCT TYPE=LDCLIST macro may be shared by a number of 3601, LUTYPE 4 and batch logical units.

2. In the DFHTCT TYPE=TERMINAL macro defining the logical unit, you specify in the LDC operand the name of an extended local LDC list. For example:

   ```
   LDC1 DFHTCT TYPE=LDC,LOCAL=INITIAL
        DFHTCT TYPE=LDC,LDC=BCHLU
        DFHTCT TYPE=LDC,DVC=BLUPRT,LDC=AA,
               PGESIZE=(6,30)
        DFHTCT TYPE=LDC,DVC=BLUPCH,LDC=BB,
               PGESIZE=(1,80)
        DFHTCT TYPE=LDC,DVC=BLUCON,LDC=CC,
               PGESIZE=(1,132)
        DFHTCT TYPE=LDC,LOCAL=FINAL
        DFHTCT TYPE=TERMINAL,TRMTYPE=BCHLU,
               LDC=LDC1, ...
   ```

   has associated the LDCs CO, R1, H1, P1, AA, BB, and CC with the batch logical unit that you are defining. Their LDC values and device characteristics for BMS functions are described in the extended local LDC list, which is named LDC1.

When CICS requests an output or message switching operation using a particular LDC mnemonic for a logical unit, it tries to resolve the mnemonic from the list (whichever form) specified by the LDC operand of the DFHTCT TYPE=TERMINAL macro. If the LDC is not located in the local list or in the extended local list, the LDC specified is not valid for that terminal entry. In this case, X'00' is inserted in the logical device code portion of the FMH, and no destination name is inserted.

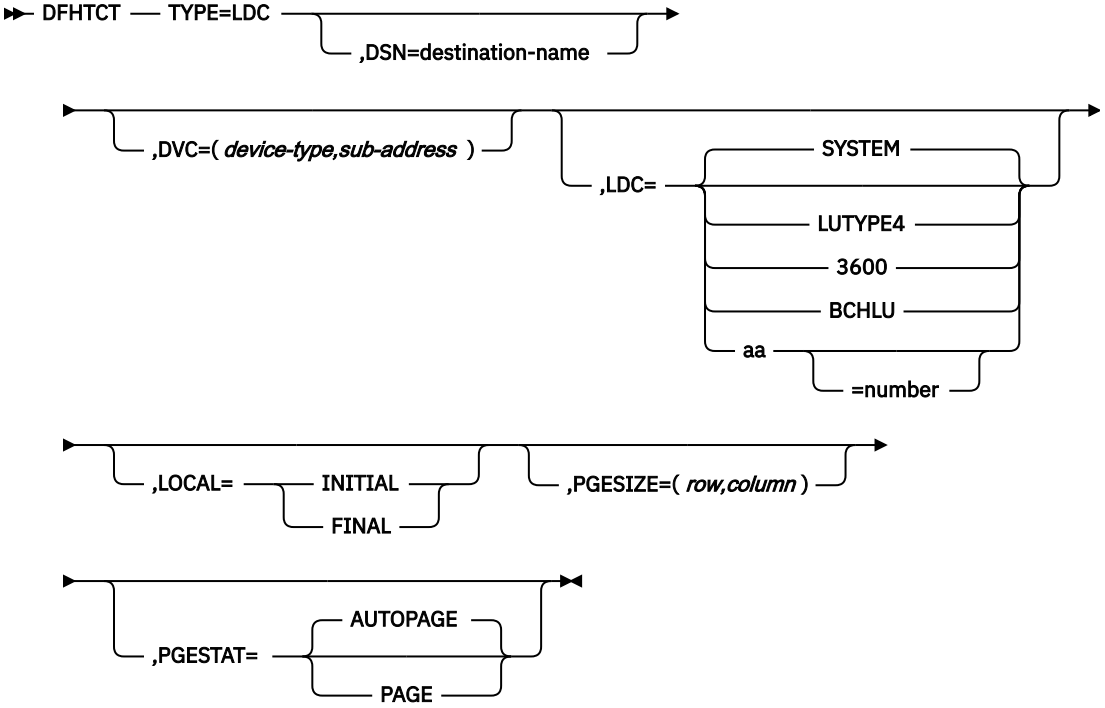
When a BMS function is requested for an LDC, and the LDC mnemonic is successfully resolved, the device characteristics (for example, device name and destination name) are accessed for the BMS function. If the LDC is in an extended local LDC list, these characteristics lie in the located extended local list entry. Otherwise, the system LDC table is searched for the LDC and the associated device characteristics.

Logical device codes—DFHTCT TYPE=LDC macro

The DFHTCT TYPE=LDC macro may only be used with 3600, LUTYPE4, 3770 batch logical unit, and 3770/3790 batch data interchange logical units.

You are responsible for setting up the LDC structure to be used with the terminal.

The expansion of this macro is the same, regardless of where it is specified in the TCT definition.



name
Code this with the name of the extended local LDC list. It should be the same as that specified in the LDC operand of the DFHTCT TYPE=TERMINAL macro, and is only required if LOCAL=INITIAL is coded.

TYPE=LDC
Code this if an LDC is being defined to the system LDC table or to the extended local LDC list.

DSN=destination-name
Code this with the name to be used by BMS for destination selection for the batch data interchange logical unit. See the relevant CICS subsystem guides for further information on destination selection.

DVC=(device-type,sub-address)
Code this with the device type associated with the LDC to be used for a BMS request. This operand can only be coded in conjunction with the LDC=*aa*[=*nnn*] operand.

device-type
May be coded as follows:

Table 18. DVC=device-type entries

Device type	Explanation
3604	Keyboard display
3610	Cut-forms document printer or journal printer (including the document/journal printer of a 3612)
3612	Passbook portion of a 3612
3618	Currently selected carriage
3618P	Primary carriage

Table 18. DVC=device-type entries (continued)	
Device type	**Explanation**
3618S	Secondary carriage
3618B	Both carriages
BLUCON	Batch logical unit console printer
BLUPRT	Printer component of a batch logical unit
BLURDR	Card input component of a batch logical unit
BLUPCH	Card output component of a batch logical unit
WPMED1	Word processing medium 1
WPMED2	Word processing medium 2
WPMED3	Word processing medium 3
WPMED4	Word processing medium 4

The device types BLUPRT, BLURDR, BLUPCH, and BLUCON are devices attached to a batch, batch data interchange, or LUTYPE4 logical unit.

The WPMED1, 2, 3, and 4 options apply to LUTYPE4 logical units only. The component to which these options apply is defined by the particular type 4 logical unit implementation.

sub-address
Code this with the media sub-address. The range is 0 through 15, with a default of 0. A value of 15 indicates any sub-address. The sub-address differentiates between two units of the same device type (for example, (BLUPRT,0) and (BLUPRT,1)), which could be two print components attached to one logical unit.

LDC={SYSTEM|LUTYPE4|3600|BCHLU|(aa[=nnn])}
Code this with the LDC mnemonic and numeric value to be defined. Only the LDC=*aa*[=*nnn*] option can be used in conjunction with the DVC, PGESIZE, and PGESTAT operands.

SYSTEM
The following system-default LDCs for 3600, batch, and LUTYPE4 logical units are to be established:

Table 19. System default LDCs			
LDC mnemonic	**LDC value**	**Device**	**Pagesize (row, column)**
DS	1	3604 Keyboard Display	6,40
JP	2	3610 Document Printer	1,80
PB	3	Passbook and Document Printer	1,40
LP	4	3618 Administrative Line Printer	50,80
MS	5	3604 Magnetic Stripe Encoder	1,40
CO	0	Console medium or default print data set group	
R1	32	Card input medium	1,80
H1	32	Card output medium	1,80
P1	48	Print medium or print data set group	50,80
W1	128	Word processing medium 1	50,80

Table 19. System default LDCs (continued)			
LDC mnemonic	**LDC value**	**Device**	**Pagesize (row, column)**
W2	144	Word processing medium 2	50,80
W3	160	Word processing medium 3	50,80
W4	192	Word processing medium 4	50,80

LUTYPE4
System-default LDC mnemonics are to be established for an LUTYPE4 (word processing) logical unit. These consist of the CO, R1, P1, H1, W1, W2, W3, and W4 mnemonics, the corresponding LDC values, and the appropriate page sizes.

3600
System-default LDC mnemonics for the 3600 are to be established. These consist of the DS, JP, PB, LP, and MS mnemonics, the corresponding LDC values, and the appropriate page-sizes and page-status.

BCHLU
System-default LDC mnemonics for a batch logical unit are to be established. These consist of the CO, R1, P1, and H1 mnemonics, the corresponding LDC values, and the appropriate page-sizes and page-status.

aa
The 2-character mnemonic to be used for this LDC.

nnn
The numeric value to be associated with the LDC in the system or extended local LDC list. The value in the system list is used as a default value for this LDC if a value is not found in a local LDC list (that is, not in the extended list) associated with a TCTTE. A value must be specified for a 3600 device. A value need not be specified for a batch, batch data interchange, or LUTYPE4 logical unit but, if one is specified, it must correspond to the LDC value for the device type.

LOCAL={INITIAL∨FINAL}
An extended local LDC list is to be generated.

INITIAL
This is the start of an extended local LDC list.

FINAL
This is the end of an extended local LDC list.

Note: LOCAL=INITIAL or FINAL may not be coded in the same DFHTCT TYPE=LDC macro as other operands. All DFHTCT TYPE=LDC entries coded after LOCAL=INITIAL and before LOCAL=FINAL form part of one extended local LDC list; the entries coded outside the structure of this group are added to the system LDC table.

The following is an example of an extended local LDC list:

```
         DFHTCT TYPE=TERMINAL,TRMIDNT=BTCH,      *
         TRMTYPE=BCHLU,ACCMETH=VTAM,LDC=LDCA     *
LDCA     DFHTCT TYPE=LDC,LOCAL=INITIAL
         DFHTCT TYPE=LDC,DVC=BLUPRT,LDC=AA,      *
         PGESIZE=(6,30)
         DFHTCT TYPE=LDC,DVC=BLUPCH,LDC=BB,      *
         PGESIZE=(1,80)
         DFHTCT TYPE=LDC,DVC=BLUCON,LDC=CC,      *
         PGESIZE=(1,132),PGESTAT=AUTOPAGE
         DFHTCT TYPE=LDC,LOCAL=FINAL
```

Note: VTAM is now z/OS Communications Server.

PGESIZE=(row,column)
Code this with the logical page size to be used with this LDC when BMS requests are processed.

The product of *row* and *column* must not exceed 32767.

PGESTAT={AUTOPAGE|PAGE}
Indicates whether the device is to use autopaging or not. Autopaging means that BMS multiple page messages are printed continuously, without operator intervention. This is what is normally required for a printer. (Contrast the requirement for multiple page messages, displayed on a 3270-type display, when the operator wants to finish reading a page, before requesting the next page to be delivered.)

Only BMS SEND commands with the PAGING option use autopaging. BMS SEND with TERMINAL or SET, does not use autopaging.

AUTOPAGE
Specify this for printers.

PAGE
Specify this for displays.

If the default PGESIZE or PGESTAT values provided by the LDC operand are to be overridden, code a specific LDC with the mnemonic to be overridden. Code this overriding LDC in the LDC table before coding the LDC operand.

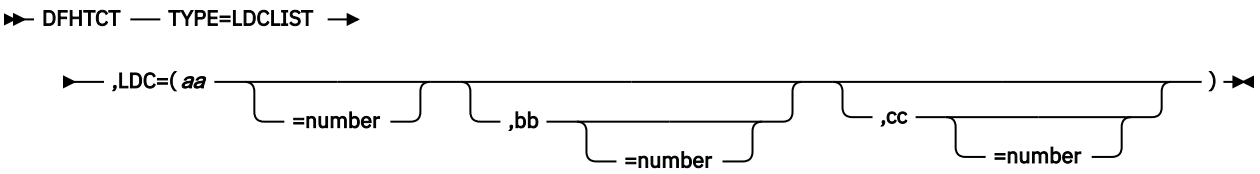
PGESTAT=AUTOPAGE may be used to override the PGESTAT specification in DFHTCT TYPE=TERMINAL.

Logical device codes—DFHTCT TYPE=LDCLIST

The DFHTCT TYPE=LDCLIST macro, which may be used with 3600, LUTYPE4, and batch logical units, allows you to build a common list of logical device codes (LDCs) to be shared by more than one TCTTE.

You are responsible for setting up the LDC structure to be used with the terminal.

The expansion of this macro is the same, regardless of where it is coded in the TCT definition.



listname
Is the required name of the LDC list. This name is referenced by TCTTEs through the LDC operand in DFHTCT TYPE=TERMINAL.

TYPE=LDCLIST
An LDC list is being defined.

LDC=(aa[=nnn][,bb[=nnn]][,cc[=nnn]][,...])
Code this with the LDCs (mnemonics and, optionally, the LDC numeric value) in this list.

(aa[=nnn][,bb[=nnn]] [,cc[=nnn]][,...])
Generates the LDCs in the list.

aa,bb,cc...
The 2-character mnemonics of the LDCs in this list.

nnn
A decimal value in the range 1 through 255 to be associated with an LDC. If a value is not specified, the system default value from the table defined by the DFHTCT TYPE=LDC macro, is used for this LDC. This value need not be coded for a batch or LUTYPE4 logical unit, but if it is, it must correspond to the LDC value for the device. LDCs for devices attached to a batch or LUTYPE4 logical unit are listed under the LDC parameter of the DFHTCT TYPE=LDC macro.

DFHTCT examples: LDC

An example showing how to code the DFHTCT TYPE=LDC macro for a 3770 batch logical unit.

```
DFHTCT TYPE=LDC,                        *
       LDC=XX,                          *
       DVC=BLUPRT,                      *
       PGESIZE=(12,80),                 *
       PGESTAT=PAGE
DFHTCT TYPE=LDC,                        *
       LDC=YY,                          *
       DVC=BLUPCH,                      *
       PGESIZE=(1,80),                  *
       PGESTAT=AUTOPAGE
DFHTCT TYPE=LDC,                        *
       LDC=SYSTEM
```

Figure 15. LDCs for 3770 batch logical unit TCT example

Sequential devices

CICS uses BSAM to control sequential terminals such as card readers, line printers, magnetic tape units and DASD. Sequential terminals provide simulation of actual terminals, and can be used before the actual terminals become available. Sequential terminals can also be used during the testing of new applications.

To define a sequential device, code the following macro instructions:

```
      DFHTCT TYPE=INITIAL,
             ACCMETH=(NONVTAM)    defining the access
                                  method
```

(Define the following macro instructions contiguously.)

```
DFHTCT TYPE=SDSCI,
       DSCNAME=isadscn,     defining the input
       DDNAME=indd, ...      data set
DFHTCT TYPE=SDSCI,
       DSCNAME=osadscn,     defining the output
       DDNAME=outdd, ...     data set
DFHTCT TYPE=LINE,
       ISADSCN=isadscn,
       OSADSCN=osadscn, ...
DFHTCT TYPE=TERMINAL,
       TRMIDNT=name, ...
```

The two data sets defined by the DFHTCT TYPE=SDSCI macros simulate a CICS terminal known by the name specified in the TRMIDNT operand of the DFHTCT TYPE=TERMINAL macro. The DSCNAMEs of the input and output data sets must be specified in the ISADSCN and OSADSCN operands of the DFHTCT TYPE=LINE macro respectively.

The end of data indicator (EODI) for sequential devices can be altered using the EODI system initialization parameter.

JCL for sequential devices

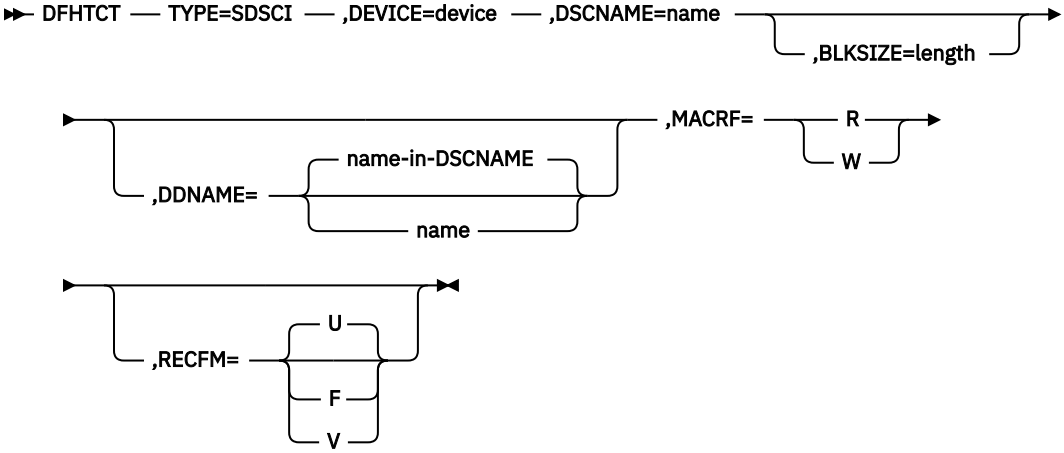
The DDNAME operands on the DFHTCT TYPE=SDSCI macros specify the ddname of the DD statements which you must provide in the CICS startup job stream.

```
    //indd   DD  ...               input data set
    //outdd  DD  ...               output data set
```

where *indd* is the data set containing input from the simulated terminal, and *outdd* is the data set to which output to the simulated terminal is sent.

Sequential devices: DFHTCT TYPE=SDSCI

The DFHTCT TYPE=SDSCI macro specifies the characteristics of the input and output data sets that simulate a CICS terminal.



BLKSIZE=length
Code this with the maximum length in bytes of a block.

The default is BLKSIZE=0. If this operand is omitted, the block size can be specified in the data definition (DD) statement associated with the data set. A more detailed explanation of this operand is given in z/OS DFSMS Macro Instructions for Data Sets.

DDNAME={name-in-DSCNAME|name}
Supplies the name of the data definition (DD) statement associated with a particular data set (line group). If this operand is omitted, the DSCNAME becomes the DDNAME.

DEVICE=device
One of the following values may be coded:

- For card readers: **{1442|2501|2520|2540|2560|2596| 3505|3525|5425}**
- For line printers: **{1403|1404|1443|1445|3203|3211|5203}**
- For disk (DASD): **{2314|3330|3340|3350|DASD|DISK}**
- For tapes: **TAPE**.

  The TAPE specification generates tape work files for both the input and the output data sets. Note that if an input tape with an expired label is used, the header may be rewritten, causing the first data records to be destroyed.

DSCNAME=name
The name of either the input or the output data set. If you are defining the input data set, ISADSCN on the DFHTCT TYPE=LINE macro must match the name that you specify: if you are defining the output data set, OSADSCN on the DFHTCT TYPE=LINE macro must match it.

MACRF=([R][,W])
Code this with the way in which access to the sequential device is to be gained.

R
Indicates the READ macro.

W
Indicates the WRITE macro.

The default is MACRF=R for a card reader and MACRF=W for a line printer. For other sequential devices, MACRF=R or MACRF=W must be coded.

RECFM={U|F|V}
Code this with the record format for the DCB.

U
Indicates undefined records. Code this option for DEVICE=1403 or 3211, or if you are using DASD for sequential terminal output (that is, if DEVICE=DASD and MACRF=W).

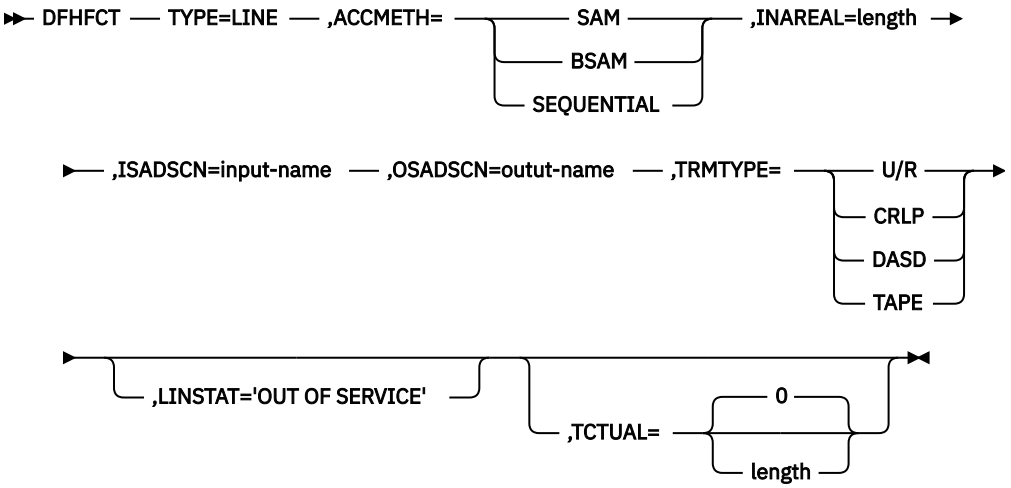
F
Indicates fixed-length records.

V
Indicates variable-length records.

If you omit this operand, you can specify the record format in the data definition (DD) statement associated with the sequential data set.

Sequential devices—DFHTCT TYPE=LINE

The DFHTCT TYPE=LINE macro specifies further characteristics of the sequential data sets that simulate a CICS terminal.



ACCMETH={SAMvBSAMvSEQUENTIAL}
Specify SAM, BSAM, or SEQUENTIAL — they are equivalent in CICS.

INAREAL=length
Code this with the message input area length. The value should be equal to the length of the longest initial logical record of a transaction that may include multiple physical records.

ISADSCN=name
The name of the input data set. The TYPE=SDSCI DSCNAME operand for the input data set must match this.

LINSTAT='OUT OF SERVICE'
The line is to be initiated with an 'out of service' status.

The default is 'in service'.

OSADSCN=name
The name of the output data set. The TYPE=SDSCI DSCNAME operand for the output data set must match this.

TCTUAL={0vlength}
Indicates the length in bytes (0 through 255) of the user area (the process control information field or PCI) for all terminal entries (TCTTEs) associated with this line. Make it as small as possible. The TCT user area is initialized to zeros at system initialization. If you want fields of different (variable) lengths, you can specify the TCTUAL value in one or more TYPE=TERMINAL macro instructions for terminals associated with this line.

TRMTYPE=(U/RvCRLPvDASDvTAPE)
Indicates the sequential device type:

U/R
Any reader or printer

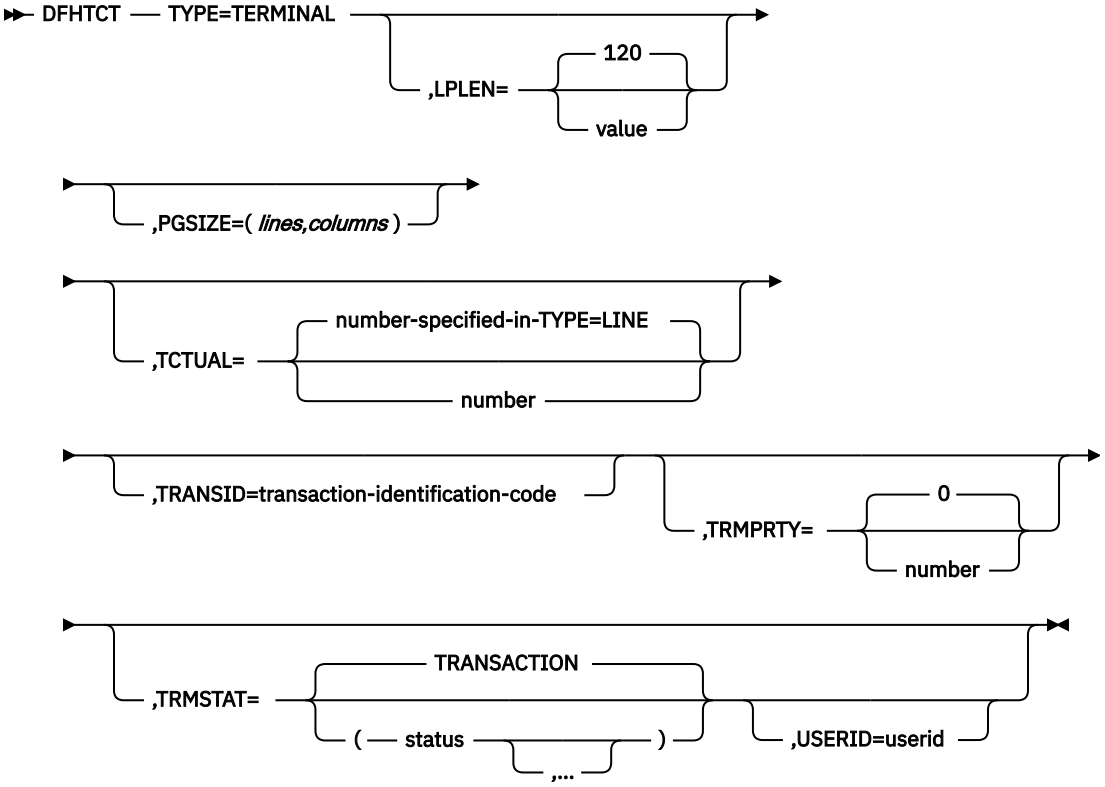
CRLP
A card reader and a line printer

DASD
A direct access storage device

TAPE
A magnetic tape device

Sequential devices: DFHTCT TYPE=TERMINAL

The DFHTCT TYPE=TERMINAL macro specifies the terminal name and other characteristics of a CICS terminal that is simulated by a pair of sequential data sets.



LPLEN={120|value}
Controls the length of the print line for SAM output line printers. If no NL symbol is found in a segmented write, the print line length is the LPLEN value. The default is LPLEN=120.

PGESIZE=(lines,columns)
The default page size for a 1403 or CRLP terminal is (12,80). Code PGESIZE if BMS is required for a device that has TRMTYPE=DASD specified, and specify the number of lines and columns you want to use. These two values multiplied together must equal the value specified for INAREAL. The product of *lines* and *columns* must not exceed 32767.

TCTUAL={number-specified-in-TYPE=LINE|number}
Indicates the length in bytes (0 through 255) of the user area (the process control information field or PCI) for the terminal entry (TCTTE) associated with this terminal. Make it as small as possible. The TCT user area is initialized to zeros at system initialization.

Use the TCTUAL operand of the DFHTCT TYPE=TERMINAL macro if you want fields of different (variable) lengths for terminals associated with this line. In any case, the PCI field is generated for each terminal after the last terminal entry of the last line. The address of the PCI field is located at TCTTECIA; the length is located at TCTTECIL.

TRANSID=transaction-identification-code
Code this with a 1- to 4-character transaction code. This code specifies a transaction that is to be initiated each time input is received from the terminal when there is no active task.

If a TRANSID is not specified in the TCTTE, the TRANSID in a RETURN command from the previous transaction is used. Otherwise, the first one to four characters of the data passed in the TIOA are used as the transaction code. A delimiter is required for transaction identifications of fewer than four characters.

TRMIDNT=name
Code this with a unique 4-character symbolic identification of each terminal. The identification supplied is left-justified and padded with blanks to four characters if less than four characters are supplied.

The value CERR is reserved, as this is the identification generated for the error console.

TRMPRTY={0|number}
Establishes the terminal priority. This decimal value (0 through 255) is used in establishing the overall transaction processing priority. (Transaction processing priority is equal to the sum of the terminal priority, transaction priority, and operator priority, and must not exceed 255.)

TRMSTAT={TRANSACTION|(status,...)}
Code this with the types of activity that may occur at a given terminal. This terminal status is initially set in the TCTTE and is a combination of the processing status and the service status.

TRANSACTION
A terminal with TRANSACTION status is used in the processing of transactions such as inquiries or order entries. A display station or a hard-copy terminal, to which no messages are sent without a terminal request, and through which transactions are entered, is a TRANSACTION terminal.

INPUT
Indicates a terminal that can send messages to, but cannot receive messages from, CICS.

Note: System messages may be routed to an input terminal under conditions such as invalid transaction identification and ATP batch count. This causes DFHTACP to be scheduled. To handle this situation, code a DFHTEP to perform any action that the user requires.

'OUT OF SERVICE'
Indicates a terminal that can neither receive messages nor transmit input. Such terminals are not polled by CICS. The 'OUT OF SERVICE' parameter can be used in combination with any status setting.

Any terminal except the main terminal can be designated as 'OUT OF SERVICE'. When appropriate, the terminal can be placed in service by the main terminal and polling is resumed.

RECEIVE
Indicates a terminal to which messages are sent but from which no input is allowed. An example of this type of terminal is one that is located in a remote location, such as a warehouse, and is unattended, but may receive messages. Automatic transaction initiation is implemented as for TRANSCEIVE.

TRANSCEIVE
A terminal with TRANSCEIVE status is a TRANSACTION terminal to which messages are sent automatically. The automatic transaction initiation, either by transient data control or interval control, sets a condition in an appropriate terminal control table terminal entry. If the terminal status is TRANSCEIVE and if there is no transaction at the terminal, terminal control initiates the user-defined task. This task is expected to send messages to the terminal.

USERID=userid
Code this to specify a user identifier for devices such as printers that are unable to sign on using CESN. (You can also specify USERID for a display device, in which case the display is permanently signed on. Operators are unable to sign on.) You must code this operand if you want to use preset security with this device. All access to protected resources depends on USERID.

The userid is referred to in security error messages, security violation messages, and the audit trail. It must be defined to the security manager.

Userid must be a unique 1- to 8-character user identification. (A-Z 0-9 # $ and @ are acceptable characters.)

Remote terminals for transaction routing

Regarding CICS intercommunication, when you are concerned with resource definition, the system where the TCT is installed is the **local** system. The system that is being defined in the TCT is the **remote** system. *Transaction routing* enables terminals in one CICS system to invoke transactions in another CICS system.

You can use transaction routing between systems connected by MRO or by an LUTYPE 6.2 link.

Remote definitions for terminals for transaction routing

There are two possible methods of defining the terminals using macros.

The two methods are:

- **Method 1:**

```
DFHTCT TYPE=REGION, ...   (one for each region)

DFHTCT TYPE=SDSCI, ...    (for non-SNA LU only;
                           ignored for remote definitions)

DFHTCT TYPE=LINE, ...     (for non-SNA LU only)

DFHTCT TYPE=TERMINAL, ... (for non-SNA: one for each LU)
```

- **Method 2:**

```
DFHTCT TYPE=REMOTE, ...   (one for each terminal)
```

Tip: Another method, called 'shipping terminal definitions', is possible using RDO. See “Terminals for transaction routing” on page 176.)

Both methods allow the same terminal definitions to be used to generate the required entries in both the local and the remote system.

Method 1:
You can use copybooks to include the same source code in the TCTs for local and remote systems. The information not needed (that is, the whole of the TYPE=SDSCI macro, and some of the TYPE=LINE and TYPE=TERMINAL macros) is discarded for remote entries.

CICS decides whether to create a remote or a local definition on the basis of the SYSIDNT operand on the TYPE=REGION macro. This is compared with the SYSIDNT operand in DFHTCT TYPE=INITIAL. If they are the same, the definition(s) are local. If they are different, the definition(s) are remote.

Method 2:
Employs a single DFHTCT TYPE=REMOTE macro.

CICS decides whether to create a remote or a local definition on the basis of the SYSIDNT operand on the TYPE=REMOTE macro. This is compared with the SYSIDNT operand in DFHTCT TYPE=INITIAL. If they are the same, the definition(s) are local. If they are different, the definition(s) are remote.

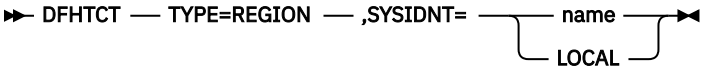
These terminals cannot use transaction routing and therefore cannot be defined as remote:

- IBM 7770 or 2260 terminals
- MVS system consoles
- Pooled 3600 or 3650 Pipeline Logical Units

Remote terminals, method 1—DFHTCT TYPE=REGION

The DFHTCT TYPE=REGION macro introduces information about the named region. The information consists of DFHTCT TYPE=LINE and TYPE=TERMINAL macros.

These macros must follow the DFHTCT TYPE=REGION macro. For a remote region, the DFHTCT TYPE=LINE macro does not generate a TCT line entry (TCTLE). Every terminal that participates in transaction routing must be defined. Only certain DFHTCT macro types and operands are relevant in remote region definitions; all others are ignored. The operands that are relevant are those listed in “Remote terminals, method 2: DFHTCT TYPE=REMOTE” on page 412.

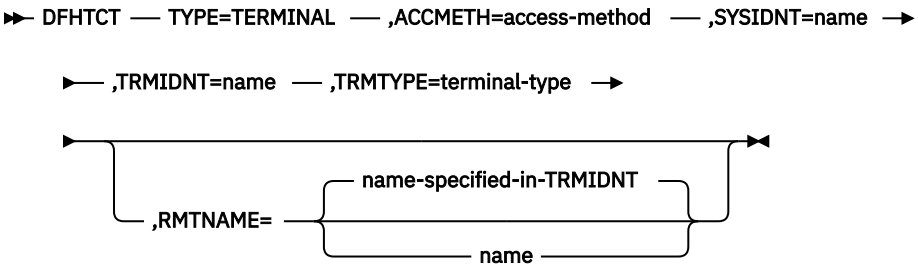


SYSIDNT={name|LOCAL}
Indicates the 4-character name of the system or region whose information starts or resumes here. SYSIDNT=LOCAL can be specified to indicate that the TYPE=TERMINAL definitions following it refer to the home region, as do all definitions preceding the first DFHTCT TYPE=REGION macro. The name of the home region (that is, the region in which this terminal control table is used) is the value of the SYSIDNT operand of the DFHTCT TYPE=INITIAL macro. The name can instead be that of a previously defined MRO link or ISC link.

Remote terminals, method 1—DFHTCT TYPE=TERMINAL

TCT macro definitions for defining remote terminals using method 1.

Note: The DFHTCT TYPE=LINE macro and the additional operands of the DFHTCT TYPE=TERMINAL macro are valid, but are ignored if the SYSIDNT operand on the preceding DFHTCT TYPE=REGION macro indicates a remote region. (For details of the DFHTCT TYPE=LINE and DFHTCT TYPE=TERMINAL macros, see “Sequential devices” on page 405)



ACCMETH=access-method
Code this with the access method of the remote terminal.

RMTNAME={name-specified-in-TRMIDNT|name}
Specifies the 1- to 4-character name by which the terminal is known in the system or region that owns the terminal (that is, in the TCT of the **other** system). If this operand is omitted, the name in the TRMIDNT operand is used.

SYSIDNT=name
Indicates the 4-character name of the system or region that owns this terminal. This may be the local system or region (that is, the name defined in the TYPE=INITIAL macro), in which case the TCT entry created is a local definition. It may be the name of a different system or region, in which case the TCT entry created is a remote definition. This SYSIDNT must be the same as the SYSIDNT on the TYPE=REGION macro that precedes this macro.

TRMIDNT=name
Specifies the 1- to 4-character name by which the terminal is known in **this** system (that is, in the local system that owns this TCT, and that owns the transactions).

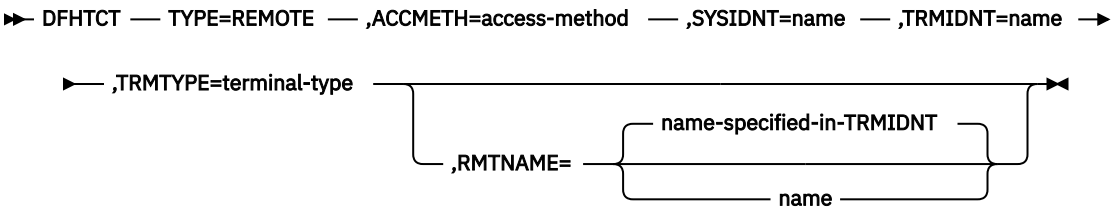
TRMTYPE=terminal-type
Code this with the terminal type. For details, see “Sequential devices” on page 405.

Remote terminals, method 2: DFHTCT TYPE=REMOTE

Terminal entries for remote systems or regions can be defined to CICS using the DFHTCT TYPE=REMOTE macro as an alternative to defining them using DFHTCT TYPE=TERMINAL macro instructions in conjunction with a DFHTCT TYPE=REGION macro.

The expansion of the DFHTCT TYPE=REMOTE macro is independent of the region currently referenced.

Note: If the SYSIDNT operand indicates that the **home** region owns the terminal, all the operands of the DFHTCT TYPE=TERMINAL macro become valid on the DFHTCT TYPE=REMOTE macro and have the same meaning as for TYPE=TERMINAL. However, if (as is normally the case) the SYSIDNT operand indicates a remote region, the additional operands of DFHTCT TYPE=TERMINAL are valid on the DFHTCT TYPE=REMOTE macro, but are ignored.



ACCMETH=access-method
Code this with the access method of the remote terminal.

RMTNAME={name-specified-in-TRMIDNT|name}
Specifies the 1- to 4-character name by which the terminal is known in the system or region that owns the terminal (that is, in the TCT of the **other** system). If this operand is omitted the name in the TRMIDNT operand is used.

SYSIDNT=name
Specifies the name of the system or region that owns this terminal. The name must be the same as that used in the SYSIDNT operand of a previous TYPE=SYSTEM macro, or the TYPE=INITIAL macro.

TRMIDNT=name
Specifies the 1- to 4-character name by which the terminal is known in **this** system (that is, in the local system that owns this TCT, and that owns the transactions).

TRMTYPE=terminal-type
Code this with the terminal type. For details, see “Sequential devices” on page 405.

DFHTCT: CICS terminals list

This release of CICS is able to communicate with a wide range of terminals, either directly or indirectly.

New or current terminals are directly supported by CICS Transaction Server for z/OS if they conform to the z/OS Communications Server interface.

Table 20 on page 412 summarizes how terminals are supported in CICS.

Table 20. IBM terminals and system types supported by CICS Transaction Server for z/OS.

Directly supported by CICS Transaction Server for z/OS using the z/OS Communications Server
3101 Display Terminal
3230 Printer
3268 Printer
3270 Information Display System
3270 PC
3270 PC/G
3270 PC/GX

Table 20. IBM terminals and system types supported by CICS Transaction Server for z/OS. (continued)

Directly supported by CICS Transaction Server for z/OS using the z/OS Communications Server
3287 Printer
3600 Finance Communication System
3630 Plant Communication System
3640 Plant Communication System
3650 Retail Store System
3680 Programmable Store System
3730 Distributed Office Communication System
3767 Communication Terminal
3770 Data Communication System
3790 Communication System
4300 Processors
4700 Finance Communication System
5280 Distributed Data System
5520 Administrative System
5550 Administrative System
5937 Rugged Terminal
6670 Information Distributor
8100 Information System
8775 Display Terminal
8815 Scanmaster
Displaywriter
Personal Computer, PS/2, PS/55
System/32
System/34
System/36
System/38
iSeries
z Systems
Teletypewriter Exchange Service (TWX 33/35)
World Trade Typewriter Terminal (WTTY)

z/OS Communications Server LUs

For a detailed list of z/OS Communications Server for SNA-supported LUs, and how to define them to CICS, see “Devices supported ” on page 225.

TLT: terminal list table

A terminal list table (TLT) generated by the DFHTLT macro instruction allows terminal and operator identifications to be grouped logically.

A TLT:

- Is **mandatory** for use by CEST (the supervisor terminal transaction), to define and limit the effective range of the operation. For example:

  ```
  CEST SET TERMINAL(*) SUPRID(CG) OUTSERVICE
  ```

  sets all terminals defined in DFHTLTCG out of service.
- May be used by CEST or CEMT (the main terminal transaction) to apply an operation to a predetermined group of terminals. (For a CEST operation, this TLT must define a subset of the TLT specified by SUPRID.) For example, each of the following commands:

  ```
  CEST SET TERMINAL(*) SUPRID(CG) CLASS(EM) INSERVICE
  CEMT SET TERMINAL(*) CLASS(EM) INSERVICE
  ```

  sets all terminals defined in DFHTLTEM in service.
- May be used singly or in combination with other TLTs to provide predefined destinations for message switching. For example:

  ```
  CMSG ROUTE=PG,'PRODUCTION MEETING AT 11.00 IN
              ROOM 2124',SEND
  ```

  sends a message to all terminals or operators defined in DFHTLTPG.

The same TLT can be used for message switching and for supervisory or main terminal functions. For example, a TLT defining the terminals that are under control of a supervisory terminal could also be used as a destination list for sending messages to those terminals.

For some logical units, logical device code (LDC) mnemonics (that may be associated with each table entry), are used for message switching and are ignored for main and supervisory terminal operations.

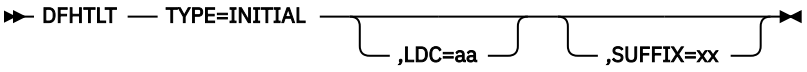
In an intercommunication network, all the terminals in a terminal list table must be owned by the system on which the table is used.

The following macros define the TLT entries:

- Control section: DFHTLT TYPE=INITIAL
- Entries in terminal list table: DFHTLT TYPE=ENTRY
- End of terminal list table: DFHTLT TYPE=FINAL (see TYPE=FINAL (end of table))

Control section: DFHTLT TYPE=INITIAL

The DFHTLT TYPE=INITIAL macro establishes the entry point and the address of the start of the terminal list table being defined.



Note: For general information about TYPE=INITIAL macros, see TYPE=INITIAL (control section).

LDC=aa
Code this with a 2-character logical device code (LDC) mnemonic. This is associated with every logical unit identification, except for those for which an LDC mnemonic is specified on a DFHTLT TYPE=ENTRY macro.

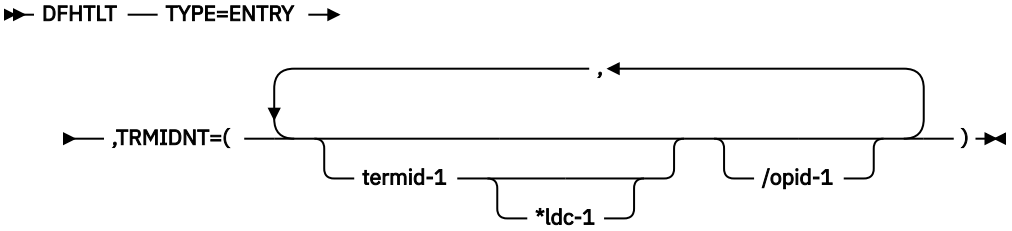
SUFFIX=xx
The module name of the TLT is DFHTLT*xx*, where *xx* is a 1-or 2-character suffix. This provides unique identification for each TLT used. Because the names TLTBA, TLTBB, TLTBC, and TLTEA are used within the TLT, suffixes BA, BB, BC, and EA must not be used.

A TLT must have a suffix to be used by the message switching transaction, CMSG.

Entries in terminal list table: DFHTLT TYPE=ENTRY

The DFHTLT TYPE=ENTRY defines an entry in the terminal list table (TLT).

Entries are coded in the TLT as follows:



TYPE=ENTRY
Code this if one or more entries are to be generated in this table, up to a maximum of 1000 entries.

TRMIDNT=([termid-1[*ldc-1]] [/opid-1][, termid-2[*ldc-2][/opid-2],...])
Code this with a list of start-stop and BSC terminal, logical unit, and operator identifications. A logical unit identification can be qualified by an LDC mnemonic.

termid
Indicates a 1- to 4-character start-stop or BSC terminal or logical unit identification.

Note: A 3614 attached to a communications controller may be used in main or supervisory terminal operations but should not be used in message switching operations. (A 3614 is not valid for a message destination.)

ldc
Indicates a 2-character LDC mnemonic, which must be preceded by an asterisk (*) and is only used following the 'termid' parameter to which it is appended.

opid
Indicates a 1- to 3-character operator identification that must be preceded by a slash (/).

Any terminal or operator identification specified should also be specified in the TRMIDNT operand of the DFHTCT macro and in your external security manager, respectively. (If you employ RACF, you use the OPIDENT operand of the ADDUSER command to record the identification for each operator.) Any LDC mnemonic specified should also be specified in the LDC operand of the DFHTCT TYPE=LDC and DFHTCT TYPE=TERMINAL macros.

Supervisory and main terminal functions use the terminal and logical unit identifications included in the TLT, but ignore all references to LDC mnemonics and operator identifications.

DFHTLT example

An example of how to create a terminal list table (TLT).

```
Example 1

DFHTLT TYPE=INITIAL,                                  *
       SUFFIX=AA
DFHTLT TYPE=ENTRY,                                    *
       TRMIDNT=(NYC,CHI,LA,WDC)
DFHTLT TYPE=ENTRY,                                    *
       TRMIDNT=SF
DFHTLT TYPE=ENTRY,                                    *
       TRMIDNT=(BSTN/OP1,ATL/OP5,/OP9,DNVR)
DFHTLT TYPE=ENTRY,                                    *
       TRMIDNT=/OP6
DFHTLT TYPE=FINAL
END

Example 2

DFHTLT TYPE=INITIAL,                                  *
       SUFFIX=XX
DFHTLT TYPE=ENTRY,                                    *
       TRMIDNT=(NYC,T361*LP,T362*LP/OP1)
DFHTLT TYPE=ENTRY,                                    *
       TRMIDNT=(T363/OP2,T364/OP5,T365)
DFHTLT TYPE=FINAL
END
```

Figure 16. Terminal list table: example

TST: temporary storage table

The temporary storage table (TST) is a list of generic names (or prefixes) used to identify sets of temporary storage queues. Any unique temporary storage identifier generated dynamically in an application program that begins with the same characters as the generic names automatically acquires the same properties as the TST entries.

CICS still supports the use of the DFHTST macro in combination with or in place of TSMODEL resource definitions. You must use a TST in the following circumstances:

- You have application programs that reference temporary storage data sharing queues by specifying an explicit SYSID on EXEC CICS temporary storage commands.
- A SYSID is added for EXEC CICS temporary storage commands by an XTSEREQ global user exit program.
- You require the TSAGE attribute.

For temporary storage queues where you do not require these functions, you can use TSMODEL resource definitions, which provide all other functions of the TST and some additional functions.

The default TST=NO system initialization parameter means that CICS initializes with only RDO support for TS queues. To use a TST in combination with TSMODEL resource definitions, you must specify a TST suffix using the TST system initialization parameter. You must also assemble the TST load module with the MIGRATE option. If the TST is not assembled with the MIGRATE option, CICS loads the TST only and does not provide any RDO support for TS queues, and any attempts to install TSMODEL resource definitions are rejected.

If you use both a TST and TSMODEL resource definitions, the use of the TST is limited to the following:

- Support for TS data sharing queues that are referenced by an explicit SYSID option specified on a TS API command.
- The TSAGE attribute.

If you use a TST alone, all the functions of the TST are used.

Generic names

In a TST, generic names are formed from the leading characters of the appropriate queue names, and can be up to seven characters long.

- The generic names coded on a DFHTST TYPE=RECOVERY macro identify queues for which CICS provides backout of changes in the event of transaction failure or protection against system failure.
- The generic name coded on a DFHTST TYPE=REMOTE macro identifies queues for which CICS routes the temporary storage request to a remote CICS region or TS server, unless the remote system name (SYSIDNT) is the same as that of the local CICS. If SYSIDNT is the same name as the local CICS, the queues specified by the DATAID option are treated by CICS as local queues.
- The generic name coded on a DFHTST TYPE=LOCAL macro identifies queues as local queues that reside in the CICS region in which the TST is installed.
- The generic name coded on a DFHTST TYPE=SECURITY macro identifies queues for which resource security checking is required.

If you specify an eight-character name, this defines a unique temporary storage queue name.

Choose a naming convention for queue names that enables you to define many queues with only a few generic names. This reduces considerably the task of TST definition. Bear in mind that CICS searches the TST for the first prefix that satisfies the particular search criteria. For example, if CICS searches for temporary storage queue ABCDEFGH, and the TST contains prefix A followed by prefix AB, A is selected. To avoid any problems, define the less-generic entries to the TST before any more-generic entries, so that the first to be found is the least generic of all possible matches.

When CICS is looking for generic names to match against a TS queue name, it searches only the types of entry in which it is interested for that particular search. CICS searches:

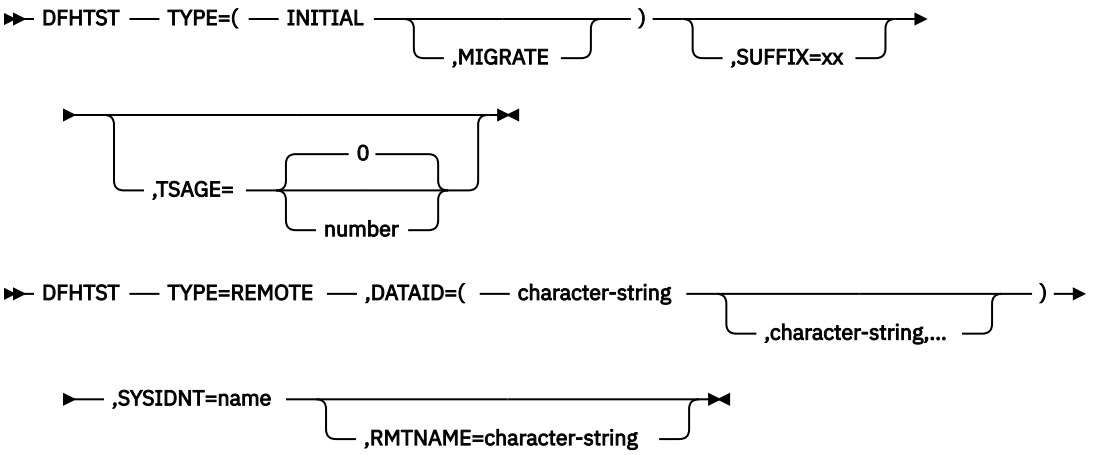
- Local *and* remote entries when determining whether a queue is remote. Thus, local and remote entries are regarded as one search category when CICS is matching a queue name against generic names.
- Recovery and remote entries when determining whether a queue is recoverable. However, if the leading characters of a queue name match **both** TYPE=RECOVERY and TYPE=REMOTE generic names, TYPE=REMOTE takes precedence, and the recovery option must be redefined in the local region in which the queue resides. (Queues in a shared TS pool cannot be recoverable.)
- Security entries only when determining whether a queue is subject to security.

Use these macros to define the TST entries:

- Control section: DFHTST TYPE=INITIAL
- Recoverable temporary storage: DFHTST TYPE=RECOVERY
- Local temporary storage: DFHTST TYPE=LOCAL
- Remote temporary storage: DFHTST TYPE=REMOTE
- Temporary storage security checking: DFHTST TYPE=SECURITY
- Temporary storage data sharing: DFHTST TYPE=SHARED
- End of temporary storage table: DFHTST TYPE=FINAL (see TYPE=FINAL (end of table))

Control section: DFHTST TYPE=INITIAL

The entry point and the beginning address for the temporary storage table being defined are established by the DFHTST TYPE=INITIAL macro.



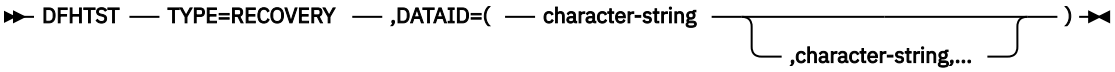
For general information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see TYPE=INITIAL (control section).

MIGRATE
Specify MIGRATE when you are assembling your TST for migration to the CSD file, or when you are using a TST in combination with TSMODEL resource definitions. When you specify a TST suffix using the TST system initialization parameter, if the TST is assembled with the MIGRATE option, CICS also processes TSMODEL resource definitions. If the TST is not assembled with the MIGRATE option, CICS loads the TST only and does not provide any RDO support for TS queues, and any attempts to install TSMODEL resource definitions are rejected.

TSAGE={0|number}
Defines the aging limit of temporary storage data used by the temporary storage domain during emergency restart of CICS. Data that has not been referenced for the specified interval is not recovered. The value is specified in days with a maximum value of 512. A value of zero indicates that no data is to be purged on this basis.

Recoverable temporary storage: DFHTST TYPE=RECOVERY

The DFHTST TYPE=RECOVERY macro specifies the generic names used for temporary storage queues for which recovery is applicable.



TYPE=RECOVERY
Code this to identify the temporary storage queue names that are recoverable. If a temporary storage queue name is such that it is defined by both a remote **and** a recovery DATAID, it is considered to be remote. Recoverability can only be specified in the CICS region in which the queue is local.

Note: TYPE=ENTRY is retained for compatibility with previous releases, and means exactly the same as TYPE=RECOVERY.

DATAID=(*character-string[,character-string,...]*)|()
Code this with one or more alphanumeric TS queue names that you want to be recoverable, where each name can be up to 8-characters in length. (See "TST: temporary storage table" on page 416 for information about generic names and matching criteria.)

character-string
Each character string can represent a generic queue name, or a unique TS queue name. Generic names are specified using 1 to 7 leading characters of TS queue names. DATAIDs that use all 8 characters define unique queues names.

Multiple names must be enclosed in parentheses, and separated by a comma. You can omit the parentheses if you specify only one name. Some CICS-generated TS queue names that you should consider for recovery are:

- **DF** refers to temporary storage queues used by CICS interval control for START commands with data, but which do not specify a REQID.
- ****** refers to temporary storage queues used by the BMS ROUTE command, and to those commands that use the PAGING operand.
- **$$** refers to temporary storage queues used by the BMS CMSG transaction when the PROTECT=YES option is specified on a START TRANSID command.

()
This special (null) operand, without any value between the parentheses, is interpreted by CICS as specifying any queue that is not more explicitly specified by other DATAIDs.

Note:

1. If a TST is generated with no TYPE=RECOVERY entries, no recovery processing is performed. If an EXEC CICS START command is issued with any of the FROM, RTRANSID, RTERMID, or QUEUE parameters specified, and a REQID is not specified, CICS generates request identifications starting with the prefix “DF”. If recovery is required for these requests, the TST should be generated with the corresponding generic name.
2. All temporary storage queues used by restartable transactions (those defined with RESTART(YES) in the transaction resource definition) should be made recoverable (including those with the default DF prefix).
3. Only data on auxiliary storage can be made recoverable. Data written to main storage is not recoverable, regardless of any recovery options that you may specify.
4. When a task modifies temporary storage data designated as recoverable, the data is protected from modification by a concurrent task by enqueuing on the queue name. The queue name is not dequeued until the task terminates or issues a task syncpoint request to designate the end of a logical unit of work. At this time a log record is written to the system log data set to provide external information sufficient to recover the data if the system subsequently terminates abnormally.

Example

This DFHTST TYPE=RECOVERY macro defines recoverable temporary storage queues:

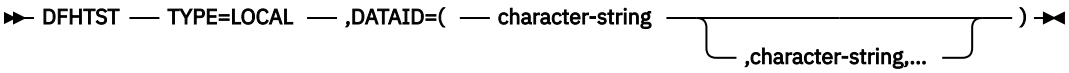
```
DFHTST TYPE=RECOVERY,
       DATAID=(DF,**,
               $$(,character-string)...)
```

- The DATAID DF makes the temporary storage queues used on CICS start requests recoverable.
- The DATAIDs ** and $$ make the default temporary storage queues used by BMS recoverable.
- The DATAID character string represents the leading characters of each temporary storage queue identifier that you want to be recoverable. For example, DATAID=(R,ZIP) makes recoverable all temporary storage queues that have identifiers starting with the character “R” or the characters “ZIP”.

Local temporary storage: DFHTST TYPE=LOCAL

The DFHTST TYPE=LOCAL macro defines temporary storage queue names that reside in the local CICS region in which the TST is installed. This macro enables you to define local queues without knowing the SYSIDNT (see the SYSIDNT option on the DFHTST TYPE=REMOTE macro for more information).

Used in conjunction with the all-generic DATAID specified on the TYPE=REMOTE macro for remote and shared queues, this macro can help you to simplify greatly the task of defining local and remote queues.



TYPE=LOCAL
Indicates that this TST entry defines a set of local temporary storage queues.

DATAID=(*character-string[,character-string,...]*)|()
Code this with one or more alphanumeric TS queue names, where each name can be up to 8-characters in length.

character-string
Each character string can represent a generic queue name, or a unique TS queue name. Typically, generic names are specified using 1 to 7 leading characters of TS queue names. DATAIDs that use all 8 characters define unique queue names.

Multiple names must be enclosed in parentheses, and separated by a comma. You can omit the parentheses if you specify only one name.

()
This special (null) operand, without any value between the parentheses, is interpreted by CICS as specifying any queue that is not more explicitly specified by other DATAIDs. You can use this as a catch-all in the following way:

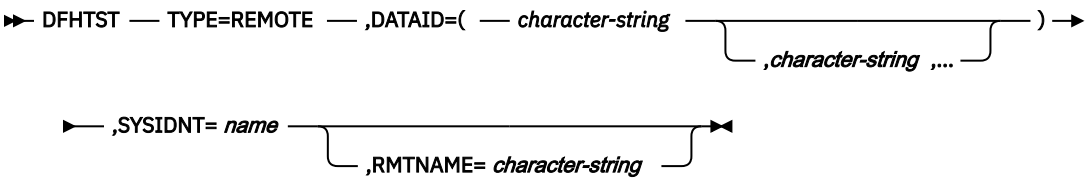
- If certain queues, which reside either in another region or in a shared TS pool, are specified on a TYPE=REMOTE macro with suitable generic DATAIDs, you can define all other queues as local by specifying DATAID=() on the TYPE=LOCAL macro.

This null option on the TYPE=LOCAL macro is mutually exclusive with DATAID=() on the TYPE=REMOTE macro, and the TST macro returns an assembly error if it is specified on both local and remote entries. Thus, if you specify DATAID=() on local TS queue entries, the TYPE=LOCAL macros must follow all TYPE=REMOTE macros.

Remote temporary storage: DFHTST TYPE=REMOTE

The DFHTST TYPE=REMOTE macro defines temporary storage queue names that reside in remote CICS regions when CICS intercommunication facilities are being used.

Use this macro also to define queues residing in a shared queue pool, which is treated like a remote region except that the name of the remote system matches the system name on a DFHTST TYPE=SHARED macro.



TYPE=REMOTE
Indicates that this TST entry defines a set of remote temporary storage queues, which can reside either in a remote CICS region or in a shared TS pool in a coupling facility.

DATAID=(*character-string[,character-string,...]*)|()
Code this with one or more alphanumeric TS queue names, where each name can be up to 8 characters in length. Use 1 to 7 leading characters of TS queue names to form generic names of

those queues for which requests are to be routed to a remote region or to a TS server. (See "TST: temporary storage table" on page 416 for information about generic names and matching criteria.)

Note: You cannot use the list form of the DATAID operand when RMTNAME is specified. If you specify the RMTNAME parameter, the syntax for DATAID is DATAID=*character-string*.

character-string
Each character string can represent a generic queue name, or a unique TS queue name. Typically, generic names are specified using 1 to 7 leading characters of TS queue names. The generic names are those used by application programs in the region in which this TST is installed.

Multiple names must be enclosed in parentheses, and separated commas. You can omit the parentheses if you specify only one name.

()
This special operand, without any value between the parentheses, is interpreted by CICS as specifying any queue that is not more explicitly specified by other DATAIDs. You can use this as a catch-all in the following way:

- If the queues with names beginning with letters L, M, and N are local, and these are specified on a TYPE=LOCAL macro with suitable generic DATAIDs, you can define all other queues as remote by specifying DATAID=() on the TYPE=REMOTE macro, as follows:

```
       DFHTST TYPE=LOCAL,       *
              DATAID=(L,M,N)
*
       DFHTST TYPE=REMOTE,      *
              DATAID=()
```

The DATAID=() option on the TYPE=REMOTE macro is mutually exclusive with DATAID=() on the TYPE=LOCAL macro, and the TST macro returns an assembly error if it is specified on both local and remote entries.

DATAID=() must be the last entry in a set of local and remote entries. Thus, if you use DATAID=() on remote TS queue entries, the TYPE=REMOTE macros must follow any TYPE=LOCAL macros.

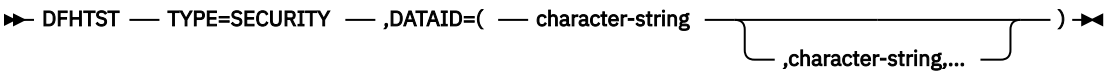
SYSIDNT=*name*
Identifies the region or server in which the remote or shared temporary storage queues reside. For a remote queue owned by another CICS region, the 4-character alphanumeric name specified must be the same as a REMOTENAME option specified in the CONNECTION definition, the first 4 chararcters of the IPCONN name on an IPCONN definition, or the SYSIDNT name specified on a DFHTST TYPE=SHARED entry.

You can use this parameter to specify the name of the local region in which the TST is installed. When the SYSIDNT operand matches the SYSIDNT specified on the system initialization parameter, the TS queues that match the DATAIDs are treated as local queues.

RMTNAME=*character-string*
Code this with the 1- to 8-character prefix that is to be used by CICS to replace that specified in the DATAID operand when a reference to the temporary storage queue is transmitted to a remote system or region. This operand defaults to the character string specified in the DATAID operand. The length of the character string specified in this operand must be the same as the length of the character string in the DATAID operand. This mechanism allows access to a temporary storage queue in the remote system with the same name as one in the local system.

Temporary storage security checking: DFHTST TYPE=SECURITY

The DFHTST TYPE=SECURITY macro indicates that security checking is required for the temporary storage queues specified in the TST.



TYPE=SECURITY
Indicates that this TST entry defines a set of temporary storage queues that require security checking.

DATAID=(*character-string[,character-string,...]*)|()
Code this with one or more alphanumeric TS queue names, where each name can be up to 8-characters in length. Use 1 to 7 leading characters from the leading characters of queue names to form generic names of those queues that are subject to security checking. (See "TST: temporary storage table" on page 416 for information about generic names and matching criteria.)

Note:

1. When this macro is used, a suitable profile must be defined to the external security manager to control access to the TSQ. Otherwise, the macro will not have the intended effect. See Refreshing resource profiles in main storage for information about profiles.
2. The full TSQ name is passed to the security manager.

character-string
Each character string can represent a generic queue name, or a unique TS queue name. Typically, generic names are specified using 1 to 7 leading characters of TS queue names. The generic names are those used by application programs in the region in which this TST is installed.

Multiple names must be enclosed in parentheses, and separated by a comma. You can omit the parentheses if you specify only one name.

()
This null operand, without any value between the parentheses, is interpreted by CICS as specifying any queue that is not more explicitly specified by other DATAIDs.

Temporary storage data sharing: DFHTST TYPE=SHARED

The DFHTST TYPE=SHARED macro specifies the remote system name by which CICS identifies a temporary storage pool in the coupling facility.

TYPE=SHARED
indicates that this TST entry defines a mapping between a system identifier (SYSIDNT) specified on a TYPE=REMOTE entry and a pool of TS data sharing queues.

SYSIDNT=*system_name*
specifies the 1- to 4-character system name that corresponds to a TS pool name.

CICS uses this SYSIDNT to map remote queues (defined by a TYPE=REMOTE entry, or an explicit SYSID on an API command) to a TS server, as follows:

- If an API temporary storage command specifies a remote queue explicitly (by means of the SYSID option), CICS maps the SYSID to a matching SYSIDNT on a TYPE=SHARED entry:
  - If a matching SYSIDNT is found, CICS uses the corresponding POOL name to identify the TS server that manages the shared TS queue.
  - If the SYSID does not match any TYPE=SHARED entry, the request is function shipped to the remote queue-owning region (QOR) named by SYSID.
- If an API temporary storage command references a remote queue identified by a TYPE=REMOTE entry, CICS checks for a matching SYSIDNT in the TYPE=SHARED entries:
  - If a TYPE=SHARED entry with a matching SYSIDNT is found, CICS uses the corresponding POOL name to identify the TS server that manages the shared TS queue.
  - If a TYPE=SHARED entry is not found, the queue is a remote queue and the request is function shipped to the QOR.

You can create multiple TYPE=SHARED entries, with different SYSIDNT names, that refer to the same POOL name. In this case, references to the same queue name refer to the same queue name regardless of which SYSID is used (on the API).

POOL=*pool_name*
specifies the 1- to 8-character name of the pool of TS queues that is to be used for TS requests that specify, implicitly or explicitly, the corresponding system name. The pool-name must match the name specified on the POOL parameter of the TS server that manages the TS pool.

DFHTST example

An example of how to code a temporary storage table (TST).

```
     DFHTST TYPE=INITIAL,             LIST OF GENERIC NAMES OF QUEUES *
            SUFFIX=01                 THAT ARE RECOVERABLE, REMOTE,
*                                     SHARED, LOCAL, OR REQUIRE
*                                     SECURITY CHECKING.
*
*  The following macro specifies that all LOCAL queues with
*  names beginning with the letter 'R' are RECOVERABLE:
*
     DFHTST TYPE=RECOVERY,                                           *
            DATAID=R
*
*  The following macro specifies that queues with names
*  beginning with C,D,E, and X are local queues:
*
     DFHTST TYPE=LOCAL,                                              *
            DATAID=(C,D,E,X)
*
*  The following macro specifies that queues with names
*  beginning with AB,L,M,N are remote queues on system RSYS:
*
     DFHTST TYPE=REMOTE,                                             *
            DATAID=(AB,L,M,N),                                       *
            SYSIDNT=RSYS,                                            *
*
*  The next macro specifies that all queues not local as defined
*  above, or remote in system RSYS as defined above, are remote
*  queues that reside in a shared TS pool TYPE=SHARE macro.
*
     DFHTST TYPE=REMOTE,                                             *
            DATAID=(),                                               *
            SYSIDNT=SHR1
*
*  The next macro specifies that remote queues with SYSIDNT=SHR1
*  are mapped to shared TS pool named TSQSHR1.
*
     DFHTST TYPE=SHARED,                                             *
            SYSIDNT=SHR1,                                            *
            POOL=TSQSHR1
```

```
*
*  The following macro specifies that queues with names
*  beginning with SAQ require security checking.
*    Note that the full TS queue name is passed to the ESM.
*
     DFHTST TYPE=SECURITY,                                           *
            DATAID=SAQ
*
     DFHTST TYPE=FINAL
     END
```

Figure 17. Temporary storage table: example

XLT: transaction list table

The transaction list table specifies transactions that can be initiated from terminals during system termination, and groups of transactions that you want to enable or disable together. Use a TRANSACTION resource in preference to XLT.

The XLT can be used to define:

- A list of transaction identifications that can be initiated from terminals during the first quiesce stage of system termination. If there are no PLT programs to execute, the first quiesce time can be short, thus giving little time to enter any XLT program before going into the second quiesce stage. You specify the

suffix of the table to be used with the XLT system initialization parameter. The main terminal operator can change the suffix at system termination, using the XLT option of the **CEMT PERFORM SHUTDOWN** command.

Note: As an alternative, you can create a PROGRAM resource to define the transaction list table. Defining it as a program also means that it can be autoinstalled; see Autoinstalling programs, map sets, and partition sets in Configuring for information on autoinstall for programs.

- A group of transaction identifications to be disabled or enabled through the main terminal. The main terminal operator specifies the suffix of the table to be used, using the CLASS option of the CEMT SET TRANSACTION.

Figure 18 on page 425 illustrates the coding to create a XLT.

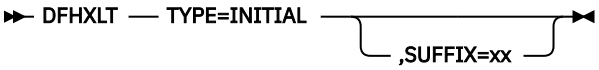
The following macros are available to define the XLT entries:

- Control section: DFHXLT TYPE=INITIAL
- Entries in transaction list table: DFHXLT TYPE=ENTRY
- End of transaction list table: DFHXLT TYPE=FINAL (see TYPE=FINAL (end of table))

Control section: DFHXLT TYPE=INITIAL

The DFHXLT TYPE=INITIAL macro establishes the entry point and start address of the XLT being defined.

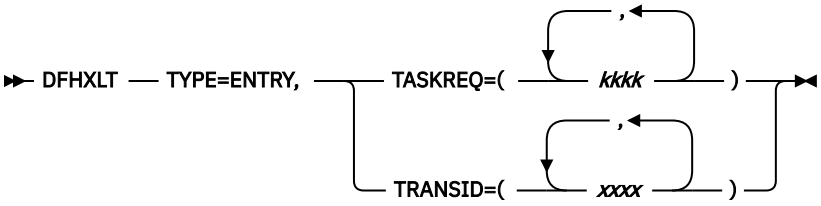
The DFHXLT TYPE=INITIAL macro establishes the entry point and start address of the XLT being defined.



For general information about TYPE=INITIAL macros, including the use of the SUFFIX operand, see TYPE=INITIAL (control section).

Entries in transaction list table: DFHXLT TYPE=ENTRY

The DFHXLT TYPE=ENTRY macro specifies a list of transaction identifications that can be initiated from terminals during the first quiesce stage of system termination.



TYPE=ENTRY
Code this if one or more entries are to be generated in the XLT.

TASKREQ=(kkkk[,kkkk],...)
kkkk can be one of the following:

- PA1 through PA3, and PF1 through PF24 indicates one of the special 3270 keys that can be used to initiate a task.
- LPA (light pen attention) indicates that a transaction is to be initiated when a light pen detectable field is selected.
- OPID (operator identification card reader) indicates that a transaction is initiated when the appropriate operator's identity badge has been read in.
- MSRE indicates that transactions are initiated when the 10/63 character magnetic slot reader is used.

Define each TASKREQ on the CSD file, and install it in the running system. (For further information, see the description of the TASKREQ attribute in TRANSACTION attributes.)

TRANSID=(xxxx[,xxxx],...)
Represents a 1- to 4-character transaction code. Define each TRANSID on the CSD file, and install it in the running system. (For further information, see the description of the TRANSACTION attribute in TRANSACTION attributes)

If the TRANSID contains a special character (for example, a comma), the TYPE=ENTRY instruction must contain only one TRANSID with quotation marks as delimiters.

Note: TASKREQ and TRANSID are mutually exclusive parameters.

DFHXLT example

An example of coding the a transaction list table (TLT).

```
   DFHXLT TYPE=INITIAL,                    LIST OF TRANSACTIONS     *
          SUFFIX=IN                        THAT ARE ACCEPTED
*                                          DURING THE FIRST QUIESCE
*                                          PHASE OF SYSTEM
*                                          TERMINATION.
   DFHXLT TYPE=ENTRY,TASKREQ=PF5           (TASKREQ MUST ALSO BE
*                                          DEFINED IN THE CSD AND
*                                          INSTALLED IN THE RUNNING
*                                          CICS SYSTEM. AN ENTRY FOR
*                                          THE XLT MUST BE MADE IN
   DFHXLT TYPE=ENTRY,TRANSID=(USR1,USR2)   THE CSD.)
   DFHXLT TYPE=ENTRY,TRANSID='AA,1'
   DFHXLT TYPE=ENTRY,TRANSID='AA,2'
   DFHXLT TYPE=FINAL
   END

   DFHXLT TYPE=INITIAL,                    LIST OF LOGICALLY RELATED*
          SUFFIX=G1                        TRANSIDS TO BE ENABLED OR
*                                          DISABLED BY MAIN
*                                          TERMINAL.
   DFHXLT TYPE=ENTRY,TRANSID=(TSSA,TSRA)   (TRANSIDS MUST ALSO BE
   DFHXLT TYPE=ENTRY,TRANSID=(TDSA,TDRA)   DEFINED IN THE CSD AND
   DFHXLT TYPE=ENTRY,TRANSID=ICSA          INSTALLED IN THE RUNNING
   DFHXLT TYPE=FINAL                       CICS SYSTEM.)
   END
```

Figure 18. Transaction list table: example

Chapter 6. Obsolete attributes retained for compatibility

Some resource definition attributes do not apply to resources used in CICS Transaction Server for z/OS, Version 5 Release 4, but are supported to provide CSD compatibility for earlier releases of CICS where they are still valid.

For information about compatibility mode, see Sharing the CSD between different releases of CICS.

The following attributes do not apply to resources used in this CICS release. Check the documentation for the earlier CICS releases that you are using to see whether the attribute is supported in that release.

Table 21. Obsolete resource definition attributes

Resource	Attribute	Description
CONNECTION	BINDPASSWORD	APPC bind-time security password
FILE	RESSECNUM	Resource security value
FILE	PASSWORD	File access verification
LSRPOOL	LSRPOOLID	Numeric identifier 1-8 for pool
MAPSET	RSL	Resource security value
PARTITIONSET	RSL	Resource security value
PROFILE	PROTECT	Message integrity and message logging
PROGRAM	HOTPOOL	Java hotpooling
PROGRAM	JVMPROFILE	JVM profile for pooled JVMs
PROGRAM	RSL	Resource security value
SESSIONS	INSERVICE	Session can be used
SESSIONS	OPERID	3-character operator ID
SESSIONS	OPERPRIORITY	Operator priority code
SESSIONS	OPERRSL	Resource security key
SESSIONS	OPERSECURITY	Preset transaction security keys
SESSIONS	RECOVNOTIFY	Notification of XRF takeover
SESSIONS	TRANSACTION	Allow only specified transaction
TCPIPSERVICE	DNSGROUP	Specifies the group name with which CICS registers to Workload Manager
TCPIPSERVICE	GRPCRITICAL	Marks the service as a critical member of the DNS group
TCPIPSERVICE	PRIVACY	Level of SSL encryption required
TCPIPSERVICE	TSQPREFIX	TS queue prefix
TERMINAL	BINDPASSWORD	APPC bind-time security password
TERMINAL	CONSOLE	Console identification number
TERMINAL	OPERID	3-character operator ID

© Copyright IBM Corp. 1974, 2023

Table 21. Obsolete resource definition attributes (continued)		
Resource	**Attribute**	**Description**
TERMINAL	OPERPRIORITY	Operator priority code
TERMINAL	OPERRSL	Resource security key
TERMINAL	OPERSECURITY	Preset transaction security keys
TRANSACTION	EXTSEC	Use an external security manager
TRANSACTION	INDOUBT	Action at abend
TRANSACTION	PRIMEDSIZE	Primed storage allocation size
TRANSACTION	RSL	Resource security value
TRANSACTION	TCLASS	Class assigned to task
TRANSACTION	TRANSEC	Transaction security value
TYPETERM	LOGMODECOM	LOGMODE compatibility
TYPETERM	XRFSIGNOFF	Force sign-off after an XRF takeover

Chapter 7. CICS-supplied resource definitions, groups, and lists

IBM supplies definitions of resources that must be installed on your CICS region, and definitions of resources used by the sample application programs.

For details of transactions supplied by CICS, see List of CICS transactions.

You initialize the CSD file by using the DFHCSDUP INITIALIZE command. Following initialization, the CSD file contains two categories of resource definition groups:

1. Groups named in DFHLIST, essential for using RDO and other CICS-supplied transactions
2. Groups of definitions for the sample application programs. For details, see "Sample application program groups" on page 437.

The CICS transactions, the TYPETERM definitions, model TERMINAL definitions, and PROFILE definitions that are supplied by IBM are in four groups:

- DFHTYPE: TYPETERM definitions
- DFHTERM: model TERMINAL definitions for automatic installation
- DFHISC: PROFILE definitions for intersystem communication sessions
- DFHSTAND: PROFILE definitions

DFHLIST definitions

DFHLIST contains definitions of resources that are required.

Table 22. DFHLIST resource definitions

Group name	Description	Programs	Transactions	Other resources
DFH$SOT	Supplied TCPIPSERVICE resources			**TCP/IP Services:** ECI HTTPNSSL HTPSSL
DFHBMS	Basic mapping support	DFHTPQ DFHTPR DFHTPS	CSPG CSPQ CSPS	
DFHBR	Bridge programs	DFHL3270 DFHBRMP DFHBRCV		
DFHCBTS	Local request queue file for BTS **Note:** This group is not protected by a lock. The definitions that it contains can be modified if required.			**File:** DFHLRQ
DFHCFC	Programs needed for CICS C++ foundation classes	ICCFCDLL		
DFHCLNT	CICS client CTIN and CCIN	DFHZCT1 DFHZCN1	CTIN CCIN	**Tranclass:** DFHCOMCL
DFHCONS	Write to CPU console	DFHCWTO	CWTO	
DFHDBCTL	DBCTL transactions	DFHDBAT DFHDBCON DFHDBCT DFHDBDI DFHDBDSC DFHDBIQ DFHDBME DFHDBMP DFHDBUEX	CDBC CDBD CDBI CDBM CDBN CDBO CDBT	**File:** DFHDBFK **Mapsets:** DFHDBIE DFHDBNE

© Copyright IBM Corp. 1974, 2023

Table 22. DFHLIST resource definitions (continued)

Group name	Description	Programs	Transactions	Other resources
DFHDB2	DB2 support	DFHD2CM0 DFHD2CM1 DFHD2CM2 DFHD2CM3 DFHD2EDF DFHD2EX1 DFHD2EX2 DFHD2INI DFHD2PXT DFHD2SPS DSNCUEXT DSNTIAC DSNTIA1	CDBF CDBQ CEX2 DSNC	
DFHDCTG	TD queues for basic CICS facilities **Note:** This group is not protected by a lock. The definitions that it contains can be modified if required.			**TD queues:** CADL CADO CADS CAIL CCPI CCSE CCSO CCZM CDB2 CDBC CDEP CDUL CECO CEJL CEPO CESE CESO CIEO CIIL CISL CISO CJRM CKQQ CMIG CMPO CMQM CMLO CPIO CRDI CRLO CRPO CSBA CSBR CSCC CSCS CSDH CSDL CSFL CSJE CSJO CSKL CSLB CSML CSMT CSNE CSOO CSPL CSQL CSRL CSSH CSSL CSTL CSZL CSZX CWBO CWBW
DFHDOC	CICS document handler template reader	DFHDHEI		
DFHDP	Application debugging profile manager 3270 interface	DFHDPLU DFHDPIN DFHDPCP	CADP CIDP	**Mapset:** DFHDPMS
DFHDPWB	Application debugging profile manager Web interface	DFHDPWB DFHDPWM0 DFHDPWM1 DFHDPWM2 DFHDPWM3 DFHDPWM4 DFHDPWM5 DFHDPWM6 DFHDPWT0 DFHDPWT1 DFHDPWT2 DFHDPWT3 DFHDPWT4 DFHDPWT5 DFHDPWT6 DFHDPWF0		
DFHEDF	Execution diagnostic facility	DFHDBMS DFHDBTI DFHEDFBR DFHEDFD DFHEDFP DFHEDFR DFHEDFX DFHEIGDS DFHEITAB	CEBR CEDF CEDG CEDX CEDY	**Mapset:** DFHEDFM **Transclass:** DFHEDFTC DFHEDFTO
DFHEDP	EXEC DLI user exit	DFHEDP		

Table 22. DFHLIST resource definitions (continued)

Group name	Description	Programs	Transactions	Other resources
DFHEP	Event processing	DFHECEAH DFHECEAM DFHECEAS DFHECEAT	CEPH CEPQ CEPS CEPT	**Profile:** DFHECEPH
DFHFCRL	Disable file in bundle	DFHFCRN	CFCR	
DFHFE	FE terminal test facility	DFHFEP DFHTRAP	CSFE	
DFHFEPI	Front End Programming Interface	DFHEITSZ DFHSZRMP	CSZI	
DFHHARDC	3270 printer - z/OS Communications Server	DFHP3270	CSPP	
DFHINDT	Indoubt test tool	DFHINDAP DFHINDT DFHINTRU	CIND	**Tranclass:** DFHTCIND
DFHINQUI	Command definition	DFHEITBS		
DFHINTER	Command interpreter	DFHECID DFHECIP DFHECSP	CECI CECS	
DFHIPECI	ECI over TCP/IP	DFHIEP	CIEP	
DFHISC	Intersystem communication	DFHCCNV DFHCHS DFHCLS3 DFHCLS4 DFHCLS5 DFHCNV DFHCRNP DFHCRQ DFHCRR DFHCRS DFHCRSP DFHCRT DFHDFST DFHDSRP DFHDYP DFHLUP DFHMIRS DFHMXP DFHRTC DFHRTE DFHSHRRP DFHSHRSP DFHUCNV DFHZLS1	CDFS CEHP CEHS CLQ2 CLR2 CLS1 CLS2 CLS3 CLS4 CMPX CPMI CQPI CQPO CRSQ CRSR CRTE CRTX CSHR CSMI CSM1 CSM2 CSM3 CSM5 CSNC CSSF CVMI CXRT	**Profiles:** DFHCICSF DFHCICSR DFHCICSS
DFHISCIP	IP interconnectivity support	DFHCIS4 DFHISAIP DFHISCIP DFHISCOP DFHISDIP DFHISEMP DFHISLQP DFHISIP DFHISPHP DFHISPRP DFHISREU DFHISREX DFHISRRP DFHISRSP	CISB CISC CISD CISE CISM CISP CISR CISQ CISS CIST CISU CISX CIS1 CIS4	**Profile:** DFHCICSC **TS model:** DFHISLQ
DFHJAVA	Programs required for Java support	DFHSJJI DFHSJITL DFHSJTHP	CJSA CJSR CJSU	
DFHLGMOD	CICS log manager			**Journalmodels:** DFHLOG DFHSHUNT DFHLGLOG
DFHLGQC	CICS log manager quiesce	DFHLGQC	CSQC	
DFHMISC	Miscellaneous programs	DFHLETRU DFHPEP DFHREST		

Table 22. DFHLIST resource definitions (continued)

Group name	Description	Programs	Transactions	Other resources
DFHMQ	Programs, transactions, and mapsets required for WebSphere MQ support	CSQCAPX CSQAVICM CSQCBDCI CSQCBE30 CSQCBP10 CSQCBP53 CSQCBP00 CSQCBP10 CSQCBR00 CSQCBR53 CSQCCB CSQCCLOS CSQCCONN CSQCCONX CSQCCTL CSQCDISC CSQCDSC CSQCDSPL CSQCQCON CSQCSSQ CSQCGET CSQCINQ CSQCOPEN CSQCPUT CSQCPUT1 CSQCRST CSQCSET CSQCSSQ CSQCSUB CSQCSUBR CSQFCTAB DFHMQBAS DFHMQBP0 DFHMQBP1 DFHMQBP2 DFHMQBP3 DFHMQBR0 DFHMQCOD DFHMQCON DFHMQCTL DFHMQDCI DFHMQDIS DFHMQDSC DFHMQDSL DFHMQMON DFHMQPLT DFHMQPOP DFHMQPRM DFHMQPUL DFHMQQCN DFHMQRET DFHMQRS DFHMQSSQ DFHMQTRU DFHMQTSK IMQB23IC IMQS23IC	CKAM CKBC CKBM CKBP CKBR CKCN CKDL CKDP CKQC CKRS CKRT CKSD CKSQ CKTI	**Mapsets:** DFHMQHC DFHMQHE DFHMQHK DFHMQHU DFHMQ1C DFHMQ1E DFHMQ1K DFHMQ1U DFHMQ2C DFHMQ2E DFHMQ2K DFHMQ2U
DFHMSWIT	Message switching program	DFHMSP	CMSG	
DFHOPCLS	Dynamic open and close program	DFHFCU	CSFU	

Table 22. DFHLIST resource definitions (continued)

Group name	Description	Programs	Transactions	Other resources
DFHOPER	Operator programs	DFHCEMNA DFHCEMNB DFHCEMNC DFHCEMND DFHCETRA DFHCETRB DFHCETRC DFHCETRD DFHCETRF DFHECBAM DFHEITMT DFHEITOT DFHEITST DFHEMTA DFHEMTD DFHEMTP DFHEOTP DFHESTP DFHLDMAP DFHSOCRL	CBAM CCRL CEMN CEMT CEOT CEST CETR CLDM	**Mapsets:** DFHSO1M DFHCMNH DFHCMNM DFHCTRH DFHCTRM
DFHOTS	OTS resynchronization	DFHOTR	CJTR	
DFHPGAIP	Autoinstall for programs	DFHPGADX DFHPGAHX DFHPGALX DFHPGAOX DFHPGAPG		**Mapset:** DFHPGAMP **Partition set:** DFHPGAPT
DFHPIPE	Web services processing	DFHJSON DFHMLBSJ DFHMLBST DFHPIAP DFHPIDSH DFHPIDSQ DFHPIEP DFHPIIR DFHPILSQ DFHPIR DFHPIRT DFHPISN1 DFHPISN2 DFHPITE DFHPITP DFHPITQ1 DFHWSADH DFHWSSE1 DFHWSXXX C128N IOSTREAM IXM4C57 IXMI33UC IXMI33DA IXMI33IN	CPIA CPIH CPIL CPIS	
DFHPIVAL	SOAP message validator **Note:** This group is not protected by a lock. The definitions that it contains can be modified if required.	DFHPIVAL		
DFHPSSGN	Persistent sessions sign-on retention	DFHZSGN DFHZPCT DFHZRPT	CPSS CPCT CRTP	
DFHRL	Resource lifecycle management	DFHRLMF DFHRLR DFHRLSC DFHRLVC	CRLR	
DFHRMI	Resource manager interface	DFHRMSY	CRSY	
DFHRQS	Request stream join program	DFHRZJN		
DFHRS	Region status, file creation	DFHRSFDL		
DFHRSEND	z/OS Communications Server resend program	DFHZRSP	CSRS	
DFHSDAP	SHUTDOWN ASSIST	DFHCESD	CESD	

Table 22. DFHLIST resource definitions (continued)

Group name	Description	Programs	Transactions	Other resources
DFHSIGN	Sign-on and sign-off programs and table	DFHCESC DFHCEGN DFHSFP DFHSNP	CESC CEGN CESF CESN CESL	**Mapsets:** DFHSNLE DFHSNPE DFHSNSE
DFHSO	Emulate CICS sockets interface	DFHSOCI DFHSOLI		
DFHSPI	Resource definition online	DFHAMP DFHDMP DFHEDAD DFHEDAP DFHEITSP DFHPUP DFHTBS DFHTOR DFHZATA DFHZATD DFHZATDX DFHZATMD DFHZATMF DFHZATR DFHZATS DFHZCQ DFHZCTDX DFHZDTDX DFHZPTDX	CATA CATD CATR CATS CDTS CEDA CEDA CEDB CEDC CFTS CITS CMTS CRMD CRMF	
DFHSTAND	Standard CICS application programs	DFHACP DFHCXCU DFHPIITL DFHPIPA DFHPIXC DFHPSIP DFHQRY DFHSTP DFHTACP DFHTEP DFHTEPT DFHTFP DFHZXCU DFHZXRE DFHZXST	CPIR CQRY CSAC CSTE CXCU CXRE	**Profiles:** DFHCICSA DFHCICSE DFHCICSP DFHCICST DFHCICSV DFHPPF01 DFHPPF02 **Mapset:** DFHXMSG
DFHTCL	Compatibility TRANCLASS definitions			**Tranclasses:** DFHTCL00 DFHTCL00 DFHTCL01 DFHTCL03 DFHTCL04 DFHTCL05 DFHTCL06 DFHTCL07 DFHTCL08 DFHTCL09 DFHTCL10 DFHTCLQ2 DFHTSDEL
DFHTERM	Model TERMINAL definitions			**Terminals:** LU2 LU3 APPC SCSP 3270 3284 L0E2 L0M2 L0M3 L0M4 L0M5 L2E2 L2M2 L2E3 L2M3 L2E4 L2M4 L2M5

Table 22. DFHLIST resource definitions (continued)

Group name	Description	Programs	Transactions	Other resources
DFHTYPE	TYPETERM definitions			**Typeterms:** DFHCONS DFHLU2 DFHLU3 DFHLU62T DFHSCSP DFH3270 DFH3270P DFHLU0E2 DFHLU0M2 DFHLU0M3 DFHLU0M4 DFHLU0M5 DFHLU2E2 DFHLU2M2 DFHLU2E3 DFHLU2M3 DFHLU2E4 DFHLU2M4 DFHLU2M5
DFHVTAM	z/OS Communications Server programs	DFHGMM DFHZNAC DFHZNEP	CSGM CSNE	
DFHVTAMP	z/OS Communications Server terminal control print key function	DFHCPY DFHEXI DFHPRK DFHRKB	CSCY CSPK CSRK	
DFHWEB	CICS Web support definitions	DFH$WB1A DFHWBA DFHWBAAX DFHWBADX DFHWBAHX DFHWBALX DFHWBAOX DFHWBA1 DFHWBBLI DFHWBCLI DFHWBENV DFHWBEP DFHWBERX DFHWBGB DFHWBIMG DFHWBIP DFHWBLT DFHWBPA DFHWBPW DFHWBPW1 DFHWBPW2 DFHWBPW3 DFHWBPW4 DFHWBST DFHWBTC DFHWBTL DFHWBTRU DFHWBTTA DFHWBTTB DFHWBTTC DFHWBUN DFHWBXN	CWBA CWBC CWBG CWXN CWXU	**TSModels:** DFHWEB **Doctemplates:** DFHWBPW1 DFHWBPW2 DFHWBPW3 DFHWBPW4
DFHWEB2	Web 2.0 definitions	DFHW2A DFHW2FD DFHW2TS	CW2A	
DFHWU	CICS management client interface	DFHWUIPG DFHWUIPI DFHWUIP1 DFHWUIP3 DFHWUIP4 DFHWUIP5	CWWU	**Doctemplates:** DFHWUIPI DFHWUIP1 DFHWUIP3 DFHWUIP4 DFHWUIP5

Table 22. DFHLIST resource definitions (continued)				
Group name	**Description**	**Programs**	**Transactions**	**Other resources**
Note: The DFHTYPE TYPETERM definitions match the z/OS Communications Server-supplied LOGMODE definitions. If you are not using the supplied z/OS Communications Server LOGMODES, you might have to modify the DFHTYPE TYPETERM definitions. For programming information about z/OS Communications Server LOGMODE definitions, see Coding entries in the VTAM LOGON mode table.				

CICS-supplied groups not in DFHLIST

Not all CICS-supplied resource definition groups are defined in list DFHLIST.

Table 23. Resource definitions not in DFHLIST				
Group Name	**Description**	**Programs**	**Transactions**	**Other resources**
DFHBRCF	Coupling facility data table sample for DFHBRNSF			**File:** DFHBRNSF
DFHBRUT	User maintained data table sample for DFHBRNSF			**File:** DFHBRNSF
DFHBRVR	VSAM RLS sample for DFHBRNSF			**File:** DFHBRNSF
DFHBRVSL	VSAM non-RLS local sample for DFHBRNSF			**File:** DFHBRNSF
DFHBRVSR	VSAM non-RLS remote sample for DFHBRNSF			**File:** DFHBRNSF
DFHCMAC	CICS online messages and codes	DFHCMAC	CMAC (alias CHLP)	**Mapset:** DFHCMCM **File:** DFHCMACD
Note: DFHCMAC is not included in DFHLIST for reasons of compatibility with earlier releases. If you want to use the online messages and codes transaction, you must add DFHCMAC to your start-up group list.				
DFHMISC3	Miscellaneous group	DFHNET		
Note: DFHMISC3 is not included in DFHLIST for reasons of compatibility with earlier releases. If you require it in your system, add it to your start-up group list.				
DFHRPC	Remote procedure call	DFHRPAL DFHRPAS DFHRPC00 DFHRPMS DFHRPRP DFHRPTRU	CRPA CRPC CRPM	**Mapset:** DFHRP0
DFHTERMC	Model TERMINAL definition for console autoinstall			**Terminals:** AUTC

CICS-supplied compatibility groups

CICS supplies RDO groups for compatibility with earlier releases.

After you upgrade a CICS system definition data set (CSD), if you plan to share the CSD with earlier releases of CICS you must include the appropriate DFHCOMP*x* compatibility groups in your startup group list.

For more information about sharing the CSD between different releases of CICS, and which DFHCOMP*x* groups you must include for the earlier releases, see CSD compatibility between different CICS releases.

Do not attempt to share a CSD file with a CICS region that is running at a higher release level than the CSD file.

Sample application program groups

These resource definitions are required to run the sample application programs that are supplied with CICS. The groups are not named in DFHLIST.

Table 24. CICS sample applications - resource definitions

Group Name	Language	Description	Resources Defined
DFH$AFLA	Assembler	FILEA sample applications	**Map sets**: DFH$AGA DFH$AGB DFH$AGC DFH$AGD DFH$AGK DFH$AGL **Programs**: DFH$AALL DFH$ABRW DFH$ACOM DFH$AMNU DFH$AREN DFH$AREP **Transactions**: AADD ABRW AINQ AMNU AORD AORQ AREP AUPD
DFH$BMSP	COBOL and PL/I	BMS partition support applications	**Partitionset**: DFH0PS **Map sets**: DFH0CGP DFH$PGP **Programs**: DFH0CPKO DFH0CPLA DFH$PPKO DFH$PPLA **Transactions**: PPKO PPLA XPKO XPLA
DFH$CFLA	COBOL	FILEA sample applications	**Map sets**: DFH0CGA DFH0CGB DFH0CGC DFH0CGD DFH0CGK DFH0CGL **Programs**: DFH0CALL DFH0CBRW DFH0CCOM DFH0CMNU DFH0CREN DFH0CREP **Transactions**: ADDS BRWS INQY MENU OREN OREQ REPT UPDT
DFH$CNSL		Sample console definitions	**Typeterms**: DFH$JCLC DFH$CONS **Terminals**: CJCL CNSL CN02

Table 24. CICS sample applications - resource definitions (continued)			
Group Name	**Language**	**Description**	**Resources Defined**
DFH$CTXT	COBOL	CUA text model application	**Files**: DFH0FCAI DFH0FCUS DFH0FHLP **Map sets**: DFH0AB DFH0ABT DFH0BRW DFH0DEL DFH0FPD DFH0HLP DFH0HP DFH0HPD DFH0LST DFH0NEW DFH0OPN DFH0PRT DFH0SAS DFH0T1 DFH0UPD **Programs**: DFH0VAB DFH0VABT DFH0VBRW DFH0VDEL DFH0VDQ DFH0VHLP DFH0VHP DFH0VLIO DFH0VLST DFH0VNEW DFH0VOL DFH0VOPN DFH0VPRT DFH0VRIO DFH0VSAS DFH0VTBL DFH0VT1 DFH0VUPD **Profile**: DFH$CUA2 **Transactions:** AC2A AC2C AC2D AC2E AC2F AC20 AC21 AC22 AC23 AC24 AC25 AC26 AC27 AC28 DELQ
DFH$DFLA	C	FILEA sample applications	**Map sets**: DFH$DGA DFH$DGB DFH$DGC DFH$DGD DFH$DGK DFH$DGL **Programs**: DFH$DALL DFH$DBRW DFH$DCOM DFH$DMNU DFH$DREN DFH$DREP **Transactions**: DADD DBRW DINQ DMNU DORD DORQ DREP DUPD
DFH$DLIV		IMS installation verification procedure	**Programs**: DFH$DLAC DFH$DLAE DFH$DLCC DFH$DLCE DFH$DLPC DFH$DLPE **Transactions:** ASMC ASME COBC COBE PLIC PLIE
DFH$EXBS		Catalog application sample	**Files:** EXMPCAT EXMPCONF **Map sets**: DFH0XS1 DFH0XS2 DFH0XS3 **Programs**: DFH0XCMN DFH0XGUI DFH0XODE DFH0XSDS DFH0XSOD DFH0XSSM DFH0XVDS DFH0XWOD **Transactions**: ECFG EGUI

Table 24. CICS sample applications - resource definitions (continued)

Group Name	Language	Description	Resources Defined
DFH$EXCI		EXCI batch call interface samples	**Connections**: EXCG EXCS **Programs**: DFH$AXCS DFH$AXVS **Sessions**: EXCG EXCS **Transactions**: EXCI HPJC
DFH$EXWS		Catalog application sample	**Program**: DFH0XCUI **Transaction**: ECLI **TCP/IP service**: EXMPPORT **Pipelines**: EXPIPE01 EXPIPE02
DFH$FILA		FILEA samples	**File:** FILEA
DFH$ICOM	Assembler	Intersystem communication (ISC)	**Map sets**: DFH$IGB DFH$IGC DFH$IGS DFH$IGX DFH$IG1 DFH$IG2 **Programs**: DFH$ICIC DFH$IFBL DFH$IFBR DFH$IMSN DFH$IMSO DFH$IQRD DFH$IQRL DFH$IQRR DFH$IQXL DFH$IQXR **Transactions**: ICIC IFBL IFBR IMSN IMSO IQRD IQRL IQRR IQXL IQXR
DFH$NACT		NACT sample application	**Files**: ACCTFILE ACCTNAM ACINUSE **Map set**: DFH0MNA **Programs**: DFH0CNA1 DFH0CNA2 DFH0CNA3 DFH0CNA4 DFH0CNA5 **Transactions**: NACP NACT
DFH$OSGI		Sample applications for Java support using a JVM server	**Bundle**: DFH$OSGB **JVM server**: DFHJVMS **Programs**: DFH$JSAM DFH$LCCA DFJ$JHE1 DFJ$JHE2 DFJ$JPC1 DFJ$JPC2 DFJ$JTD1 DFJ$JTS1 DFJ$JTSC **TDQueue**: JTD1 **Transactions**: JHE1 JHE2 JPC1 JPC2 JTD1 JTS1

Table 24. CICS sample applications - resource definitions (continued)

Group Name	Language	Description	Resources Defined
DFH$PFLA	PL/I	FILEA sample applications	**Map sets**: DFH$PGA DFH$PGB DFH$PGC DFH$PGD DFH$PGK DFH$PGL **Programs**: DFH$PALL DFH$PBRW DFH$PCOM DFH$PMNU DFH$PREN DFH$PREP **Transactions**: PADD PBRW PINQ PMNU PORD PORQ PREP PUPD
DFH$STAT		Statistics	**Program**: DFH$STED
DFH$SXP		Message domain exits	**Programs**: DFH$SXP1 DFH$SXP2 DFH$SXP3 DFH$SXP4 DFH$SXP5 DFH$SXP6
DFH$UTIL	Assembler	Transient data utility dynamic allocation	**Programs**: DFH$TDWT DFH99 **Transactions**: ADYN TDWT
DFH$VTAM		Terminal definitions	**Typeterms**: DFH$L77 DFH$L78 DFH$L79 DFH$L86 **Terminals**: L77C L77D L78A L79A L86A
DFH$WLP		WebSphere Application Server Liberty sample definitions	**JVM server**: DFHWLP **Bundles**: JDBC WLPJDBC WLPHELLO WLPTSQ
DFHAI62		Starter APPC connections	**Program**: DFHZATDY **Connections**: CBPS CBSS CCPS **Sessions**: CBPS CBSS CCPS
DFHMROAR		Starter MRO systems	**Connections**: CICD CICT **Sessions**: CICSRD CICSRT
DFHMRODR		Starter MRO systems	**Connection**: CICA **Session**: CICSRA
DFHMROFA		Starter MRO systems	**File**: FILEA **Map sets**: DFH$AGA DFH$AGB DFH$AGC DFH$AGD DFH$AGK DFH$AGL **Programs**: DFH$AALL DFH$ABRW DFH$ACOM DFH$AMNU DFH$AREN DFH$AREP **Transactions**: AADD ABRW AINQ AMNU AORD AORQ AREP AUPD

Table 24. CICS sample applications - resource definitions (continued)			
Group Name	**Language**	**Description**	**Resources Defined**
DFHMROFD		Starter MRO systems	**File**: FILEA
DFHMROFT		Starter MRO systems	**Transactions**: AADD ABRW AINQ AMNU AORD AORQ AREP AUPD
DFHMROTR		Starter MRO systems	**Connection**: CICA **Session**: CICSRA

TYPETERM definitions in group DFHTYPE

The CICS-supplied CSD group DFHTYPE contains TYPETERM definitions.

DFHCONS

MVS console:

```
TYPETERM(DFHCONS)  GROUP(DFHTYPE)
DEVICE(CONSOLE)
PAGESIZE(1,124)    AUTOPAGE(NO)
BRACKET(YES)       BUILDCHAIN(YES)     ROUTEDMSGS(NONE)
UCTRAN(YES)
```

DFHLU2

SNA logical unit type 2 (3270 displays):

```
TYPETERM(DFHLU2)   GROUP(DFHTYPE)
DEVICE(LUTYPE2)    TERMMODEL(2)
DEFSCREEN(24,80)   PAGESIZE(24,80)     AUTOPAGE(NO)
BRACKET(YES)       BUILDCHAIN(YES)     ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)  EXTENDEDDS(YES)     UCTRAN(YES)
SENDSIZE(1536)     RECEIVESIZE(256)    IOAREALEN(256,4000)
ERRLASTLINE(YES)   ERRINTENSIFY(YES)   ATI(YES)  TTI(YES)
DISCREQ(YES)       RELREQ(YES)         AUTOCONNECT(YES)
LOGONMSG(YES)      QUERY(ALL)          CREATESESS(NO)
```

This definition is for a 3278 Model 2 display. It is suitable for the following devices: 3178, 3179, 3277, 3278, 3279, 3290, 3270PC, 3270PC/G, 3270PC/GX, 8775, and 5550.

DFHLU3

SNA logical unit type 3 (3270 printers):

```
TYPETERM(DFHLU3)   GROUP(DFHTYPE)
DEVICE(LUTYPE3)    TERMMODEL(2)
DEFSCREEN(24,80)   PAGESIZE(24,80)     AUTOPAGE(YES)
BRACKET(YES)       BUILDCHAIN(NO)      ROUTEDMSGS(ALL)
SENDSIZE(256)      RECEIVESIZE(256)    IOAREALEN(512,0)
EXTENDEDDS(YES)    QUERY(ALL)          ATI(YES)  TTI(YES)
DISCREQ(YES)       RELREQ(YES)         AUTOCONNECT(YES)
LOGONMSG(NO)                           CREATESESS(NO)
```

This definition is for a 3287 printer. It is suitable for the following devices: 3262, 3268, 3284, 3286, 3287, 3288, 3289, and 5550.

DFHSCSP

SNA logical unit type 1 (3270 SCS printers):

```
TYPETERM(DFHSCSP)  GROUP(DFHTYPE)
DEVICE(SCSPRINT)
PAGESIZE(24,80)    AUTOPAGE(YES)
BRACKET(YES)       BUILDCHAIN(NO)      ROUTEDMSGS(ALL)
SENDSIZE(256)      RECEIVESIZE(256)    IOAREALEN(512,0)
EXTENDEDDS(YES)    QUERY(ALL)          ATI(YES)  TTI(YES)
DISCREQ(YES)       RELREQ(YES)         AUTOCONNECT(YES)
LOGONMSG(NO)                           CREATESESS(NO)
```

This definition is for a 3287 printer. It is suitable for the following devices: 3262, 3268, 3287, 3289, and 5550.

DFH3270

Locally attached (non-SNA) 3270 displays:

```
TYPETERM(DFH3270)  GROUP(DFHTYPE)
DEVICE(3270)       TERMMODEL(2)
DEFSCREEN(24,80)   PAGESIZE(24,80)     AUTOPAGE(NO)
BRACKET(YES)       BUILDCHAIN(NO)      ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)  EXTENDEDDS(YES)     UCTRAN(YES)
SENDSIZE(0)        RECEIVESIZE(0)   IOAREALEN(512,0)
ERRLASTLINE(YES)   ERRINTENSIFY(YES)   ATI(YES)  TTI(YES)
DISCREQ(YES)       RELREQ(YES)         AUTOCONNECT(YES)
LOGONMSG(YES)      QUERY(ALL)          CREATESESS(NO)
```

This definition is for a 3278 Model 2 display. It is suitable for the following devices: 3178, 3179, 3277, 3278, 3279, and 3290.

DFH3270P

Locally attached (non-SNA) 3270 printers:

```
TYPETERM(DFH3270P) GROUP(DFHTYPE)
DEVICE(3270P)      TERMMODEL(2)
DEFSCREEN(24,80)   PAGESIZE(24,80)     AUTOPAGE(YES)
BRACKET(YES)       BUILDCHAIN(NO)      ROUTEDMSGS(ALL)
SENDSIZE(0)        RECEIVESIZE(0)      IOAREALEN(512,0)
EXTENDEDDS(YES)    QUERY(ALL)          ATI(YES)  TTI(YES)
DISCREQ(YES)       RELREQ(YES)         AUTOCONNECT(YES)
LOGONMSG(NO)                           CREATESESS(NO)
```

This definition is for a 3284 Model 2 printer. It is suitable for the following devices: 3262, 3268, 3284, 3287, 3288, 3289, and 5550.

DFHLU62T

SNA logical unit type 6.2 (APPC) single session terminal:

```
TYPETERM(DFHLU62T) GROUP(DFHTYPE)
DEVICE(APPC)
                   PAGESIZE(1,40)      AUTOPAGE(YES)
BRACKET(YES)       BUILDCHAIN(YES)     ROUTEDMSGS(NONE)
SENDSIZE(2048)     RECEIVESIZE(2048)   IOAREALEN(0,0)
                                       ATI(YES) TTI(YES)
LOGONMSG(NO)                           CREATESESS(NO)
```

This definition is for an APPC single session terminal and is also suitable for the following devices: DISPLAYWRITER, SCANMASTER, and SYSTEM/38.

DFHLU0E2

Non-SNA model 2 with extended data stream (Query):

```
TYPETERM(DFHLU0E2) GROUP(DFHTYPE)
DEVICE(3270)       TERMMODEL(2)        SHIPPABLE(YES)
DEFSCREEN(24,80)   PAGESIZE(24,80)     AUTOPAGE(NO)
ALTSCREEN(0,0)     ALTPAGE(0,0)
BRACKET(YES)       BUILDCHAIN(NO)      ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)  EXTENDEDDS(YES)     UCTRAN(YES)
RECEIVESIZE(0)     SENDSIZE(0)         IOAREALEN(512,0)
ERRASTLINE(YES)    ERRINTENSIFY(YES)   QUERY(ALL)
DISCREQ(YES)       RELREQ(YES)         AUTOCONNECT(NO)
LOGONMSG(YES)      CREATESESS(NO)
ATI(YES)           TTI(YES)
```

Non-SNA model 2 with extended data stream (Query). This definition matches the z/OS Communications Server-supplied LOGMODE NSX32702

DFHLU0M2

Non-SNA model 2:

```
TYPETERM(DFHLU0M2)  GROUP(DFHTYPE)
DEVICE(3270)        TERMMODEL(2)        SHIPPABLE(YES)
DEFSCREEN(24,80)    PAGESIZE(24,80)     AUTOPAGE(NO)
ALTSCREEN(0,0)      ALTPAGE(0,0)
```

```
BRACKET(YES)         BUILDCHAIN(NO)       ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)    EXTENDEDDS(NO)       UCTRAN(YES)
RECEIVESIZE(0)       SENDSIZE(0)          IOAREALEN(512,0)
ERRLASTLINE(YES)     ERRINTENSIFY(YES)    QUERY(NO)
DISCREQ(YES)         RELREQ(YES)          AUTOCONNECT(NO)
LOGONMSG(YES)        CREATESESS(NO)
ATI(YES)             TTI(YES)
```

Non-SNA model 2 with extended data stream (Query). This definition matches the z/OS Communications Server-supplied LOGMODE D4B32782

DFHLU0M3

Non-SNA model 3:

```
TYPETERM(DFHLU0M3)   GROUP(DFHTYPE)
DEVICE(3270)         TERMMODEL(2)         SHIPPABLE(YES)
DEFSCREEN(24,80)     PAGESIZE(24,80)      AUTOPAGE(NO)
ALTSCREEN(32,80)     ALTPAGE(32,80)
BRACKET(YES)         BUILDCHAIN(NO)       ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)    EXTENDEDDS(NO)       UCTRAN(YES)
RECEIVESIZE(0)       SENDSIZE(0)          IOAREALEN(512,0)
ERRLASTLINE(YES)     ERRINTENSIFY(YES)    QUERY(NO)
DISCREQ(YES)         RELREQ(YES)          AUTOCONNECT(NO)
LOGONMSG(YES)        CREATESESS(NO)
ATI(YES)             TTI(YES)
```

This definition matches the z/OS Communications Server-supplied LOGMODE D4B32783.

DFHLU0M4

Non-SNA model 4:

```
TYPETERM(DFHLU0M4)     GROUP(DFHTYPE)
DEVICE(3270)           TERMMODEL(2)           SHIPPABLE(YES)
DEFSCREEN(24,80)       PAGESIZE(24,80)        AUTOPAGE(NO)
ALTSCREEN(43,80)       ALTPAGE(43,80)
BRACKET(YES)           BUILDCHAIN(NO)         ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)      EXTENDEDDS(NO)         UCTRAN(YES)
RECEIVESIZE(0)         SENDSIZE(0)            IOAREALEN(512,0)
ERRLASTLINE(YES)       ERRINTENSIFY(YES)      QUERY(NO)
DISCREQ(YES)           RELREQ(YES)            AUTOCONNECT(NO)
LOGONMSG(YES)          CREATESESS(NO)
ATI(YES)               TTI(YES)
```

This definition matches the z/OS Communications Server-supplied LOGMODE D4B32784.

DFHLU0M5

Non-SNA model 5:

```
TYPETERM(DFHLU0M5)     GROUP(DFHTYPE)
DEVICE(3270)           TERMMODEL(2)           SHIPPABLE(YES)
DEFSCREEN(24,80)       PAGESIZE(24,80)        AUTOPAGE(NO)
ALTSCREEN(27,132)      ALTPAGE(27,132)
BRACKET(YES)           BUILDCHAIN(NO)         ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)      EXTENDEDDS(NO)         UCTRAN(YES)
RECEIVESIZE(0)         SENDSIZE(0)            IOAREALEN(512,0)
ERRLASTLINE(YES)       ERRINTENSIFY(YES)      QUERY(NO)
DISCREQ(YES)           RELREQ(YES)            AUTOCONNECT(NO)
LOGONMSG(YES)          CREATESESS(NO)
ATI(YES)               TTI(YES)
```

This definition matches the z/OS Communications Server-supplied LOGMODE D4B32785.

DFHLU2E2

SNA LU type 2 model 2 with extended data stream (Query):

```
TYPETERM(DFHLU2E2)     GROUP(DFHTYPE)
DEVICE(LUTYPE2)        TERMMODEL(2)           SHIPPABLE(YES)
DEFSCREEN(24,80)       PAGESIZE(24,80)        AUTOPAGE(NO)
ALTSCREEN(0,0)         ALTPAGE(0,0)
BRACKET(YES)           BUILDCHAIN(YES)        ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)      EXTENDEDDS(YES)        UCTRAN(YES)
RECEIVESIZE(1024)      SENDSIZE(3840)         IOAREALEN(256,4000)
ERRLASTLINE(YES)       ERRINTENSIFY(YES)      QUERY(ALL)
DISCREQ(YES)           RELREQ(YES)            AUTOCONNECT(NO)
```

```
LOGONMSG(YES)          CREATESESS(NO)
ATI(YES)               TTI(YES)
```

SNA LU type 2 model 2 with extended data stream (Query). This definition matches the z/OS Communications Server-supplied LOGMODE SNX32702.

DFHLU2M2

SNA LU type 2 model 2:

```
TYPETERM(DFHLU2M2)     GROUP(DFHTYPE)
DEVICE(LUTYPE2)        TERMMODEL(2)           SHIPPABLE(YES)
DEFSCREEN(24,80)       PAGESIZE(24,80)        AUTOPAGE(NO)
ALTSCREEN(0,0)         ALTPAGE(0,0)
BRACKET(YES)           BUILDCHAIN(YES)        ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)      EXTENDEDDS(NO)         UCTRAN(YES)
RECEIVESIZE(1024)      SENDSIZE(1536)         IOAREALEN(256,4000)
ERRLASTLINE(YES)       ERRINTENSIFY(YES)      QUERY(NO)
DISCREQ(YES)           RELREQ(YES)            AUTOCONNECT(NO)
LOGONMSG(YES)          CREATESESS(NO)
ATI(YES)               TTI(YES)
```

This definition matches the z/OS Communications Server-supplied LOGMODE D4A32782.

DFHLU2E3

SNA LU type 2 model 3 with extended data stream (Query):

```
TYPETERM(DFHLU2E3)     GROUP(DFHTYPE)
DEVICE(LUTYPE2)        TERMMODEL(2)           SHIPPABLE(YES)
DEFSCREEN(24,80)       PAGESIZE(24,80)        AUTOPAGE(NO)
ALTSCREEN(32,80)       ALTPAGE(32,80)
BRACKET(YES)           BUILDCHAIN(YES)        ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)      EXTENDEDDS(YES)        UCTRAN(YES)
RECEIVESIZE(1024)      SENDSIZE(3840)         IOAREALEN(256,4000)
ERRLASTLINE(YES)       ERRINTENSIFY(YES)      QUERY(ALL)
DISCREQ(YES)           RELREQ(YES)            AUTOCONNECT(NO)
LOGONMSG(YES)          CREATESESS(NO)
ATI(YES)               TTI(YES)
```

SNA LU type 2 model 3 with extended data stream (Query). This definition matches the z/OS Communications Server-supplied LOGMODE SNX32703.

DFHLU2M3

SNA LU type 2 model 3:

```
TYPETERM(DFHLU2M3)     GROUP(DFHTYPE)
DEVICE(LUTYPE2)        TERMMODEL(2)           SHIPPABLE(YES)
DEFSCREEN(24,80)       PAGESIZE(24,80)        AUTOPAGE(NO)
ALTSCREEN(32,80)       ALTPAGE(32,80)
BRACKET(YES)           BUILDCHAIN(YES)        ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)      EXTENDEDDS(NO)         UCTRAN(YES)
RECEIVESIZE(1024)      SENDSIZE(1536)         IOAREALEN(256,4000)
ERRLASTLINE(YES)       ERRINTENSIFY(YES)      QUERY(NO)
DISCREQ(YES)           RELREQ(YES)            AUTOCONNECT(NO)
LOGONMSG(YES)          CREATESESS(NO)
ATI(YES)               TTI(YES)
```

This definition matches the z/OS Communications Server-supplied LOGMODE D4A32783.

DFHLU2E4

SNA LU type 2 model 4 with extended data stream (Query):

```
TYPETERM(DFHLU2E4)     GROUP(DFHTYPE)
DEVICE(LUTYPE2)        TERMMODEL(2)           SHIPPABLE(YES)
DEFSCREEN(24,80)       PAGESIZE(24,80)        AUTOPAGE(NO)
ALTSCREEN(43,80)       ALTPAGE(43,80)
BRACKET(YES)           BUILDCHAIN(YES)        ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)      EXTENDEDDS(YES)        UCTRAN(YES)
RECEIVESIZE(1024)      SENDSIZE(3840)         IOAREALEN(256,4000)
ERRLASTLINE(YES)       ERRINTENSIFY(YES)      QUERY(ALL)
DISCREQ(YES)           RELREQ(YES)            AUTOCONNECT(NO)
LOGONMSG(YES)          CREATESESS(NO)
ATI(YES)               TTI(YES)
```

SNA LU type 2 model 4 with extended data stream (Query). This definition matches the z/OS Communications Server-supplied LOGMODE SNX32704.

DFHLU2M4
SNA LU type 2 model 4:

```
TYPETERM(DFHLU2M4)     GROUP(DFHTYPE)
DEVICE(LUTYPE2)        TERMMODEL(2)           SHIPPABLE(YES)
DEFSCREEN(24,80)       PAGESIZE(24,80)        AUTOPAGE(NO)
ALTSCREEN(43,80)       ALTPAGE(43,80)
BRACKET(YES)           BUILDCHAIN(YES)        ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)      EXTENDEDDS(NO)         UCTRAN(YES)
RECEIVESIZE(1024)      SENDSIZE(1536)         IOAREALEN(256,4000)
ERRLASTLINE(YES)       ERRINTENSIFY(YES)      QUERY(NO)
DISCREQ(YES)           RELREQ(YES)            AUTOCONNECT(NO)
LOGONMSG(YES)          CREATESESS(NO)
ATI(YES)               TTI(YES)
```

This definition matches the z/OS Communications Server-supplied LOGMODE D4A32784.

DFHLU2M5
SNA LU type 2 model 5:

```
TYPETERM(DFHLU2M5)     GROUP(DFHTYPE)
DEVICE(LUTYPE2)        TERMMODEL(2)           SHIPPABLE(YES)
DEFSCREEN(24,80)       PAGESIZE(24,80)        AUTOPAGE(NO)
ALTSCREEN(27,132)      ALTPAGE(27,132)
BRACKET(YES)           BUILDCHAIN(YES)        ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)      EXTENDEDDS(NO)         UCTRAN(YES)
RECEIVESIZE(1024)      SENDSIZE(1536)         IOAREALEN(256,4000)
ERRLASTLINE(YES)       ERRINTENSIFY(YES)      QUERY(NO)
DISCREQ(YES)           RELREQ(YES)            AUTOCONNECT(NO)
LOGONMSG(YES)          CREATESESS(NO)
ATI(YES)               TTI(YES)
```

This definition matches the z/OS Communications Server-supplied LOGMODE D4A32785.

DFHLU2E5
SNA LU type 2 model 5 with extended data stream (Query):

```
TYPETERM(DFHLU2E5)     GROUP(DFHTYPE)
DEVICE(LUTYPE2)        TERMMODEL(2)           SHIPPABLE(YES)
DEFSCREEN(24,80)       PAGESIZE(24,80)        AUTOPAGE(NO)
ALTSCREEN(27,132)      ALTPAGE(27,132)
BRACKET(YES)           BUILDCHAIN(YES)        ROUTEDMSGS(ALL)
AUDIBLEALARM(YES)      EXTENDEDDS(NO)         UCTRAN(YES)
RECEIVESIZE(1024)      SENDSIZE(3480)         IOAREALEN(256,4000)
ERRLASTLINE(YES)       ERRINTENSIFY(YES)      QUERY(ALL)
DISCREQ(YES)           RELREQ(YES)            AUTOCONNECT(NO)
LOGONMSG(YES)          CREATESESS(NO)
ATI(YES)               TTI(YES)
```

SNA LU type 2 model 5 with extended data stream (Query). This definition matches the z/OS Communications Server-supplied LOGMODE LSX32705.

Model TERMINAL definitions in group DFHTERM

The CICS-supplied CSD group DFHTERM contains model TERMINAL definitions for automatic installation.

LU2
SNA 3270 Model 2 display using TYPETERM DFHLU2:

```
TERMINAL(LU2)          GROUP(DFHTERM)         TYPETERM(DFHLU2)
AUTINSTMODEL(ONLY)     AUTINSTNAME(DFHLU2)
```

This definition is for a 3278 Model 2 display. It is suitable for the following devices: 3178, 3179, 3277, 3278, 3279, 3290, 3270PC, 3270PC/G, 3270PC/GX, 8775, and 5550.

LU3
SNA 3270 Model 2 printer using TYPETERM DFHLU3:

```
TERMINAL(LU3)          GROUP(DFHTERM)         TYPETERM(DFHLU3)
AUTINSTMODEL(ONLY)     AUTINSTNAME(DFHLU3)
```

This definition is for a 3287 printer. It is suitable for the following devices: 3262, 3268, 3284, 3286, 3287, 3288, 3289, and 5550.

SCSP
SNA 3278 Model 2 printer using TYPETERM DFHSCSP:

```
TERMINAL(SCSP)      GROUP(DFHTERM)      TYPETERM(DFHSCSP)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHSCSP)
```

This definition is for a 3287 printer. It is suitable for the following devices: 3262, 3268, 3287, 3289, and 5550.

3270
Non-SNA 3270 Model 2 display using TYPETERM DFH3270:

```
TERMINAL(3270)      GROUP(DFHTERM)      TYPETERM(DFH3270)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFH3270)
```

This definition is for a 3278 Model 2 display. It is suitable for the following devices: 3178, 3179, 3277, 3278, 3279, and 3290.

3284
Non-SNA 3270 Model 2 printer using TYPETERM DFH3270P:

```
TERMINAL(3284)      GROUP(DFHTERM)      TYPETERM(DFH3270P)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFH3270P)
```

This definition is for a 3284 Model 2 printer. It is suitable for the following devices: 3262, 3268, 3284, 3287, 3288, 3289, and 5550.

LU62
APPC (LU6.2) single session terminal using TYPETERM DFHLU62T:

```
TERMINAL(LU62)      GROUP(DFHTERM)       TYPETERM(DFHLU62T)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU62T)
```

This definition is for an APPC single session terminal and is also suitable for the following devices: DISPLAYWRITER, SCANMASTER, and SYSTEM/38.

L0E2
Non-SNA Model 2 display using TYPETERM DFHLU0E2:

```
TERMINAL(L0E2)      GROUP(DFHTERM)       TYPETERM(DFHLU0E2)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU0E2)
```

Non-SNA model 2 with extended data stream (Query). This definition matches the z/OS Communications Server-supplied LOGMODE NSX32702.

L0M2
Non-SNA Model 2 display using TYPETERM DFHLU0M2:

```
TERMINAL(L0M2)      GROUP(DFHTERM)       TYPETERM(DFHLU0M2)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU0M2)
```

This definition matches the z/OS Communications Server-supplied LOGMODE D4B32782.

L0M3
Non-SNA Model 3 display using TYPETERM DFHLU0M3:

```
TERMINAL(L0M3)      GROUP(DFHTERM)       TYPETERM(DFHLU0M3)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU0M3)
```

This definition matches the z/OS Communications Server-supplied LOGMODE D4B32783.

L0M4
Non-SNA Model 4 display using TYPETERM DFHLU0M4:

```
TERMINAL(L0M4)      GROUP(DFHTERM)        TYPETERM(DFHLU0M4)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU0M4)
```

This definition matches the z/OS Communications Server-supplied LOGMODE D4B32784.

L0M5
Non-SNA Model 5 display using TYPETERM DFHLU0M5:

```
TERMINAL(L0M5)      GROUP(DFHTERM)        TYPETERM(DFHLU0M5)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU0M5)
```

This definition matches the z/OS Communications Server-supplied LOGMODE D4B32785.

L2E2
SNA LU2 Model 2 display using TYPETERM DFHLU2E2:

```
TERMINAL(L2E2)      GROUP(DFHTERM)        TYPETERM(DFHLU2E2)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU2E2)
```

SNA LU type 2 model 2 with extended data stream (Query). This definition matches the z/OS Communications Server-supplied LOGMODE SNX32702.

L2M2
SNA LU2 Model 2 display using TYPETERM DFHLU2M2:

```
TERMINAL(L2M2)      GROUP(DFHTERM)        TYPETERM(DFHLU2M2)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU2M2)
```

This definition matches the z/OS Communications Server-supplied LOGMODE D4A32782.

L2E3
SNA LU2 Model 3 display using TYPETERM DFHLU2E3:

```
TERMINAL(L2E3)      GROUP(DFHTERM)        TYPETERM(DFHLU2E3)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU2E3)
```

SNA LU type 2 model 3 with extended data stream (Query). This definition matches the z/OS Communications Server-supplied LOGMODE SNX32703.

L2M3
SNA LU2 Model 3 display using TYPETERM DFHLU2M3:

```
TERMINAL(L2M3)      GROUP(DFHTERM)        TYPETERM(DFHLU2M3)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU2M3)
```

This definition matches the z/OS Communications Server-supplied LOGMODE D4A32783.

L2E4
SNA LU2 Model 4 display using TYPETERM DFHLU2E4:

```
TERMINAL(L2E4)      GROUP(DFHTERM)        TYPETERM(DFHLU2E4)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU2E4)
```

SNA LU type 2 model 4 with extended data stream (Query). This definition matches the z/OS Communications Server-supplied LOGMODE SNX32704.

L2M4
SNA LU2 Model 4 display using TYPETERM DFHLU2M4:

```
TERMINAL(L2M4)      GROUP(DFHTERM)        TYPETERM(DFHLU2M4)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU2M4)
```

This definition matches the z/OS Communications Server-supplied LOGMODE D4A32784.

L2E5
SNA LU2 Model 5 display using TYPETERM DFHLU2E5:

```
TERMINAL(L2E5)      GROUP(DFHTERM)        TYPETERM(DFHLU2E5)
AUTINSTMODEL(ONLY)  AUTINSTNAME(DFHLU2E5)
```

SNA LU type 2 model 5 with extended data stream (Query). This definition matches the z/OS Communications Server-supplied LOGMODE LSX32705.

L2M5
SNA LU2 Model 5 display using TYPETERM DFHLU2M5:

```
TERMINAL(L2M5)       GROUP(DFHTERM)         TYPETERM(DFHLU2M5)
AUTINSTMODEL(ONLY)   AUTINSTNAME(DFHLU2M5)
```

This definition matches the z/OS Communications Server-supplied LOGMODE D4A32785.

CBRF
Default template terminal for use with the 3270 bridge

```
TERMINAL(CBRF)       GROUP(DFHTERM)         TYPETERM(DFHLU2)
NETNAME(CBRF)        REMOTESYSTEM(CBR)      REMOTENAME(CBRF)
```

PROFILE definitions in group DFHEP

The CICS-supplied CSD group DFHEP contains PROFILE definitions for event processing.

DFHECEPH
CICS uses this profile for the CEPH transaction (HTTP EP adapter) to manage timeouts set by the RTIMOUT parameter.

Here is the definition:

```
PROFILE(DFHECEPH)
GROUP(DFHEP)
SCRNSIZE(DEFAULT)
MSGJRNL(NO)
MSGINTEG(NO)
ONEWTE(NO)
DVSUPRT(ALL)
INBFMH(NO)
RAQ(NO)
LOGREC(NO)
NEPCLASS(000)
RTIMOUT(0005)
```

PROFILE definitions in group DFHISC

The supplied CSD group DFHISC contains PROFILE definitions for intersystem communication sessions.

DFHCICSF
CICS uses this profile for the session to the remote system or region when a CICS application program issues a function shipping request.

The definition is:

```
PROFILE(DFHCICSF)
GROUP(DFHISC)
SCRNSIZE(DEFAULT)
MSGJRNL(NO)
MSGINTEG(NO)
ONEWTE(NO)
DVSUPRT(ALL)
INBFMH(ALL)
RAQ(NO)
LOGREC(NO)
NEPCLASS(000)
```

DFHCICSR
CICS uses this profile in transaction routing for communication between the user transaction (running in the application-owning region) and the interregion link or APPC link.

The definition is:

```
PROFILE(DFHCICSR)
GROUP(DFHISC)
SCRNSIZE(DEFAULT)
MSGJRNL(NO)
```

```
MSGINTEG(NO)
ONEWTE(NO)
DVSUPRT(ALL)
INBFMH(ALL)
RAQ(NO)
LOGREC(NO)
NEPCLASS(000)
```

DFHCICSS
CICS uses this profile in transaction routing for communication between the relay transaction (running in the terminal-owning region) and the interregion link or APPC link. You can specify a different profile by means of the TRPROF option on the TRANSACTION definition.

The definition is:

```
PROFILE(DFHCICSS)
GROUP(DFHISC)
SCRNSIZE(DEFAULT)
MSGJRNL(NO)
MSGINTEG(NO)
ONEWTE(NO)
DVSUPRT(ALL)
INBFMH(ALL)
RAQ(NO)
LOGREC(NO)
NEPCLASS(000)
```

PROFILE definitions in group DFHSTAND

The CICS-supplied CSD group DFHSTAND contains PROFILE definitions.

The CICS-supplied CSD group DFHSTAND contains the following PROFILE definitions:

DFHCICSA
This is the default profile for alternate facilities acquired by the application program ALLOCATE command. A different profile can be named explicitly on the ALLOCATE command.

The definition is:

```
PROFILE(DFHCICSA)
GROUP(DFHSTAND)
SCRNSIZE(DEFAULT)
MSGJRNL(NO)
MSGINTEG(NO)
ONEWTE(NO)
PROTECT(NO)
DVSUPRT(ALL)
INBFMH(ALL)
RAQ(NO)
LOGREC(NO)
NEPCLASS(000)
```

DFHCICSE
This is the error profile for principal facilities. CICS uses this profile to pass an error message to the principal facility when the required profile cannot be found.

The definition is:

```
PROFILE(DFHCICSE)
GROUP(DFHSTAND)
SCRNSIZE(DEFAULT)
MSGJRNL(NO)
MSGINTEG(NO)
ONEWTE(NO)
PROTECT(NO)
DVSUPRT(ALL)
INBFMH(NO)
RAQ(NO)
LOGREC(NO)
NEPCLASS(000)
```

DFHCICSP
This is the default profile for the page retrieval transaction CSPG. You can specify a different profile for a particular transaction by means of the PROFILE option on the TRANSACTION definition.

The definition is:

```
PROFILE(DFHCICSP)
GROUP(DFHSTAND)
SCRNSIZE(DEFAULT)
UCTRAN(YES)
MSGJRNL(NO)
MSGINTEG(NO)
ONEWTE(NO)
PROTECT(NO)
DVSUPRT(ALL)
INBFMH(NO)
RAQ(NO)
LOGREC(NO)
NEPCLASS(000)
```

DFHCICST
This is the default profile for principal facilities. You can specify a different profile for a particular transaction by means of the PROFILE option on the TRANSACTION definition.

The definition is:

```
PROFILE(DFHCICST)
GROUP(DFHSTAND)
SCRNSIZE(DEFAULT)
MSGJRNL(NO)
MSGINTEG(NO)
ONEWTE(NO)
PROTECT(NO)
DVSUPRT(ALL)
INBFMH(NO)
RAQ(NO)
LOGREC(NO)
NEPCLASS(000)
```

DFHCICSV
This is the profile for principal facilities, when the transaction supports only z/OS Communications Server devices.

The definition is:

```
PROFILE(DFHCICSV)
GROUP(DFHSTAND)
SCRNSIZE(DEFAULT)
MSGJRNL(NO)
MSGINTEG(NO)
ONEWTE(NO)
PROTECT(NO)
DVSUPRT(VTAM)
INBFMH(NO)
RAQ(NO)
LOGREC(NO)
NEPCLASS(000)
```

Note: VTAM is now z/OS Communications Server.

DFHPPF01
Profile DFHPPF01 is used during CICS initialization, for tasks that are attached before the CSD file definitions have been installed.

The definitions are:

```
PROFILE(DFHPPF01)
GROUP(DFHSTAND)
DESCRIPTION(VTAM-ONLY PROFILE)
SCRNSIZE(DEFAULT)
MSGJRNL(NO)
MSGINTEG(NO)
ONEWTE(NO)
PROTECT(NO)
```

```
DVSUPRT(VTAM)
INBFMH(NO)
RAQ(NO)
LOGREC(NO)
NEPCLASS(000)
```

DFHPPF02

Profile DFHPPF02 is used during CICS initialization, for tasks that are attached before the CSD file definitions have been installed.

The definitions are:

```
PROFILE(DFHPPF02)
GROUP(DFHSTAND)
DESCRIPTION(ALL-NULLS PROFILE)
SCRNSIZE(DEFAULT)
MSGJRNL(NO)
MSGINTEG(NO)
ONEWTE(NO)
PROTECT(NO)
DVSUPRT(ALL)
INBFMH(NO)
RAQ(NO)
LOGREC(NO)
NEPCLASS(000)
```

Model definitions in group DFHPGAIP

CICS supplies a number of model definitions in group DFHPGAIP to support program autoinstall.

DFHPGAPG

This is the default PROGRAM definition for program autoinstall.

The definition is:

```
PROGRAM(DFHPGAPG)  GROUP(DFHPGAIP)
DESCRIPTION(default program for program autoinstall)
                   RELOAD(NO)            RESIDENT(NO)
USAGE(NORMAL)      USELPACOPY(NO)        STATUS(ENABLED)
RSL(00)            CEDF(YES)             DATALOCATION(ANY)
EXECKEY(USER)      EXECUTIONSET(FULLAPI)
```

If the specification of `DATALOCATION(ANY)` is an unsuitable default, you can prevent `DATALOCATION` from defaulting to `ANY` in one of the following ways:

- Copy the definition for DFHPGAPG to your own group and alter the DATALOCATION setting. Ensure that the definition is installed after group DFHPGAIP.
- Specify the name of your own program definition as a model definition to be used as an output parameter from the program autoinstall exit.

DFHPGAMP

This is the default MAPSET definition for program autoinstall.

The definition is:

```
MAPSET(DFHPGAMP)  GROUP(DFHPGAIP)
DESCRIPTION(default mapset for program autoinstall)
RESIDENT(NO)       USAGE(NORMAL)
USELPACOPY(NO)     STATUS(ENABLED)
RSL(00)
```

DFHPGAPT

This is the default PARTITION definition for program autoinstall.

The definition is:

```
PARTITIONSET(DFHPGAPT)  GROUP(DFHPGAIP)
DESCRIPTION(default partitionset for program autoinstall)
RESIDENT(NO)       USAGE(NORMAL)
USELPACOPY(NO)     STATUS(ENABLED)
RSL(00)
```

Notices

This information was developed for products and services offered in the United States of America. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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

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

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

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

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

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

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119 Armonk,
NY 10504-1785
United States of America

© Copyright IBM Corp. 1974, 2023

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 Client Relationship Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming interface information

IBM CICS supplies some documentation that can be considered to be Programming Interfaces, and some documentation that cannot be considered to be a Programming Interface.

Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 4 (CICS TS 5.4) are included in the following sections of the online product documentation:

- Developing applications
- Developing system programs
- Securing overview
- Developing for external interfaces
- Application development reference
- Reference: system programming
- Reference: connectivity

Information that is NOT intended to be used as a Programming Interface of CICS TS 5.4, but that might be misconstrued as Programming Interfaces, is included in the following sections of the online product documentation:

- Troubleshooting and support
- Reference: diagnostics

If you access the CICS documentation in manuals in PDF format, Programming Interfaces that allow the customer to write programs to obtain the services of CICS TS 5.4 are included in the following manuals:

- Application Programming Guide and Application Programming Reference
- Business Transaction Services

- Customization Guide
- C++ OO Class Libraries
- Debugging Tools Interfaces Reference
- Distributed Transaction Programming Guide
- External Interfaces Guide
- Front End Programming Interface Guide
- IMS Database Control Guide
- Installation Guide
- Security Guide
- CICS Transactions
- CICSPlex System Manager (CICSPlex SM) Managing Workloads
- CICSPlex SM Managing Resource Usage
- CICSPlex SM Application Programming Guide and Application Programming Reference
- Java Applications in CICS

If you access the CICS documentation in manuals in PDF format, information that is NOT intended to be used as a Programming Interface of CICS TS 5.4, but that might be misconstrued as Programming Interfaces, is included in the following manuals:

- Data Areas
- Diagnosis Reference
- Problem Determination Guide
- CICSPlex SM Problem Determination Guide

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Apache, Apache Axis2, Apache Maven, Apache Ivy, the Apache Software Foundation (ASF) logo, and the ASF feather logo are trademarks of Apache Software Foundation.

Gradle and the Gradlephant logo are registered trademark of Gradle, Inc. and its subsidiaries in the United States and/or other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Red Hat®, and Hibernate® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Spring Boot is a trademark of Pivotal Software, Inc. in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Zowe™, the Zowe logo and the Open Mainframe Project™ are trademarks of The Linux Foundation.

The Stack Exchange name and logos are trademarks of Stack Exchange Inc.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability
These terms and conditions are in addition to any terms of use for the IBM website.

Personal use
You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use
You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights
Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM online privacy statement

IBM Software products, including software as a service solutions, (*Software Offerings*) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information (PII) is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect PII. If this Software Offering uses cookies to collect PII, specific information about this offering's use of cookies is set forth below:

For the CICSPlex SM Web User Interface (main interface):
Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's user name and other PII for purposes of session management, authentication, enhanced user usability, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface (data interface):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's user name and other PII for purposes of session management, authentication, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface ("hello world" page):
Depending upon the configurations deployed, this Software Offering may use session cookies that do not collect PII. These cookies cannot be disabled.

For CICS Explorer:

Depending upon the configurations deployed, this Software Offering may use session and persistent preferences that collect each user’s user name and password, for purposes of session management, authentication, and single sign-on configuration. These preferences cannot be disabled, although storing a user's password on disk in encrypted form can only be enabled by the user's explicit action to check a check box during sign-on.

If the configurations deployed for this Software Offering provide you, as customer, the ability to collect PII from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM Privacy Policy and IBM Online Privacy Statement, the section entitled *Cookies, Web Beacons and Other Technologies* and the IBM Software Products and Software-as-a-Service Privacy Statement.

Index

Numerics

3290 terminal 228
3600 option
  LDC operand 403
3770 Data Communication System
  LDCs for batch LU
    TCT example 405

A

ABCODE operand
  DFHSRT TYPE=SYSTEM 393
  DFHSRT TYPE=USER 393
abend codes
  DFHSRT TYPE=SYSTEM 393
  DFHSRT TYPE=USER 393
ACCESSMETHOD attribute
  CONNECTION definition 10
ACCMETH operand
  DFHFCT TYPE=FILE 359
  DFHTCT TYPE=INITIAL 396
  DFHTCT TYPE=LINE 407
  DFHTCT TYPE=REMOTE 412
  DFHTCT TYPE=TERMINAL 411
  remote terminals 411, 412
  sequential devices 407
ACCOUNTREC attribute
  DB2CONN definition 26
  DB2ENTRY definition 33
ACTION attribute
  TRANSACTION definition 197
ADD attribute
  FILE definition 49
ADD command
  CEDA 275
ADD command, DFHCSDUP utility program 323
ADD option
  SERVREQ operand 363
ADDCNT option
  PERFORM operand 371
ALIAS attribute
  TRANSACTION definition 197
ALTER command
  CEDA 277
ALTER command, DFHCSDUP utility program
  generic naming in 326
ALTPAGE attribute
  TYPETERM definition 232
ALTPRINTCOPY attribute
  TERMINAL definition 182
ALTPRINTER attribute
  TERMINAL definition 182
ALTSCREEN attribute
  TYPETERM definition 233
ALTSUFFIX attribute
  TYPETERM definition 233
ANALYZER attribute
  URIMAP definition 258
APF (authorized program facility) 353, 390
API attribute
  PROGRAM definition 121
APLKYBD attribute
  TYPETERM definition 234
APLTEXT attribute
  TYPETERM definition 234
APPC (LUTYPE6.2) links and parallel sessions 131
APPC (LUTYPE6.2) single session terminal 176
APPC devices for transaction routing 180
APPEND command
  CEDA 280
APPEND command, DFHCSDUP utility program
  examples 327
APPENDCRLF attribute
  DOCTEMPLATE definition 40
APPLID attribute
  IPCONN definition 66
APPLID of remote system 14
APPLID operand
  DFHSIT macro 8
ASCII attribute
  TYPETERM definition 234
AT-TLS 151
ATI attribute
  TYPETERM definition 235
ATIFACILITY attribute
  TDQUEUE definition 157
ATOMDEF object 4
ATOMSERVICE attribute
  ATOMSERVICE definition 1
  URIMAP definition 258
ATOMSERVICE definition
  ATOMSERVICE attribute 1
  ATOMTYPE attribute 1
  BINDFILE attribute 2
  CONFIGFILE attribute 2
  RESOURCENAME attribute 3
  RESOURCETYPE attribute 3
  STATUS attribute 3
Atomservice definitions view 4
ATOMSERVICE resource definition 1
ATOMTYPE attribute
  ATOMSERVICE definition 1
ATTACHSEC attribute
  CONNECTION definition 11
  TCPIPSERVICE definition 142
  TERMINAL definition 182
ATTLSAWARE 151
attributes
  CONNECTION definition 9
  DB2CONN definition 20
  DB2ENTRY definition 32
  DB2TRAN definition 37
  DOCTEMPLATE definition 40

attributes *(continued)*
ENQMODEL definition 44
FILE definition 49
JOURNALMODEL definition 75
LIBRARY definition 84
LSRPOOL definition 88
MAPSET definition 94
MQCONN definition 96
PARTITIONSET definition 103
PARTNER definition 105
PROCESSTYPE definition 112
PROFILE definition 114
PROGRAM definition 121
SESSIONS definition 132
TCPIPSERVICE definition 142
TDQUEUE definition 155
TERMINAL definition 181
TRANCLASS definition 192
TRANSACTION definition 196
TSMODEL definition 212
TYPETERM definition 230
AUDIBLEALARM attribute
TYPETERM definition 235
AUDITLEVEL attribute
PROCESSTYPE definition 112
AUDITLOG attribute
PROCESSTYPE definition 113
AUTHENTICATE attribute
TCPIPSERVICE definition 142
AUTHID attribute
DB2CONN definition 26
DB2ENTRY definition 34
authorized program facility (APF) 353, 390
AUTHTYPE attribute
DB2CONN definition 26
DB2ENTRY definition 34
AUTINSTMODEL attribute
TERMINAL definition 183
AUTINSTMODEL TERMINAL definition
CICS-supplied 445
AUTINSTNAME attribute
TERMINAL definition 183
AUTOCONNECT attribute
CONNECTION definition 12
IPCONN definition 66
SESSIONS definition 133
TYPETERM definition 235
AUTOPAGE attribute
TYPETERM definition 236

B

BACKLOG attribute
TCPIPSERVICE definition 142
BACKTRANS attribute
TYPETERM definition 236
BACKUPTYPE attribute
FILE definition 49, 50
basic mapping support (BMS)
multiple page messages to printers 404
BCHLU option
LDC operand 403
BINDFILE attribute
ATOMSERVICE definition 2
BINDPASSWORD attribute
CONNECTION 427
BINDSECURITY attribute
CONNECTION attribute 12
TERMINAL definition 183
BLKKEYL operand
DFHFCT TYPE=FILE 359
BLKSIZE operand
DFHFCT TYPE=FILE 359
DFHTCT TYPE=SDSCI 406
sequential devices 406
BLOCKED option
RECFORM operand 362
BLOCKFORMAT attribute
TDQUEUE definition 158
BLOCKSIZE attribute
TDQUEUE definition 158
BMS (basic mapping support)
multiple page messages to printers 404
BRACKET attribute
TYPETERM definition 236
BREXIT attribute
TRANSACTION definition 198
BROWSE attribute
FILE definition 49
BROWSE option
SERVREQ operand 363
BUILDCHAIN attribute
SESSIONS definition 134
TYPETERM definition 236
BUNDLE resources 4

C

CADL transient data destination 167
CADO transient data destination 167
CADS transient data destination 167
CAIL transient data destination 167
CCPI transient data destination 167
CCSE transient data destination 167
CCSI transient data destination 167
CCSO transient data destination 167
CCZM transient data destination 167
CDB2 transient data destination 167
CDBC transient data destination 167
CDEP transient data destination 168
CDUL transient data destination 167
CECO transient data destination 168
CEDA
ADD command 275
ALTER command 277
APPEND command 280
CHECK command 281
COPY command 283
DEFINE command 287
DELETE command 290
DISPLAY command 293
DISPLAY GROUP command 295
DISPLAY LIST command 295
EXPAND command 297
EXPAND GROUP command 299
EXPAND LIST command 300
INSTALL command 301
LOCK command 304

CEDA *(continued)*
MOVE command 306
REMOVE command 309
RENAME command 311
UNLOCK command 313
USERDEFINE command 315
VIEW command 319
CEDF attribute
PROGRAM definition 121
CEJL transient data destination 168
CEPO transient data destination 168
CERTIFICATE attribute
IPCONN definition 66
TCPIPSERVICE definition 142
URIMAP definition 259
CESE transient data destination 168
CFDTPOOL attribute
FILE definition 49
CGCSGID attribute
TYPETERM definition 236
CHAINCONTROL attribute
PROFILE definition 115
CHARACTERSET attribute
URIMAP definition 259
CHECK command
CEDA 281
checking definitions of DB2 conenction resources 31
checking definitions of DB2 entry resources 37
checking definitions of DB2 transaction resources 39
checking terminal definitions 191
CICS-supplied TYPETERM definitions
ALTPAGE attribute 232
CICS-supplied 441
CIEO transient data destination 168
CIPHERS attribute
IPCONN definition 66
TCPIPSERVICE definition 142
URIMAP definition 259
CISL transient data destination 168
CISO transient data destination 168
CJRM transient data destination 168
CKQQ transient data destination 168
CLASS operand
DFHMCT TYPE=EMP 369
DFHMCT TYPE=RECORD 372
CLOCK operand
DFHMCT TYPE=EMP 370
CLT (command list table)
CLT (command list table) 353
DFHCLT macros 353
DFHCLT TYPE=COMMAND 355
DFHCLT TYPE=INITIAL 353
DFHCLT TYPE=LISTEND 356
DFHCLT TYPE=LISTSTART 354
DFHCLT TYPE=WTO 355
CMDSEC attribute
TRANSACTION definition 198
CMLO transient data destination 168
CMPO transient data destination 168
CMQM transient data destination 168
coding examples
FCT 365
MCT 386
PLT 389
coding examples *(continued)*
RST 392
SRT 394
TLT 416
XLT 425
COLOR attribute
TYPETERM definition 237
COMAUTHID attribute
DB2CONN definition 29
COMAUTHTYPE attribute
DB2CONN definition 29
command list table (CLT)
CLT (command list table) 353
DFHCLT macros 353
DFHCLT TYPE=COMMAND 355
DFHCLT TYPE=INITIAL 353
DFHCLT TYPE=LISTEND 356
DFHCLT TYPE=LISTSTART 354
DFHCLT TYPE=WTO 355
COMMAND operand
DFHCLT TYPE=COMMAND 355
command thread attributes
DB2CONN definition 29
COMPLETE option
MIGRATE operand 397
COMTHREADLIMIT attribute
DB2CONN definition 30
CONCURRENCY attribute
PROGRAM definition 121
CONFDATA attribute
TRANSACTION definition 198
CONFIGFILE attribute
ATOMSERVICE definition 2
configurators
DFHFCT TYPE=FILE 364
CONNDEF object 20
CONNECTERROR attribute
DB2CONN definition 22
CONNECTION attribute
CONNECTION definition 12
SESSIONS definition 134
CONNECTION definition
ACCESSMETHOD attribute 8, 10
ATTACHSEC attribute 11
AUTOCONNECT attribute 12
BINDPASSWORD attribute 427
BINDSECURITY attribute 12
CONNECTION attribute 12
CONNTYPE attribute 12
DATASTREAM attribute 13
INDSYS attribute 8, 13
INSERVICE attribute 13
MAXQTIME attribute 14
NETNAME attribute 8, 14
PROTOCOL attribute 8, 16
PSRECOVERY attribute 16
QUEUELIMIT attribute 16
RECORDFORMAT attribute 16
REMOTENAME attribute 17
REMOTESYSNET attribute 17
REMOTESYSTEM attribute 17
SECURITYNAME attribute 18
SINGLESESS attribute 18
USEDFLTUSER attribute 18

CONNECTION definition *(continued)*
XLNACTION attribute 18
CONNECTION definitions
installing 19
CONNECTION resources 6
connections, INDIRECT
resource definitions 8
CONNTYPE attribute
CONNECTION definition 12
CONSNAME attribute
TERMINAL definition 183
CONSOLE attribute
TERMINAL definition 427
consoles
preset security 190
continuous printing (autopaging) 404
CONVERTER attribute
URIMAP definition 260
COPY attribute
TYPETERM definition 237
COPY command
CEDA 283
COPY command, DFHCSDUP utility program
examples 331
FROMCSD option 330
generic naming in 330
MERGE option 330
REPLACE option 330
COUNT operand
DFHMCT TYPE=EMP 370
coupling facility data tables 47
CPIO transient data destination 168
CRDI transient data destination 168
CREATESESS attribute
TYPETERM definition 237
CRITICAL attribute
LIBRARY definition 84
CRLO transient data destination 168
CRPO transient data destination 169
CSBA transient data destination 169
CSBR transient data destination 169
CSCC transient data destination 169
CSCS transient data destination 169
CSD
Sharing between releases 437
CSD (CICS system definition) file
contents after initialization 429
groups of resource definitions
DFHISC 429
DFHSTAND 429
DFHTERM 429
DFHTYPE 429
CSDH transient data destination 169
CSDL transient data destination 169
CSFL transient data destination 169
CSFU transaction 360
CSJE transient data destination 169
CSJO transient data destination 169
CSKL transient data destination 169
CSLB transient data destination 169
CSML transient data destination 170
CSNE transient data destination 170
CSOO transient data destination 170
CSPL transient data destination 170
CSQL transient data destination 170
CSRL transient data destination 170
CSSH transient data destination 170
CSSL transient data destination 170
CSTL transient data destination 170
CSZL transient data destination 170
CSZX transient data destination 170
CTLAPPLS operand
DFHRST TYPE=RSE 391
CWBO transient data destination 170
CWBW transient data destination 170

D

data sets
relationship with files 357
data sharing
DFHTST TYPE=SHARED 422
temporary storage 422
DATA12K attribute
LSRPOOL definition 90
DATA16K attribute
LSRPOOL definition 90
DATA1K attribute
LSRPOOL definition 90
DATA20K attribute
LSRPOOL definition 90
DATA24K attribute
LSRPOOL definition 90
DATA28K attribute
LSRPOOL definition 90
DATA2K attribute
LSRPOOL definition 90
DATA32K attribute
LSRPOOL definition 91
DATA4K attribute
LSRPOOL definition 90
DATA512 attribute
LSRPOOL definition 90
DATA8K attribute
LSRPOOL definition 90
database control (DBCTL) 390
DATABUFFERS attribute
FILE definition 49
TDQUEUE definition 158
DATAID operand
DFHTST TYPE=LOCAL 420
DFHTST TYPE=RECOVERY 418
DFHTST TYPE=REMOTE 420
DFHTST TYPE=SECURITY 422
DATALOCATION attribute
PROGRAM definition 121
DATASTREAM attribute
CONNECTION definition 13
DB2 connection definitions view 31
DB2 entry resources 31
DB2 transaction resources 37
DB2CDEF object 31
DB2CONN
installing and discarding 30
DB2CONN attribute
DB2CONN definition 22
DB2CONN definition
ACCOUNTREC attribute 26

DB2CONN definition *(continued)*
AUTHID attribute 26
AUTHTYPE attribute 26
COMAUTHID attribute 29
COMAUTHTYPE attribute 29
command thread attributes
COMAUTHID 29
COMAUTHTYPE 29
COMTHREADLIMIT 30
COMTHREADLIMIT attribute 30
CONNECTERROR attribute 22
connection attributes
CONNECTERROR 22
DB2GROUPID 22
DB2ID 22
MSGQUEUE1 22
MSGQUEUE2 22
MSGQUEUE3 22
NONTERMREL 22
PURGECYCLE 22
RESYNCMEMBER 22
SIGNID 22
STANDBYMODE 22
STATSQUEUE 22
TCBLIMIT 22
THREADERROR 22
DB2CONN attribute 22
DB2GROUPID attribute 22
DB2ID attribute 22
DROLLBACK attribute 26
MSGQUEUE1 attribute 22
MSGQUEUE2 attribute 22
MSGQUEUE3 attribute 22
NONTERMREL attribute 22
PLAN attribute 26
PLANEXITNAME attribute 26
pool thread attributes
ACCOUNTREC 26
AUTHID 26
AUTHTYPE 26
DROLLBACK 26
PLAN 26
PLANEXITNAME 26
THREADLIMIT 26
THREADWAIT 26
PURGECYCLE attribute 22
RESYNCMEMBER attribute 22
SIGNID attribute 22
STANDBYMODE attribute 22
STATSQUEUE attribute 22
TCBLIMIT attribute 22
THREADERROR attribute 22
THREADLIMIT attribute 26
THREADWAIT attribute 26
DB2CONN resources 20
DB2EDEF object 37
DB2ENTRY attribute
DB2ENTRY definition 32
DB2ENTRY definition
ACCOUNTREC attribute 33
AUTHID attribute 34
AUTHTYPE attribute 34
DB2ENTRY attribute 32
DROLLBACK attribute 35
DB2ENTRY definition *(continued)*
PLAN attribute 35
PLANEXITNAME attribute 35
PRIORITY attribute 35
PROTECTNUM attribute 36
thread operation attributes
ACCOUNTREC 33
AUTHID 34
AUTHTYPE 34
DROLLBACK 35
PLAN 35
PLANEXITNAME 35
PRIORITY 35
PROTECTNUM 36
THREADLIMIT 36
THREADWAIT 36
thread selection attributes
TRANSID 33
THREADLIMIT attribute 36
THREADWAIT attribute 36
TRANSID attribute 33
DB2GROUPID attribute
DB2CONN definition 22
DB2ID attribute
DB2CONN definition 22
DB2TDEF object 39
DB2TRAN attribute
DB2TRAN definition 37
DB2TRAN definition
DB2TRAN attribute 37
ENTRY attribute 38
RDO command 37
TRANSID attribute 38
DBCTL (database control) 390
DDName attribute
DOCTEMPLATE definition 40
DDNAME attribute
TDQUEUE definition 159
DDNAME operand
DFHTCT TYPE=SDSCI 406
sequential devices 406
Default values
TYPETERM attributes 217
define
ENQMODEL 45
file 64
LIBRARY 87
DEFINE command
CEDA 287
DEFINE command, DFHCSDUP utility program
examples 333
defining resources
DB2 entries 37
DB2 transactions 39
FEPI nodes 46
FEPI pools 46
FEPI property sets 46
FEPI targets 46
DEFSCREEN attribute
TYPETERM definition 237
DELETE attribute
FILE definition 49
DELETE command
CEDA 290

DELETE command, DFHCSDUP utility program
examples 336
DELIVER option
PERFORM operand 372
Dependent default values 218–225
DESC operand
DFHCLT TYPE=WTO 356
DESCRIPTION attribute
DOCTEMPLATE definition 41
ENQMODEL definition 44
FILE definition 49
IPCONN definition 66
PROCESSTYPE definition 113
PROGRAM definition 121
SESSIONS 134
TDQUEUE definition 159
TRANSACTION definition 198
TYPETERM definition 238
DEVICE operand
DFHTCT TYPE=SCSCI 406
sequential devices 406
devices with LDC lists 175
DFHCLT operands
TYPE=COMMAND 355
TYPE=INITIAL 353
TYPE=LISTEND 356
TYPE=LISTSTART 354
TYPE=WTO 355
DFHCSDUP system definition utility program
ADD command 323
ALTER command 325
APPEND command 327
COPY command 329
DEFINE command 332
DELETE command 334
EXTRACT command 337
INITIALIZE command 338
LIST command 338
PROCESS command 340
REMOVE command 341
SCAN command 342
SERVICE command 345
UPGRADE command 345
USERDEFINE command 347
VERIFY command 349
DFHDLPSB operands
TYPE=ENTRY 356
TYPE=INITIAL 356
DFHFCT operands
TYPE=FILE 358
TYPE=INITIAL 357
DFHMCT operands
TYPE=EMP 369
TYPE=INITIAL 366
TYPE=RECORD 372
DFHPLT operands
TYPE=ENTRY 388
TYPE=INITIAL 388
DFHRDT, RDO-eligible TCT entries 395
DFHRST operands
TYPE=INITIAL 391
TYPE=RSE 391
TYPE=SUBSYS 391
DFHSIT macro
DFHSIT macro *(continued)*
APPLID operand 8
DFHSRT operands
TYPE=INITIAL 392
TYPE=SYSTEM 393
TYPE=USER 393
DFHTCT operands
remote terminals 411, 412
sequential devices 406–408
TYPE=GROUP 397
TYPE=INITIAL 396
TYPE=LINE 407
TYPE=REGION 411
TYPE=REMOTE 412
TYPE=SDSCI 406
TYPE=TERMINAL 408, 411
DFHTLT operands
TYPE=ENTRY 415
TYPE=INITIAL 414
DFHTST operands
TYPE=ENTRY 418
TYPE=INITIAL 418
TYPE=RECOVERY 418, 420, 421
TYPE=SHARED 422
DFHXLT operands
TYPE=INITIAL 424
DISCREQ attribute
SESSIONS definition 134
TYPETERM definition 238
DISP operand
DFHFCT TYPE=FILE 359
DISPLAY command
CEDA 293
display devices
associating with printers 174
DISPLAY GROUP command
CEDA 295
DISPLAY LIST command
CEDA 295
DISPOSITION attribute
FILE definition 49
TDQUEUE definition 159
DL/I (data language 1)
DL/I PSB directory (PDIR) 356
DOCDEF object 43
DOCTEMPLATE definition
APPENDCRLF attribute 40
DDName attribute 40
DESCRIPTION attribute 41
EXITPGM attribute 41
FILE attribute 41
HFSFILE attribute 41
MEMBERNAME attribute 42
NAME attribute 41
PROGRAM attribute 42
TDQUEUE attribute 42
TEMPLATENAME attribute 42
TSQUEUE attribute 42
TYPE attribute 43
DOCTEMPLATE resources 39
Document template definition view 43
DROLLBACK attribute
DB2CONN definition 26
DB2ENTRY definition 35

DSCNAME operand
DFHTCT TYPE=SDSCI 406
sequential devices 406
DSN operand
DFHTCT TYPE=LDC 401
DSNAME attribute
FILE definition 49
LIBRARY definition 85
TDQUEUE definition 159
DSNAME operand
DFHFCT TYPE=FILE 359
DSNSHARING attribute
FILE definition 49
DTIMOUT attribute
TRANSACTION definition 198
dual-purpose definition 177
dual-purpose resource definition
for transient data 154
DUALCASEKYBD attribute
TYPETERM definition 239
dummy TDQ 171
DUMP attribute
TRANSACTION definition 199
DVC operand
DFHTCT TYPE=LDC 401
DVSUPRT attribute
PROFILE definition 115
DYNAMIC attribute
PROGRAM definition 121
TRANSACTION definition 199

E

ENQMDEF object 46
ENQMODEL attribute
ENQMODEL definition 44
ENQMODEL definition
DESCRIPTION attribute 44
ENQMODEL attribute 44
ENQNAME attribute 44
ENQSCOPE attribute 44
installing definitions 45
STATUS attribute 45
ENQMODEL resources 43
ENQNAME attribute
ENQMODEL 44
ENQSCOPE attribute
ENQMODEL definition 44
Enqueue model definitions view 46
ENTRY attribute
DB2TRAN definition 38
ERRATT operand
DFHTCT TYPE=INITIAL 396
ERRCOLOR attribute
TYPETERM definition 239
ERRHILIGHT attribute
TYPETERM definition 239
ERRINTENSIFY attribute
TYPETERM definition 239
ERRLASTLINE attribute
TYPETERM definition 239
ERROPTION attribute
TDQUEUE definition 159
event, monitoring point
event, monitoring point *(continued)*
DFHMCT TYPE=EMP 369
examples
FCT 365
MCT 386
PLT 389
RST 392
SRT 394
TLT 416
XLT 425
EXCLUDE operand
DFHMCT TYPE=RECORD 372
EXCNT option
PERFORM operand 371
EXECKEY attribute
PROGRAM definition 121
EXECUTIONSET attribute
PROGRAM definition 121
EXITPGM attribute
DOCTEMPLATE definition 41
EXPAND command
CEDA 297
EXPAND GROUP command
CEDA 299
EXPAND LIST command
CEDA 300
EXPIRYINTMIN attribute
TSMODEL definition 212
extended local LDC list and system LDC table
DFHTCT TYPE=LDC 401
extended recovery facility (XRF)
command list table (CLT) 353
EXTENDEDDS attribute
TYPETERM definition 239
EXTRACT command, DFHCSDUP utility program
examples 338
OBJECTS option 337
USERPROGRAM option 337
EXTSEC attribute
TRANSACTION definition 427

F

FACILITYID attribute
TDQUEUE definition 160
FACILITYLIKE attribute
PROFILE definition 116
FCT (file control table)
coding example 365
DFHFCT macros 357
DFHFCT TYPE=FILE 358
DFHFCT TYPE=INITIAL 357
summary table 364
FENODDEF object 46
FEPI node list definitions view 46
FEPI pool definitions view 46
FEPI property set definitions view 46
FEPI target definition 46
FEPI target list definitions view 46
FEPOODEF object 46
FEPRODEF object 46
FETRGDEF object 46
FIELD operand
DFHMCT TYPE=EMP 370

FILE attribute
DOCTEMPLATE definition 41
FILE definition 49
PROCESSTYPE definition 113
file control table (FCT)
coding example 365
DFHFCT macros 357
DFHFCT TYPE=FILE 358
DFHFCT TYPE=INITIAL 357
summary table 364
FILE definition
ADD attribute 49
BACKUPTYPE attribute 49, 50
BROWSE attribute 49
CFDTPOOL attribute 49
DATABUFFERS attribute 49
DELETE attribute 49
DESCRIPTION attribute 49
DISPOSITION attribute 49
DSNAME attribute 49
DSNSHARING attribute 49
FILE attribute 49
FWDRECOVLOG attribute 49
INDEXBUFFERS attribute 49
installing definitions 64
JNLADD attribute 49
JNLREAD attribute 49
JNLSYNCREAD attribute 49
JNLSYNCWRITE attribute 49
JNLUPDATE attribute 49
JOURNAL attribute 49
KEYLENGTH attribute 49
LOAD attribute 49
LSRPOOLID attribute 49
LSRPOOLNUM attribute 49
MAXNUMRECS attribute 49
NSRGROUP attribute 49
OPENTIME attribute 49
PASSWORD attribute 49
READINTEG attribute 49
RECORDFORMAT attribute 49
RECORDSIZE attribute 49
RECOVERY attribute 49
REMOTENAME attribute 60
REMOTESYSTEM attribute 60
RESSECNUM attribute 427
RLSACCESS attribute 61
STATUS attribute 62
STRINGS attribute 62
TABLE attribute 62
TABLENAME attribute 63
UPDATE attribute 64
UPDATEMODEL attribute 64
File definitions view 65
FILE operand
DFHFCT TYPE=FILE 360
FILE resources 46
FILE, DFHFCT TYPE= macro 358
FILEDEF object 65
files
relationship with data sets 357
FILSTAT operand
DFHFCT TYPE=FILE 360
FINAL option
FINAL option *(continued)*
LOCAL operand 403
FIXED option
RECFORM operand 363
FMHPARM attribute
TYPETERM definition 240
FORALT operand
DFHCLT TYPE=LISTSTART 354
FORMFEED attribute
TYPETERM definition 240
FWDRECOVLOG attribute
FILE definition 49

G

generic naming in the ALTER command 326
generic naming in the COPY command 330
GROUP operand
DFHTCT TYPE=GROUP 398

H

HFSFILE attribute
DOCTEMPLATE definition 41
URIMAP definition 260
HILIGHT attribute
TYPETERM definition 240
HORIZFORM attribute
TYPETERM definition 241
HOST attribute
IPCONN definition 66
URIMAP definition 262
HOSTCODEPAGE attribute
URIMAP definition 263
HSDATA12K attribute
LSRPOOL definition 91
HSDATA16K attribute
LSRPOOL definition 91
HSDATA20K attribute
LSRPOOL definition 91
HSDATA24K attribute
LSRPOOL definition 91
HSDATA28K attribute
LSRPOOL definition 91
HSDATA32K attribute
LSRPOOL definition 91
HSDATA4K attribute
LSRPOOL definition 91
HSDATA8K attribute
LSRPOOL definition 91
HSINDEX12K attribute
LSRPOOL definition 92
HSINDEX16K attribute
LSRPOOL definition 92
HSINDEX20K attribute
LSRPOOL definition 92
HSINDEX24K attribute
LSRPOOL definition 92
HSINDEX28K attribute
LSRPOOL definition 92
HSINDEX32K attribute
LSRPOOL definition 92
HSINDEX4K attribute

HSINDEX4K attribute *(continued)*
LSRPOOL definition 92
HSINDEX8K attribute
LSRPOOL definition 92

I

ID operand
DFHMCT TYPE=EMP 369
IDPROP attribute
IPCONN definition 66
IMS
defining CICS-IMS links and sessions 8
INAREAL operand
DFHTCT TYPE=LINE 407
sequential devices 407
INBFMH attribute
PROFILE definition 116
INCLUDE operand
DFHMCT TYPE=RECORD 385
INDEX12K attribute
LSRPOOL definition 92
INDEX16K attribute
LSRPOOL definition 92
INDEX1K attribute
LSRPOOL definition 92
INDEX20K attribute
LSRPOOL definition 92
INDEX24K attribute
LSRPOOL definition 92
INDEX28K attribute
LSRPOOL definition 92
INDEX2K attribute
LSRPOOL definition 92
INDEX32K attribute
LSRPOOL definition 92
INDEX4K attribute
LSRPOOL definition 92
INDEX512 attribute
LSRPOOL definition 92
INDEX8K attribute
LSRPOOL definition 92
INDEXBUFFERS attribute
FILE definition 49
indirect connections
defining 8
INDIRECT connections
resource definitions 8
INDIRECTNAME attribute
TDQUEUE definition 160
INDOUBT attribute
TRANSACTION definition 427
INDSYS attribute
CONNECTION
for INDIRECT connections 8
CONNECTION definition 13
INITIAL option
LOCAL operand 403
INITIALIZE command, DFHCSDUP utility program 338
INITQNAME attribute
MQCONN definition 96
INPUT option
TRMSTAT operand 409
INSERVICE attribute
CONNECTION definition 13
IPCONN definition 66
SESSIONS definition 427
TERMINAL definition 184
INSTALL command
CEDA 301
installation of resource definitions
transient data queues 170
INTENSIFY option
ERRATT operand 397
IOAREALEN attribute
SESSIONS definition 135
TYPETERM definition 241
IPADDRESS attribute
TCPIPSERVICE definition 142
IPCONDEF object 75
IPCONN attribute
IPCONN definition 66
IPCONN definition
APPLID attribute 66
AUTOCONNECT attribute 66
CERTIFICATE attribute 66
CIPHERS attribute 66
DESCRIPTION attribute 66
HOST attribute 66
IDPROP attribute 66
INSERVICE attribute 66
IPCONN attribute 66
MAXQTIME attribute 66
MIRRORLIFE attribute 66
NETWORKID attribute 66
PORT attribute 66
QUEUELIMIT attribute 66
RECEIVECOUNT attribute 66
SECURITYNAME attribute 66
SENDCOUNT attribute 66
SSL attribute 66
TCPIPSERVICE attribute 66
USERAUTH attribute 66
XLNACTION attribute 66
IPCONN definitions
installing 74
IPIC connection definitions view 75
ISADSCN operand
DFHTCT TYPE=LINE 407
sequential devices 407
ISC/MRO connection definitions view 20
ISOLATE attribute
TRANSACTION definition 200

J

JES operands
DFHCLT TYPE=INITIAL 354
JESCHAR operands
DFHCLT TYPE=INITIAL 354
JESID operand
DFHCLT TYPE=INITIAL 354
JID operand
DFHFCT TYPE=FILE 361
JNLADD attribute
FILE definition 49
JNLREAD attribute

JNLREAD attribute *(continued)*
FILE definition 49
JNLSYNCREAD attribute
FILE definition 49
JNLSYNCWRITE attribute
FILE definition 49
JNLUPDATE attribute
FILE definition 49
JOBNAME operand
DFHRST TYPE=SUBSYS 392
JOURNAL attribute
FILE definition 49
PROFILE definition 116
Journal model definitions view 80
JOURNALMODEL attribute
JOURNALMODEL definition 76
JOURNALMODEL definition
JOURNALMODEL attribute 76
JOURNALNAME attribute 76
STREAMNAME attribute 77
TYPE attribute 79
JOURNALMODEL resources 75
JOURNALNAME attribute
JOURNALMODEL definition 76
JREQ operand
DFHFCT TYPE=FILE 361
JRNMDEF object 80
JVM attribute
PROGRAM definition 121
JVMCLASS attribute
PROGRAM definition 121
JVMPROFILE attribute
PROGRAM definition 121
JVMSERVER resources 81

K

KATAKANA attribute
TYPETERM definition 241
KEY option
SERVREQ operand 363
KEYLEN operand
DFHFCT TYPE=FILE 362
KEYLENGTH attribute
FILE definition 49

L

LASTLINE option
ERRATT operand 397
LDC (logical device code) 398, 404
LDC list
extended local, and system LDC table, DFHTCT TYPE=LDC 401
local, DFHTCT TYPE=LDCLIST 404
LDC lists
devices with 175
LDC operand
DFHTCT TYPE=LDC 402
DFHTCT TYPE=LDCLIST 404
DFHTLT TYPE=INITIAL 414
LDCLIST attribute
TYPETERM definition 241
LDCs for 3770 batch logical unit
TCT example 405
LIBDEF object 88
LIBRARY definition
CRITICAL attribute 84
DSNAME attribute 85
installing definitions 87
RANKING attribute 86
STATUS attribute 87
LIBRARY definitions view 88
LIGHTPEN attribute
TYPETERM definition 242
LINSTAT='OUT OF SERVICE' operand
DFHTCT TYPE=LINE 407
sequential devices 407
LIST command, DFHCSDUP utility program
examples 339
OBJECTS option 339
LOAD attribute
FILE definition 49
local definition 176
local LDC list
DFHTCT TYPE=LDCLIST 404
LOCAL operand
DFHTCT TYPE=LDC 403
local temporary storage DATAIDs
DFHTST TYPE=LOCAL 420
LOCALQ attribute
TRANSACTION definition 201
LOCATION attribute
TSMODEL definition 213
URIMAP definition 263
LOCK command
CEDA 304
LOG operand
DFHFCT TYPE=FILE 362
logical device code (LDC) 398, 404
LOGMODE attribute
TYPETERM definition 242
LOGMODECOM attribute
TRANSACTION definition 427
LOGONMSG attribute
TYPETERM definition 243
LOGREC attribute
PROFILE definition 117
LPLEN operand
DFHTCT TYPE=TERMINAL 408
sequential devices 408
LRECL operand
DFHFCT TYPE=FILE 362
LSR pool definitions view 93
LSRDEF object 93
LSRPOOL attribute
LSRPOOL definition 93
LSRPOOL definition
DATA12K attribute 90
DATA16K attribute 90
DATA1K attribute 90
DATA20K attribute 90
DATA24K attribute 90
DATA28K attribute 90
DATA2K attribute 90
DATA32K attribute 91
DATA4K attribute 90

LSRPOOL definition *(continued)*
DATA512 attribute 90
DATA8K attribute 90
HSDATA12K attribute 91
HSDATA16K attribute 91
HSDATA20K attribute 91
HSDATA24K attribute 91
HSDATA28K attribute 91
HSDATA32K attribute 91
HSDATA4K attribute 91
HSDATA8K attribute 91
HSINDEX12K attribute 92
HSINDEX16K attribute 92
HSINDEX20K attribute 92
HSINDEX24K attribute 92
HSINDEX28K attribute 92
HSINDEX32K attribute 92
HSINDEX4K attribute 92
HSINDEX8K attribute 92
INDEX12K attribute 92
INDEX16K attribute 92
INDEX1K attribute 92
INDEX20K attribute 92
INDEX24K attribute 92
INDEX28K attribute 92
INDEX2K attribute 92
INDEX32K attribute 92
INDEX4K attribute 92
INDEX512 attribute 92
INDEX8K attribute 92
LSRPOOL attribute 93
LSRPOOLID attribute 93, 427
LSRPOOLNUM attribute 93
MAXKEYLENGTH attribute 93
SHARELIMIT attribute 93
STRINGS attribute 93
LSRPOOL resources 88
LSRPOOLID 88
LSRPOOLID attribute
FILE definition 49
LSRPOOL definition 93, 427
LSRPOOLNUM 88
LSRPOOLNUM attribute
FILE definition 49
LSRPOOL definition 93
LU6.2 connection definition 20
LUTYPE4 option
LDC operand 403
LUTYPE6.1
CICS-CICS ISC links and sessions
resource definitions 7
CICS-IMS links and
sessions
resource definitions 8
MRO sessions
resource definitions 6
LUTYPE6.1 CICS-CICS ISC links and sessions 131
LUTYPE6.1 CICS-IMS links and sessions 131

M

MACRF operand
DFHTCT TYPE=SDSCI 406
sequential devices 406
Map set definitions view 95
MAPDEF object 95
MAPSET definition
MAPSET attribute 94
RESIDENT attribute 95
RSL attribute 427
USAGE attribute 95
USELPACOPY attribute 95
MAPSET resources 93
MAXACTIVE attribute
TRANCLASS definition 193
MAXDATALEN attribute
TCPIPSERVICE definition 142
MAXIMUM attribute
for APPC parallel sessions 7
SESSIONS
for APPC single session terminals 7
SESSIONS definition 135
MAXKEYLENGTH attribute
LSRPOOL definition 93
MAXNUMRECS attribute
FILE definition 49
MAXPERSIST attribute
TCPIPSERVICE definition 142
MAXQTIME attribute
CONNECTION definition 14
IPCONN definition 66
MCT (monitoring control table)
coding examples 386
control data recording 372
DFHMCT macros 366
DFHMCT TYPE=EMP 369
DFHMCT TYPE=INITIAL 366
DFHMCT TYPE=RECORD 372
user event monitoring points 369
MEDIATYPE attribute
URIMAP definition 263
MEMBERNAME attribute
DOCTEMPLATE definition 42
MF operand
DFHCLT TYPE=WTO 356
MIGRATE operand
DFHTCT TYPE=INITIAL 397
DFHTST TYPE=INITIAL 418
migrating TCT entries to RDO 397
MIRRORLIFE attribute
IPCONN definition 66
MLTCNT option
PERFORM operand 371
MODENAME attribute
PROFILE definition 117
SESSIONS
for APPC parallel sessions 7
for APPC single session terminals 7
SESSIONS definition 135
TERMINAL definition 184
monitoring control table (MCT)
coding examples 386
control data recording 372
DFHMCT macros 366
DFHMCT TYPE=EMP 369
DFHMCT TYPE=INITIAL 366
DFHMCT TYPE=RECORD 372
user event monitoring points 369

monitoring point, event
DFHMCT TYPE=EMP 369
MOVE command
CEDA 306
MQCONDEF object 99
MQCONN
installing 95
MQCONN attribute
MQCONN definition 96
MQCONN definition
INITQNAME attribute 96
MQCONN attribute 96
MQNAME attribute 96
RESYNCMEMBER attribute 96
MQCONN resources 95
MQNAME attribute
MQCONN definition 96
MRO links and sessions 131
MSGINTEG attribute
PROFILE definition 117
MSGJRNL attribute
PROFILE definition 117
MSGQUEUE1 attribute
DB2CONN definition 22
MSGQUEUE2 attribute
DB2CONN definition 22
MSGQUEUE3 attribute
DB2CONN definition 22
MSRCONTROL attribute
TYPETERM definition 243
multiregion operation (MRO)
resource definitions for links and sessions 6
MXSSASZ operand
DFHDLPSB TYPE=ENTRY 357

N

NACNT option
PERFORM operand 371
NAME attribute
DOCTEMPLATE definition 41
NATLANG attribute
TERMINAL definition 184
NEPCLASS attribute
PROFILE definition 117
SESSIONS definition 136
TYPETERM definition 243
NETNAME attribute
CONNECTION
for INDIRECT connections 8
CONNECTION definition 14
PARTNER definition 106
TERMINAL definition 185
NETNAMEQ attribute
SESSIONS
for LUTYPE6.1 CICS-IMS sessions 8
SESSIONS definition 136
NETWORK attribute
PARTNER definition 106
NETWORKID attribute
IPCONN definition 66
NOEXCTL option
SERVREQ operand 363
NONTERMREL attribute
DB2CONN definition 22
NONVTAM option
specifying ACCMETH=NONVTAM 396
NSRGROUP attribute
FILE definition 49

O

OBFORMAT attribute
TYPETERM definition 244
OBJECTS option
LIST command (DFHCSDUP) 339
OBOPERID attribute
TYPETERM definition 244
ONEWTE attribute
PROFILE definition 118
OPENTIME attribute
FILE definition 49
TDQUEUE definition 160
OPERID attribute
SESSIONS definition 427
OPERPRIORITY attribute
SESSIONS definition 427
OPERRSL attribute
SESSIONS definition 427
OPERSECURITY attribute
SESSIONS definition 427
ORCNT option
PERFORM operand 371
OSADSCN operand
DFHTCT TYPE=LINE 407
sequential devices 407
OUT OF SERVICE option
LINSTAT operand 407
TRMSTAT operand 409
OUTLINE attribute
TYPETERM definition 244

P

PAGESIZE attribute
TYPETERM definition 244
parallel APPC sessions 7
parallel sessions and APPC (LUTYPE6.2) links 131
PARTDEF object 108
Partition set definitions view 105
PARTITIONS attribute
TYPETERM definition 245
PARTITIONSET attribute
TRANSACTION definition 202
PARTITIONSET definition
PARTITIONSET attribute 104
RESIDENT attribute 104
RSL attribute 427
STATUS attribute 104
USAGE attribute 104
USELPACOPY attribute 104
PARTITIONSET resources 102
PARTNER attribute
PARTNER definition 106
PARTNER definition
NETNAME attribute 106

PARTNER definition *(continued)*
NETWORK attribute 106
PARTNER attribute 106
PROFILE attribute 107
TPNAME attribute 107
XTPNAME attribute 107
Partner definitions view 108
PARTNER resources 105
PASSWORD attribute
FILE definition 49
PATH attribute
URIMAP definition 263
PCLOCK option
PERFORM operand 372
PCPUCLK option
PERFORM operand 372
PDIR (program specification block directory)
DFHDLPSB macros 356
DFHDLPSB TYPE=ENTRY 356
DFHDLPSB TYPE=INITIAL 356
PERFORM operand
DFHMCT TYPE=EMP 370
PGESIZE operand
DFHTCT TYPE=LDC 403
DFHTCT TYPE=TERMINAL 408
sequential devices 408
PGESTAT operand
DFHTCT TYPE=LDC 404
PIPELINE attribute
URIMAP definition 264
PIPELINE definition
RESPWAIT attribute 110
SHELF attribute 110
WSDIR attribute 110
PIPELINE resources 108
pipeline TERMINAL definitions 174
pipeline terminals 229
PLAN attribute
DB2CONN definition 26
DB2ENTRY definition 35
PLANEXITNAME attribute
DB2CONN definition 26
DB2ENTRY definition 35
PLT (program list table)
coding example 389
control section 388
DFHPLT TYPE=ENTRY 388
DFHPLT TYPE=INITIAL 388
POOL attribute
TERMINAL definition 185
POOLNAME attribute
TSMODEL definition 213
PORT attribute
IPCONN definition 66
URIMAP definition 265
PORTNUMBER attribute
TCPIPSERVICE definition 142
PREFIX attribute
TSMODEL definition 214
preset security 409, 427
preset terminal security
consoles 190
PRIMEDSIZE attribute
TRANSACTION 427
PRINTADAPTER attribute
TYPETERM definition 245
PRINTCONTROL attribute
TDQUEUE definition 160
PRINTER attribute
TERMINAL definition 186
PRINTERCOMP attribute
PROFILE definition 118
PRINTERCOPY attribute
TERMINAL definition 186
printers
associating with display devices 174
PGESTAT operand 404
terminal definitions for 173
printing
terminals for 172
printing, continuous (autopaging) 404
PRIORITY attribute
DB2ENTRY definition 35
TRANSACTION definition 202
PRIVACY attribute
TCPIPSERVICE definition 427
PROCDEF object 114
PROCESS command, DFHCSDUP utility program 340
Process type definitions view 114
PROCESSTYPE attribute
PROCESSTYPE definition 113
PROCESSTYPE definition
AUDITLEVEL attribute 112
AUDITLOG attribute 113
DESCRIPTION attribute 113
FILE attribute 113
PROCESSTYPE attribute 113
PROCESSTYPE resources 111
PROFDEF object 120
PROFILE attribute
PARTNER definition 107
PROFILE definition 118
TRANSACTION definition 202
PROFILE definition
CHAINCONTROL attribute 115
CICS-supplied 448, 451
DVSUPRT attribute 115
FACILITYLIKE attribute 116
INBFMH attribute 116
JOURNAL attribute 116
LOGREC attribute 117
MODENAME attribute 117
MSGINTEG attribute 117
MSGJRNL attribute 117
NEPCLASS attribute 117
ONEWTE attribute 118
PRINTERCOMP attribute 118
PROFILE attribute 118
PROTECT attribute 427
RAQ attribute 118
RTIMOUT attribute 119
SCRNSIZE attribute 119
UCTRAN attribute 119
Profile definitions view 120
PROFILE resources 114
profiles, CICS-supplied defaults
DFHCICSA 449
DFHCICSE 449

profiles, CICS-supplied defaults *(continued)*
DFHCICSF 448
DFHCICSP 450
DFHCICSR 448
DFHCICSS 449
DFHCICST 450
DFHCICSV 450
DFHECEPH 448
DFHPGAMP 451
DFHPGAPG 451
DFHPGAPT 451
DFHPPF01 450
DFHPPF02 451
PROGDEF object 131
PROGRAM attribute
DOCTEMPLATE definition 42
PROGRAM definition 121
TRANSACTION definition 203
URIMAP definition 265
PROGRAM definition
API attribute 121
CEDF attribute 121
CONCURRENCY attribute 121
DATALOCATION attribute 121
DESCRIPTION attribute 121
DYNAMIC attribute 121
EXECKEY attribute 121
EXECUTIONSET attribute 121
JVM attribute 121
JVMCLASS attribute 121
JVMPROFILE attribute 121
JVMSERVER attribute
PROGRAM definition 121
LANGUAGE attribute
PROGRAM definition 121
RPG 427
PROGRAM attribute 121
PROGRAM definition
JVMSERVER attribute 121
LANGUAGE attribute 121
RELOAD attribute 121
REMOTENAME attribute 121
REMOTESYSTEM attribute 121
RESIDENT attribute 121
STATUS attribute 121
TRANSID attribute 121
USAGE attribute 121
USELPACOPY attribute 121
RELOAD attribute
PROGRAM definition 121
REMOTENAME attribute
PROGRAM definition 121
REMOTESYSTEM attribute
PROGRAM definition 121
RESIDENT attribute
PROGRAM definition 121
RSL attribute 427
STATUS attribute
PROGRAM definition 121
TRANSID attribute
PROGRAM definition 121
USAGE attribute
PROGRAM definition 121
USELPACOPY attribute
PROGRAM definition *(continued)*
USELPACOPY attribute *(continued)*
PROGRAM definition 121
Program definitions view 131
program list table (PLT)
coding example 389
control section 388
DFHPLT TYPE=ENTRY 388
DFHPLT TYPE=INITIAL 388
PROGRAM operand
DFHPLT TYPE=ENTRY 388
PROGRAM resources 120
program specification block (PSB) 356, 357
PROGSYMBOLS attribute
TYPETERM definition 246
PROTECT attribute
PROFILE 427
PROTECTNUM attribute
DB2ENTRY definition 36
PROTOCL(HTTP) TCPIPSERVICES 151
PROTOCOL attribute
CONNECTION
for APPC links 7
for APPC single session terminals 7
for LUTYPE6.1 CICS-IMS links 8
for LUTYPE6.1 links 7
for MRO links 6
CONNECTION definition 16
SESSIONS
for APPC parallel sessions 7
for APPC single session terminals 7
for LUTYPE6.1 CICS-IMS sessions 8
for LUTYPE6.1 sessions 7
for MRO sessions 6
SESSIONS definition 136
TCPIPSERVICE definition 142
PRTNDEF object 105
PSB (program specification block) 356, 357
PSB directory (PDIR)
DFHDLPSB macros 356
DFHDLPSB TYPE=ENTRY 356
DFHDLPSB TYPE=INITIAL 356
PSB operand
DFHDLPSB TYPE=ENTRY 357
PSRECOVERY attribute
CONNECTION definition 16
PURGECYCLE attribute
DB2CONN definition 22
PURGETHRESH attribute
TRANCLASS definition 193

Q

QUERY attribute
TYPETERM definition 246
QUEUELIMIT attribute
CONNECTION definition 16
IPCONN definition 66

R

RANKING attribute

RANKING attribute *(continued)*
LIBRARY definition 86
RAQ attribute
PROFILE definition 118
RDO-eligible TCT entries, DFHRDT 395
READ option
SERVREQ operand 363
READINTEG attribute
FILE definition 49
REALM attribute
TCPIPSERVICE definition 142
RECEIVE option
TRMSTAT operand 409
RECEIVECOUNT attribute
IPCONN definition 66
SESSIONS
for LUTYPE6.1 CICS-IMS sessions 8
for LUTYPE6.1 sessions 7
for MRO sessions 6
SESSIONS definition 136
RECEIVEPFX attribute
SESSIONS
for LUTYPE6.1 sessions 7
for MRO sessions 6
SESSIONS definition 137
RECEIVESIZE attribute
SESSIONS definition 137
TYPETERM definition 246
RECFM operand
DFHTCT TYPE=SDSCI 406
sequential devices 406
RECFORM operand
DFHFCT TYPE=FILE 362
RECORDFORMAT attribute
CONNECTION definition 16
FILE definition 49
TDQUEUE definition 160
RECORDSIZE attribute
FILE definition 49
TDQUEUE definition 161
RECOVER operand
DFHSRT TYPE=SYSTEM 393
DFHSRT TYPE=USER 393
recoverable service table (RST)
coding example 392
DFHRST macro types 391
DFHRST TYPE=INITIAL 391
DFHRST TYPE=RSE 391
DFHRST TYPE=SUBSYS 391
recoverable temporary storage
DFHTST TYPE=RECOVERY 418
RECOVERY attribute
FILE definition 49
RECOVNOTIFY attribute
SESSIONS definition 427
TYPETERM definition 247
RECOVOPTION attribute
SESSIONS definition 138
TYPETERM definition 248
RECOVSTATUS attribute
TDQUEUE definition 161
TSMODEL definition 214
REDIRECTTYPE attribute
REDIRECTTYPE attribute *(continued)*
URIMAP definition 265
RELREQ attribute
SESSIONS definition 139
TYPETERM definition 249
RELTYPE operand
DFHFCT TYPE=FILE 363
remote definition 177
remote files 47
remote temporary storage DATAIDs
DFHTST TYPE=REMOTE 420
remote terminals for transaction routing
DFHTCT macros 410
DFHTCT TYPE=REGION operands 411
DFHTCT TYPE=REMOTE operands 412
DFHTCT TYPE=TERMINAL operands 411
REMOTELENGTH attribute
TDQUEUE definition 162
REMOTENAME attribute
CONNECTION definition 17
FILE definition 60
TDQUEUE definition 162
TERMINAL definition 186
TRANSACTION definition 203
REMOTEPREFIX attribute
TSMODEL definition 214
REMOTESYSNET attribute
CONNECTION definition 17
TERMINAL definition 187
REMOTESYSTEM attribute
CONNECTION definition 17
FILE definition 60
TDQUEUE definition 162
TERMINAL definition 187
TRANSACTION definition 203
TSMODEL definition 215
REMOVE command
CEDA 309
REMOVE command, DFHCSDUP utility program
examples 341
RENAME command
CEDA 311
resource definition views
Atomservice definitions 4
DB2 connection definitions 31
DB2 entry definition 37
DB2 transaction resource definitions 39
Document template resource definitions 43
Enqueue model definitions 46
FEPI node list definitions 46
FEPI pool definitions 46
FEPI property set definitions 46
FEPI target list definitions 46
FILE definitions 65
IPIC connection definitions view 75
ISC/MRO connection definitions 20
Journal model definitions 80
LIBRARY definitions 88
LSR pool definitions 93
Map set definitions 95
Partition set definitions 105
Partner definitions 108
Process type definitions 114
Profile definitions 120

resource definition views *(continued)*
Program definitions 131
Session definitions 141
TCP/IP service definitions 154
Temporary storage model definitions 216
Terminal definitions 192
Transaction class definitions 194
Transaction definitions 211
Transient data queue definitions 172
Typeterm definitions 254
URI mapping definitions 271
web service definitions 274
WebSphere MQ connection definitions 99
resource definitions
for APPC links and parallel sessions 7
for APPC single session terminals
advantages of TERMINAL-TYPETERM method 7
CONNECTION and SESSIONS method 7
for INDIRECT connections 8
for LUTYPE6.1 CICS-CICS ISC links and sessions 6
for LUTYPE6.1 CICS-IMS links and sessions 8
for LUTYPE6.1 links and sessions 7
for MRO links and sessions 6
RESOURCENAME attribute
ATOMSERVICE definition 3
RESOURCETYPE attribute
ATOMSERVICE definition 3
RESPWAIT attribute
PIPELINE definition 110
RESSEC attribute
TRANSACTION definition 204
RESSECNUM attribute
FILE definition 427
RESTART attribute
TRANSACTION definition 204
RESYNCMEMBER attribute
DB2CONN definition 22
MQCONN definition 96
REWIND attribute
TDQUEUE definition 163
RKP operand
DFHFCT TYPE=FILE 363
RLSACCESS attribute
FILE definition 61
RLSACCESS(YES)
effects on other FILE attributes 61
RMTNAME operand
DFHDLPSB TYPE=ENTRY 357
DFHTCT TYPE=REMOTE 412
DFHTCT TYPE=TERMINAL 411
DFHTST TYPE=REMOTE 421
remote terminals 411, 412
ROUTABLE attribute
TRANSACTION definition 204
ROUTCDE operand
DFHCLT TYPE=WTO 356
ROUTEDMSGS attribute
TYPETERM definition 249
RPG value
of LANGUAGE attribute
of PROGRAM definition 427
RSL attribute
MAPSET definition 427
RSL attribute *(continued)*
PARTITIONSET definition 427
PROGRAM definition 427
TRANSACTION definition 427
RST (recoverable service table)
coding example 392
DFHRST macro types 391
DFHRST TYPE=INITIAL 391
DFHRST TYPE=RSE 391
DFHRST TYPE=SUBSYS 391
RSTSIGNOFF attribute
TYPETERM definition 249
RTIMOUT attribute
PROFILE definition 119
RUNAWAY attribute
TRANSACTION definition 204

S

SCAN command, DFHCSDUP utility program 342
SCHEME attribute
URIMAP definition 266
SCLOCK option
PERFORM operand 372
SCPUCLK option
PERFORM operand 372
SCRNSIZE attribute
PROFILE definition 119
security
preset 409, 427
SECURITY attribute
TSMODEL definition 215
security checking
DFHTST TYPE=SECURITY 421
temporary storage 421
SECURITYNAME attribute
CONNECTION definition 18
IPCONN definition 66
TERMINAL definition 187
segment search argument, PSB 357
SENDCOUNT attribute
IPCONN definition 66
SESSIONS
for LUTYPE6.1 CICS-IMS sessions 8
for LUTYPE6.1 sessions 7
for MRO sessions 6
SESSIONS definition 139
SENDPFX attribute
SESSIONS
for LUTYPE6.1 sessions 7
for MRO sessions 6
SESSIONS definition 139
SENDSIZE attribute
SESSIONS definition 140
TYPETERM definition 249
sequential terminal devices
DFHTCT TYPE=LINE 405
DFHTCT TYPE=SDSCI 405
DFHTCT TYPE=TERMINAL 405
SERVICE command, DFHCSDUP utility program
FROMCSD operand 345
LEVEL operand 345
SERVREQ operand

SERVREQ operand *(continued)*
DFHFCT TYPE=FILE 363
SESSDEF object 141
Session definitions view 141
sessions and LUTYPE6.1 CICS-CICS ISC links 131
sessions and LUTYPE6.1 CICS-IMS links 131
sessions and MRO links 131
SESSIONS attribute
SESSIONS definition 140
SESSIONS definition
AUTOCONNECT attribute 133
BUILDCHAIN attribute 134
CONNECTION attribute 134
DESCRIPTION attribute 134
DISCREQ attribute 134
INSERVICE attribute 427
installation 131
IOAREALEN attribute 135
MAXIMUM attribute 135
MODENAME attribute 135
NEPCLASS attribute 136
NETNAMEQ attribute 136
OPERID attribute 427
OPERPRIORITY attribute 427
OPERRSL attribute 427
OPERSECURITY attribute 427
PROTOCOL attribute 136
RDO command 132
RECEIVECOUNT attribute 136
RECEIVEPFX attribute 137
RECEIVESIZE attribute 137
RECOVNOTIFY attribute 427
RECOVOPTION attribute 138
RELREQ attribute 139
SENDCOUNT attribute 139
SENDPFX attribute 139
SENDSIZE attribute 140
SESSIONS attribute 140
SESSNAME attribute 140
SESSPRIORITY attribute 140
TRANSACTION attribute 427
USERAREALEN attribute 140
USERID attribute 140
SESSIONS resources 131
SESSIONTYPE attribute
TYPETERM definition 250
SESSNAME attribute
SESSIONS
for LUTYPE6.1 CICS-IMS sessions 8
SESSIONS definition 140
SESSPRIORITY attribute
SESSIONS definition 140
SHARELIMIT attribute
LSRPOOL definition 93
Sharing CSD
between releases 437
SHELF attribute
PIPELINE definition 110
SHIPPABLE attribute
TYPETERM definition 250
shipping terminal definitions to application-owning regions 178
SHUTDOWN attribute
SHUTDOWN attribute *(continued)*
TRANSACTION definition 205
SIGNID attribute
DB2CONN definition 22
SIGNOFF attribute
TYPETERM definition 251
single session terminals
APPC
advantages of TERMINAL-TYPETERM method 7
resource definitions, CONNECTION and SESSIONS method 7
SINGLESESS attribute
CONNECTION
for APPC single session terminals 7
CONNECTION definition 18
SOCKETCLOSE attribute 142
SOLICITED attribute
TERMINAL definition 188
SOSI attribute
TYPETERM definition 251
SPURGE attribute
TRANSACTION definition 205
SRCHM operand
DFHFCT TYPE=FILE 364
SRT (system recovery table)
coding example 394
DFHSRT TYPE=INITIAL 392
SSL attribute
IPCONN definition 66
STANDBYMODE attribute
DB2CONN definition 22
STATSQUEUE attribute
DB2CONN definition 22
STATUS attribute
ATOMSERVICE definition 3
ENQMODEL definition 45
FILE definition 62
LIBRARY definition 87
TRANSACTION definition 205
URIMAP definition 266
storage queues, DFHTST TYPE=LOCAL
local temporary 420
storage queues, DFHTST TYPE=REMOTE
remote temporary 420
STORAGECLEAR attribute
TRANSACTION definition 205
STREAMNAME attribute
JOURNALMODEL definition 77
STRINGS attribute
FILE definition 62
LSRPOOL definition 93
SUBCNT option
PERFORM operand 372
SUBSYSID operand
DFHRST TYPE=SUBSYS 392
SUFFIX operand
DFHTLT TYPE=INITIAL 415
SYSIDNT operand
DFHDLPSB TYPE=ENTRY 357
DFHSIT TYPE=INITIAL 397
DFHTCT TYPE=INITIAL 397
DFHTCT TYPE=REGION 411
DFHTCT TYPE=REMOTE 412
DFHTCT TYPE=TERMINAL 411

SYSIDNT operand *(continued)*
DFHTST TYPE=REMOTE 421
lowercase in TCT 395
remote terminals 411, 412
SYSOUTCLASS attribute
TDQUEUE definition 163
system LDC table and extended local LDC list
DFHTCT TYPE=LDC 401
SYSTEM option
LDC operand 402
system recovery table (SRT)
coding example 394
DFHSRT TYPE=INITIAL 392

T

TABLE attribute
FILE definition 62
TABLENAME attribute
FILE definition 63
TASKDATAKEY attribute
TRANSACTION definition 205
TASKDATALOC attribute
TRANSACTION definition 206
TASKLIMIT attribute
TERMINAL definition 188
TASKREQ attribute
TRANSACTION definition 206
TASKREQ operand
DFHXLT TYPE=ENTRY 424
TCBLIMIT attribute
DB2CONN definition 22
TCLASS attribute
TRANSACTION definition 427
TCP/IP service definition 154
TCP/IP service definitions view 154
TCPDEF object 154
TCPIPSERVICE attribute
IPCONN definition 66
TCPIPSERVICE definition 142
URIMAP definition 266
TCPIPSERVICE definition
ATTACHSEC attribute 142
AUTHENTICATE attribute 142
BACKLOG attribute 142
CERTIFICATE attribute 142
CIPHERS attribute 142
IPADDRESS attribute 142
MAXDATALEN attribute 142
MAXPERSIST attribute 142
PORTNUMBER attribute 142
PRIVACY attribute 427
PROTOCOL attribute 142
REALM attribute 142
SOCKETCLOSE attribute 142
SSL attribute 142
STATUS attribute 142
TCPIPSERVICE attribute 142
TRANSACTION attribute 142
TSQPREFIX attribute 142
URM attribute 142
TCPIPSERVICE resources 141
TCT (terminal control table)
DFHTCT macros, summary 395
TCT (terminal control table) *(continued)*
DFHTCT TYPE=GROUP 397
introduction 394
list of CICS terminals 412
migrating entries to RDO 397
sequential devices 405
TCTUAL operand
DFHTCT TYPE=LINE 407
DFHTCT TYPE=TERMINAL 408
sequential devices 407, 408
TDQDEF object 172
TDQUEUE attribute
DOCTEMPLATE definition 42
TDQUEUE definition 163
TDQUEUE definition
ATIFACILITY attribute 157
BLOCKFORMAT attribute 158
BLOCKSIZE attribute 158
DATABUFFERS attribute 158
DDNAME attribute 159
DESCRIPTION attribute 159
DISPOSITION attribute 159
DSNAME attribute 159
ERROPTION attribute 159
FACILITYID attribute 160
INDIRECTNAME attribute 160
installing 170
OPENTIME attribute 160
PRINTCONTROL attribute 160
RECORDFORMAT attribute 160
RECORDSIZE attribute 161
RECOVSTATUS attribute 161
REMOTELENGTH attribute 162
REMOTENAME attribute 162
REMOTESYSTEM attribute 162
REWIND attribute 163
SYSOUTCLASS attribute 163
TDQUEUE attribute 163
TRANSID attribute 164
TRIGGERLEVEL attribute 164
TYPE attribute 164
TYPEFILE attribute 165
USERID attribute 165
WAIT attribute 166
WAITACTION attribute 166
TDQUEUE resources 154
TEMPLATENAME attribute
DOCTEMPLATE definition 42
URIMAP definition 267
temporary storage
data sharing, DFHTST TYPE=SHARED 422
recoverable, DFHTST TYPE=RECOVERY 418
security checking, DFHTST TYPE=SECURITY 421
temporary storage data sharing
DFHTST TYPE=SHARED 422
Temporary storage model definitions view 216
temporary storage security checking
DFHTST TYPE=SECURITY 421
temporary storage table (TST)
control section 418
data sharing temporary storage 422
DFHTST TYPE=INITIAL 418
DFHTST TYPE=RECOVERY 418, 420, 421
DFHTST TYPE=SHARED 422

temporary storage table (TST) *(continued)*
local temporary storage DATAIDs, DFHTST TYPE=LOCAL 420
recoverable temporary storage 418, 420, 421
recoverable temporary storage, DFHTST TYPE=RECOVERY 418
remote temporary storage DATAIDs, DFHTST TYPE=REMOTE 420
temporary storage data sharing, DFHTST TYPE=SHARED 422
temporary storage security checking, DFHTST TYPE=SECURITY 421
TERMDEF object 192
TERMINAL attribute
TERMINAL definition 188
terminal control table (TCT)
DFHTCT macros, summary 395
DFHTCT TYPE=GROUP 397
introduction 394
list of CICS terminals 412
migrating entries to RDO 397
sequential devices 405
terminal definition
summary 217
TERMINAL definition
ALTPRINTCOPY attribute 182
ALTPRINTER attribute 182
ATTACHSEC attribute 182
AUTINSTMODEL attribute 183
AUTINSTNAME attribute 183
BINDSECURITY attribute 183
CONSNAME attribute 183
CONSOLE attribute 427
INSERVICE attribute 184
MODENAME attribute 184
NATLANG attribute 184
NETNAME attribute 185
POOL attribute 185
PRINTER attribute 186
PRINTERCOPY attribute 186
REMOTENAME attribute 186
REMOTESYSNET attribute 187
REMOTESYSTEM attribute 187
SECURITYNAME attribute 187
SOLICITED attribute 188
TASKLIMIT attribute 188
TERMINAL attribute 188
TERMPRIORITY attribute 189
TRANSACTION attribute 189
TYPETERM attribute 189
USEDFLTUSER attribute 190
USERID attribute 190
TERMINAL definitions
installing 191
pipeline 174
Terminal definitions view 192
terminal list table (TLT)
coding examples 416
control section 414
DFHTLT TYPE=ENTRY 415
DFHTLT TYPE=INITIAL 414
TERMINAL resources 172
terminals
for printing 172
terminals *(continued)*
supported by RDO 225
terminals for transaction routing 176
terminals, CICS, list of 412
TERMMODEL attribute
TYPETERM definition 251
TERMPRIORITY attribute
TERMINAL definition 189
TEXTKYBD attribute
TYPETERM definition 251
TEXTPRINT attribute
TYPETERM definition 251
THREADERROR attribute
DB2CONN definition 22
THREADLIMIT attribute
DB2CONN definition 26
DB2ENTRY definition 36
THREADWAIT attribute
DB2CONN definition 26
DB2ENTRY definition 36
TLT (terminal list table)
coding examples 416
control section 414
DFHTLT TYPE=ENTRY 415
DFHTLT TYPE=INITIAL 414
TPNAME attribute
PARTNER definition 107
TRANSACTION definition 207
TPURGE attribute
TRANSACTION definition 207
TRACE attribute
TRANSACTION definition 207
TRANCLASS attribute
TRANSACTION definition 193, 207
TRANCLASS definition
MAXACTIVE attribute 193
PURGETHRESH attribute 193
TRANCLASS resources 192
TRANDEF object 211
TRANSACTION attribute
SESSIONS definition 427
TERMINAL definition 189
TRANSACTION definition 208
URIMAP definition 268
Transaction class definitions view 194
TRANSACTION definition
ACTION attribute 197
ALIAS attribute 197
BREXIT attribute 198
CMDSEC attribute 198
CONFDATA attribute 198
DESCRIPTION attribute 198
DTIMOUT attribute 198
DUMP attribute 199
DYNAMIC attribute 199
EXTSEC attribute 427
INDOUBT attribute 427
ISOLATE attribute 200
LOCALQ attribute 201
LOGMODECOM attribute 427
PARTITIONSET attribute 202
PRIMEDSIZE attribute 427
PRIORITY attribute 202
PROFILE attribute 202

TRANSACTION definition *(continued)*
PROGRAM attribute 203
REMOTENAME attribute 203
REMOTESYSTEM attribute 203
RESSEC attribute 204
RESTART attribute 204
ROUTABLE attribute 204
RSL attribute 427
RUNAWAY attribute 204
SHUTDOWN attribute 205
SPURGE attribute 205
STATUS attribute 205
STORAGECLEAR attribute 205
TASKDATAKEY attribute 205
TASKDATALOC attribute 206
TASKREQ attribute 206
TCLASS attribute 427
TPNAME attribute 207
TPURGE attribute 207
TRACE attribute 207
TRANCLASS attribute 193, 207
TRANSACTION attribute 208
TRANSEC attribute 427
TRPROF attribute 208
TWASIZE attribute 209
WAIT attribute 209
WAITTIME attribute 210
XTPNAME attribute 210
XTRANID attribute 211
Transaction definitions view 211
transaction list table (XLT)
coding example 425
control section 424
DFHXLT TYPE=INITIAL 424
TRANSACTION option
TRMSTAT operand 409
TRANSACTION resources 194
transaction routing
APPC devices for 180
DFHTCT macros for remote terminals 410
macros for remote terminals 410
summary 180
terminals for 176
using INDIRECT connections for 8
TRANSCEIVE option
TRMSTAT operand 409
TRANSEC attribute
TRANSACTION definition 427
TRANSID attribute
DB2ENTRY definition 33
DB2TRAN definition 38
TDQUEUE definition 164
TRANSID operand
DFHTCT TYPE=TERMINAL 409
DFHXLT TYPE=ENTRY 425
sequential devices 409
transient data
dual-purpose resource definition for 154
transient data queue definitions
data set definitions 154
installing 170
Transient data queue definitions view 172
transient data queues
disabling 171
transient data queues *(continued)*
dummy 171
replacing existing definitions 171
TRIGGERLEVEL attribute
TDQUEUE definition 164
TRMIDNT operand
DFHTCT TYPE=REMOTE 412
DFHTCT TYPE=TERMINAL 409, 411
DFHTLT TYPE=ENTRY 415
lowercase in TCT 395
remote terminals 411, 412
sequential devices 409
TRMPRTY operand
DFHTCT TYPE=TERMINAL 409
sequential devices 409
TRMSTAT operand
DFHTCT TYPE=TERMINAL 409
sequential devices 409
TRMTYPE operand
DFHTCT TYPE=LINE 407
DFHTCT TYPE=REMOTE 412
DFHTCT TYPE=TERMINAL 411
remote terminals 411, 412
sequential devices 407
TRNCLDEF object 194
TRPROF attribute
TRANSACTION definition 208
TSAGE operand
DFHTST TYPE=INITIAL 418
TSMDEF object 216
TSMODEL attribute
TSMODEL definition 215
TSMODEL definition
EXPIRYINTMIN attribute 212
LOCATION attribute 213
POOLNAME attribute 213
PREFIX attribute 214
RECOVSTATUS attribute 214
REMOTEPREFIX attribute 214
REMOTESYSTEM attribute 215
SECURITY attribute 215
TSMODEL attribute 215
XPREFIX attribute 215
XREMOTEPFX attribute 215
TSMODEL resources 211
TSQPREFIX attribute
TCPIPSERVICE definition 142
TSQUEUE attribute
DOCTEMPLATE definition 42
TST (temporary storage table)
control section 418
data sharing temporary storage 422
DFHTST TYPE=INITIAL 418
DFHTST TYPE=RECOVERY 418, 420, 421
DFHTST TYPE=SHARED 422
local temporary storage DATAIDs, DFHTST TYPE=LOCAL 420
recoverable temporary storage 418, 420, 421
recoverable temporary storage, DFHTST TYPE=RECOVERY 418
remote temporary storage DATAIDs, DFHTST TYPE=REMOTE 420
temporary storage data sharing, DFHTST TYPE=SHARED 422

TST (temporary storage table) *(continued)*
temporary storage security checking, DFHTST
TYPE=SECURITY 421
TTI attribute
TYPETERM definition 252
TWASIZE attribute
TRANSACTION definition 209
TYPE attribute
DOCTEMPLATE definition 43
JOURNALMODEL definition 79
TDQUEUE definition 164
TYPE=LOCAL, DFHTST macro
local temporary storage DATAIDs, DFHTST 420
TYPE=REMOTE, DFHTST macro
remote temporary storage DATAIDs, DFHTST 420
TYPE=SECURITY macro
temporary storage security checking, DFHTST 421
TYPE=SHARED macro
temporary storage data sharing, DFHTST 422
TYPEFILE attribute
TDQUEUE definition 165
TYPETERM
Default values of attributes 217
TYPETERM attribute
TERMINAL definition 189
TYPETERM definition 252
TYPETERM definition
ALTPAGE attribute 232
ALTSCREEN attribute 233
ALTSUFFIX attribute 233
APLKYBD attribute 234
APLTEXT attribute 234
ASCII attribute 234
ATI attribute 235
AUDIBLEALARM attribute 235
AUTOCONNECT attribute 235
AUTOPAGE attribute 236
BACKTRANS attribute 236
BRACKET attribute 236
BUILDCHAIN attribute 236
CGCSGID attribute 236
COLOR attribute 237
COPY attribute 237
CREATESESS attribute 237
DEFSCREEN attribute 237
DESCRIPTION attribute 238
DISCREQ attribute 238
DUALCASEKYBD attribute 239
ERRCOLOR attribute 239
ERRHILIGHT attribute 239
ERRINTENSIFY attribute 239
ERRLASTLINE attribute 239
EXTENDEDDS attribute 239
FMHPARM attribute 240
FORMFEED attribute 240
HILIGHT attribute 240
HORIZFORM attribute 241
IOAREALEN attribute 241
KATAKANA attribute 241
LDCLIST attribute 241
LIGHTPEN attribute 242
LOGMODE attribute 242
LOGONMSG attribute 243
MSRCONTROL attribute 243
TYPETERM definition *(continued)*
NEPCLASS attribute 243
OBFORMAT attribute 244
OBOPERID attribute 244
OUTLINE attribute 244
PAGESIZE attribute 244
PARTITIONS attribute 245
PRINTADAPTER attribute 245
PROGSYMBOLS attribute 246
QUERY attribute 246
RECEIVESIZE attribute 246
RECOVNOTIFY attribute 247
RECOVOPTION attribute 248
RELREQ attribute 249
ROUTEDMSGS attribute 249
RSTSIGNOFF attribute 249
SENDSIZE attribute 249
SESSIONTYPE attribute 250
SHIPPABLE attribute 250
SIGNOFF attribute 251
SOSI attribute 251
TERMMODEL attribute 251
TEXTKYBD attribute 251
TEXTPRINT attribute 251
TTI attribute 252
TYPETERM attribute 252
UCTRAN attribute 252
USERAREALEN attribute 253
VALIDATION attribute 253
VERTICALFORM attribute 253
Typeterm definitions view 254
TYPETERM resources 216
TYPTMDEF object 254

U

UCTRAN attribute
PROFILE definition 119
TYPETERM definition 252
UNBLOCKED option
RECFORM operand 363
UNDEFINED option
RECFORM operand 363
UNLOCK command
CEDA 313
UPDATE attribute
FILE definition 64
UPDATE option
SERVREQ operand 363
UPDATEMODEL attribute
FILE definition 64
UPGRADE command, DFHCSDUP utility program
REPLACE operand 346
USING operand 346
URI mapping definitions view 271
URIMAP attribute
URIMAP definition 268
URIMAP definition
ANALYZER attribute 258
ATOMSERVICE attribute 258
CERTIFICATE attribute 259
CHARACTERSET attribute 259
CIPHERS attribute 259
CONVERTER attribute 260

URIMAP definition *(continued)*
HFSFILE attribute 260
HOST attribute 262
HOSTCODEPAGE attribute 263
LOCATION attribute 263
MEDIATYPE attribute 263
PATH attribute 263
PIPELINE attribute 264
PORT attribute 265
PROGRAM attribute 265
REDIRECTTYPE attribute 265
SCHEME attribute 266
SOCKETCLOSE attribute 266
STATUS attribute 266
TCPIPSERVICE attribute 266
TEMPLATENAME attribute 267
TRANSACTION attribute 268
URIMAP attribute 268
USAGE attribute 268
USERID attribute 269
WEBSERVICE attribute 269
URIMAP object 271
URIMAP resources 254
URM attribute
TCPIPSERVICE definition 142
USAGE attribute
URIMAP definition 268
USEDFLTUSER attribute
CONNECTION definition 18
TERMINAL definition 190
USERAREALEN attribute
SESSIONS definition 140
TYPETERM definition 253
USERAUTH attribute
IPCONN definition 66
USERDEFINE command
CEDA 315
USERDEFINE command, DFHCSDUP utility program 347
USERID attribute
SESSIONS definition 140
TDQUEUE definition 165
TERMINAL definition 190
URIMAP definition 269
USERID operand
DFHTCT TYPE=TERMINAL 409
sequential devices 409
USERVAR of remote system 14

V

VALIDATION attribute
TYPETERM definition 253
VARIABLE option
RECFORM operand 363
VERIFY command, DFHCSDUP utility program 349
VERIFY operand
DFHFCT TYPE=FILE 364
VERTICALFORM attribute
TYPETERM definition 253
VIEW command
CEDA 319
VTAM option
ACCMETH operand 396

W

WAIT attribute
TDQUEUE definition 166
TRANSACTION definition 209
WAITACTION attribute
TDQUEUE definition 166
WAITTIME attribute
TRANSACTION definition 210
web service definitions view 274
WEBSERVICE attribute
URIMAP definition 269
WEBSERVICE resources 271
WebSphere MQ connection definitions view 99
WEBSVDEF object 274
wildcard characters as transaction IDs 38
WSDIR attribute
PIPELINE definition 110
WTOL operand
DFHCLT TYPE=WTO 356

X

XLNACTION attribute
CONNECTION definition 18
IPCONN definition 66
XLT (transaction list table)
coding example 425
control section 424
DFHXLT TYPE=INITIAL 424
XPREFIX attribute
TSMODEL definition 215
XREMOTEPFX attribute
TSMODEL definition 215
XTPNAME attribute
PARTNER definition 107
TRANSACTION definition 210
XTRANID attribute
TRANSACTION definition 211

Z

z/OS Communications Server
application name of remote system 14