

IBM CICS Transaction Server for z/VSE
Version 2 Release 2

Enhancements Guide



Note! Before using this information and the product it supports, be sure to read the general information under [“Notices” on page 483.](#)

This edition applies to Version 2 Release 2 of the IBM® licensed program CICS Transaction Server for z/VSE, program number 5655-VSE, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions.

This edition replaces SC34–2685–00.

Order publications through your IBM representative or IBM branch office serving your locality. Publications are not stocked at the address given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Deutschland Research & Development GmbH
Department 3282
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

You may also send your comments by FAX or via the Internet:

Internet: s390id@de.ibm.com
FAX (Germany): 07031-16-3456
FAX (other countries): (+49)+7031-16-3456

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

© **Copyright International Business Machines Corporation 1994, 2017.**

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

Contents

- Figures..... xi**
- Tables..... xiii**
- About this publication..... xvii**
 - How to use this publication..... xvii
 - What you need to know to understand this publication..... xvii
 - Notes on terminology..... xvii
 - Where to find more information..... xvii
- Summary of changes..... xix**
- Part 1. CICS Explorer..... 1**
 - Chapter 1. Introduction to the CICS Explorer..... 3
 - Overview of the CICS Explorer Workbench..... 3
 - Example that uses a selection of CICS Explorer views..... 6
 - Chapter 2. Supported Operations Views..... 7
 - Chapter 3. Supported Definitions Views..... 9
 - Chapter 4. Supported CICS Management Interface resources..... 11
 - Using an HTTP GET request for information on resources..... 12
 - Using an HTTP PUT request to process a single resource..... 13
 - Using HTTP requests to process resource definitions..... 14
 - Chapter 5. Restrictions when connecting to a CICS TS for z/VSE system..... 15
 - Chapter 6. Installing and configuring the CICS Explorer..... 17
 - Configuring the z/VSE host for use with the CICS Explorer..... 17
 - Obtaining a copy of the CICS Explorer..... 20
 - Performing the installation of the CICS Explorer..... 20
 - Chapter 7. Configuring a new connection to a CICS system..... 23
 - Adding a set of CICS Explorer credentials..... 23
 - Adding a CICS Explorer connection..... 24
 - Chapter 8. Using an existing connection to a CICS system..... 27
 - Changing a CICS Explorer user workspace..... 28
 - Chapter 9. Using Job EYUPARM to enter or modify debugging commands..... 29
 - Chapter 10. Diagnosing problems within the CICS Explorer..... 31
 - Debugging transactions COD0 and CODB..... 31
 - Using traces in the CICS Explorer Server part..... 32
 - How to format CPSM trace entries from an external CICS AUXTRACE data set..... 32
 - How to format CPSM trace entries from an internal CICS trace within a system dump..... 33

Chapter 11. Messages generated when using the CICS Explorer.....	35
Part 2. CICS Web support	37
Chapter 12. Configuring CICS Web support.....	39
Controlling web support with system initialization parameters.....	39
Defining resources to CICS.....	40
CICS supplied resource definitions.....	40
DOCTEMPLATE definitions.....	40
TCPIPSERVICE definitions.....	42
TRANSACTION definitions for extra alias transactions.....	42
PROGRAM definitions for user-replaceable programs.....	43
Setting up a sublibrary for the template manager.....	43
Defining a conversion table.....	43
Configuring TCP/IP for z/VSE.....	45
Reserving ports for CICS Web support.....	45
Identifying the TCP/IP server.....	45
Specifying a name server.....	46
Enabling lightpen support.....	46
Running the sample application.....	46
Using Sample Programs for Security.....	47
Chapter 13. CICS TS for z/VSE Support of HTTP/1.1.....	49
HTTP/1.1 compliance for CICS as an HTTP server	49
CICS Web support behavior in compliance with HTTP/1.1.....	50
HTTP functions not supported by CICS Web support.....	51
How CICS Web support handles chunked transfer-coding.....	51
How CICS Web support handles pipelining.....	52
How CICS Web support handles persistent connections.....	52
Using chunked transfer-coding to send an HTTP request or response.....	53
Application programming for non-HTTP requests	54
Other changes.....	55
Chapter 14. EXEC CICS DOCUMENT.....	57
DOCUMENT CREATE.....	57
DOCUMENT INSERT.....	60
DOCUMENT RETRIEVE.....	62
DOCUMENT SET.....	63
Chapter 15. EXEC CICS EXTRACT.....	67
EXTRACT TCPIP.....	67
EXTRACT CERTIFICATE.....	68
Chapter 16. EXEC CICS START.....	71
START BREXIT.....	71
Passing data to the bridge exit.....	72
Chapter 17. EXEC CICS TS.....	75
DELETEQ TS.....	75
READQ TS.....	76
WRITEQ TS.....	78
Chapter 18. EXEC CICS WEB.....	83
WEB ENDBROWSE FORMFIELD.....	83
WEB ENDBROWSE HTTPHEADER.....	83
WEB EXTRACT.....	84

WEB READ FORMFIELD.....	86
WEB READ HTTPHEADER.....	88
WEB READNEXT FORMFIELD.....	89
WEB READNEXT HTTPHEADER.....	90
WEB RECEIVE.....	91
WEB RETRIEVE.....	94
WEB SEND.....	95
WEB STARTBROWSE FORMFIELD.....	100
WEB STARTBROWSE HTTPHEADER.....	101
WEB WRITE HTTPHEADER.....	102
Chapter 19. Reference Information.....	105
HTTP header reference for CICS Web support.....	105
HTTP status code reference for CICS Web support.....	108
IANA media types and character sets.....	113
HTML coded character sets.....	114
Reference Information for DFHWBADX.....	115
Summary of parameters for analyzer programs.....	115
Part 3. Using Secure Sockets Layer (SSL).....	123
Chapter 20. Introduction to Secure Sockets Layer (SSL).....	125
Overview of SSL.....	125
SSL and the Web.....	126
Encryption and Keys.....	126
Authentication and Certificates.....	126
Client authentication.....	127
The Role of Certificate Authorities.....	127
Chapter 21. Configuring CICS to use SSL.....	129
Hardware prerequisites.....	129
Software prerequisites.....	129
Step 1: Decide Which Authorization Level You Require.....	129
Step 2: Define the System Initialization Parameters.....	130
Step 3: Define a TCPIPSERVICE Resource.....	130
Step 4: Set Up Your z/VSE System for SSL Support.....	131
Step 4A: Set Up your z/VSE System to use OpenSSL.....	131
Step 5: Configure for Server Authentication.....	131
Step 6: Configure for Client Authentication.....	132
Step 7: Configure for Client Certificate Mapping.....	132
Application programming considerations.....	132
A sample application program: DFHOWBCA.....	132
Currently-Supported SSL Cipher Suites.....	132
Part 4. Using the CICS Transaction Gateway with ECI.....	133
Chapter 22. Introduction to the ECI / CICS Transaction Gateway.....	135
How the ECI and CICS Transaction Gateway are used.....	135
How the CICS Transaction Gateway accesses CICS.....	135
The External Call Interface (ECI).....	136
Chapter 23. Configuring for ECI via TCP/IP.....	137
Hardware prerequisites.....	137
Software prerequisites.....	137
Defining a TCPIPSERVICE Resource for ECI.....	137
Displaying the ATTACHSEC Attribute.....	138

Part 5. Using CICS documents.....	139
Chapter 24. The DOCUMENT application programming interface.....	141
Creating a document.....	141
The BINARY parameter.....	141
The TEXT parameter.....	141
Inserting one document into another.....	142
Using document templates.....	142
Chapter 25. Programming with documents.....	143
Setting symbol values.....	143
Embedded template commands.....	144
Using templates in your application.....	145
The lifespan of a document.....	145
Retrieving the document without control information.....	146
Using multiple calls to construct a document.....	147
Bookmarks and inserting data.....	147
Replacing data in the document.....	148
Codepages and codepage conversion.....	149
Part 6. CICS Resource Definitions Online changes.....	151
Chapter 26. CEDA DEFINE DOCTEMPLATE.....	153
Chapter 27. CEDA DEFINE PROFILE.....	155
Chapter 28. CEDA DEFINE TCPIP SERVICE.....	161
Part 7. CICS Supplied Transactions changes.....	165
Chapter 29. CEBR—temporary storage browse.....	167
The HELP panel.....	168
Chapter 30. Execution diagnostic facility CEDF & CEDX.....	169
CEDF transaction.....	169
CEDX transaction.....	169
Command syntax.....	169
Command options.....	169
Chapter 31. CEMT INQUIRE/SET commands.....	171
CEMT INQUIRE DOCTEMPLATE.....	171
CEMT INQUIRE/SET TASK.....	173
CEMT INQUIRE/SET TCPIP.....	177
CEMT INQUIRE/SET TCPIP SERVICE.....	178
CEMT INQUIRE/SET TRANSACTION.....	181
CEMT INQUIRE/SET TSQUEUE.....	184
Part 8. CICS channels and containers.....	187
Chapter 32. Overview of CICS channels and containers.....	189
Chapter 33. CICS API commands for using channels and containers.....	191
ASSIGN.....	191
DELETE CONTAINER (CHANNEL).....	192
ENDBROWSE CONTAINER.....	192
GET CONTAINER (CHANNEL).....	193

GETNEXT CONTAINER.....	197
LINK.....	198
MOVE CONTAINER (CHANNEL).....	199
PUT CONTAINER (CHANNEL).....	201
RETURN.....	204
START CHANNEL.....	206
STARTBROWSE CONTAINER.....	209
XCTL.....	210
EXEC Interface Block.....	211
Chapter 34. How to use channels and containers.....	213
Channels: quick start.....	213
Channels and containers.....	213
Basic examples.....	214
Using channels: some typical scenarios.....	216
One channel, one program.....	216
One channel, several programs (a component).....	217
Several channels, one component.....	217
Multiple interactive components.....	218
Creating a channel.....	218
The current channel.....	219
Current channel example, with LINK commands.....	220
Current channel example, with XCTL commands.....	222
Current channel: START and RETURN commands.....	223
The scope of a channel.....	224
Scope example, with LINK commands.....	224
Scope example, with LINK and XCTL commands.....	226
Processing containers in a called sub-routine.....	227
Passing a channel to another program or task.....	227
Discovering which containers were passed to a program.....	228
Discovering which containers were returned from a link.....	228
CICS read-only containers.....	228
Channels and Security.....	229
Designing a channel: Best practices.....	229
Constructing and using a channel: an example.....	230
Dynamic transaction routing program with channels.....	231
Data conversion.....	231
Why is data conversion needed?.....	231
Preparing for code page conversion with channels.....	232
Data conversion with channels.....	232
Benefits of channels.....	236
Migrating from COMMAREAs to channels.....	236
Migrating LINK commands that pass COMMAREAs.....	237
Migrating XCTL commands that pass COMMAREAs.....	237
Migrating pseudoconversational COMMAREAs on RETURN commands.....	238
Migrating START data.....	238
Channels and containers storage requirements.....	239
Channels and containers performance.....	240
Part 9. CICS Assembler Programming.....	241
Chapter 35. Relative Addressing Instructions in Assembler Programs.....	243
DFHEIENT Macro.....	243
DFHEIRET Macro.....	244
Chapter 36. LE MAIN for Assembler.....	245
Translator Options.....	245

Defining Translator Options.....	246
Using the EXEC Interface Modules.....	246
EXAMPLE Assembler language PROGRAM using LEASM.....	247
Writing Global User Exit Programs.....	247
Writing a Task-Related User Exit Program.....	248
Language Environment Considerations for Assembler Applications.....	248
Using Language Environment Abend-handling.....	249
Part 10. Migration to CICS Transaction Server for z/VSE 2.2.....	251
Chapter 37. Migration Tasks - Overview	253
Chapter 38. System Initialization Table - DFHSIT	257
Migration Considerations.....	257
New System Initialization Parameters.....	257
Chapter 39. Resource Definition Macro Changes.....	259
Reassembling Control Tables.....	259
Chapter 40. Migration Planning for CICS System Definition Data Set (CSD).....	261
Upgrading the CSD.....	261
Chapter 41. Migrate TCPIPService Definitions.....	263
Chapter 42. Changed internal CICS control blocks	265
Chapter 43. The Global User Exit Programming Interface.....	267
Changes because of channels.....	267
Internal Control block changes.....	268
Chapter 44. Changes to User-Replaceable Programs.....	269
DFHCNV.....	269
DFHUCNV.....	269
Chapter 45. Upgrade Dynamic Transaction Routing Programs.....	271
Information Passed to the Dynamic Routing Program.....	271
Chapter 46. Monitoring and Statistics.....	273
Summary of CICS monitoring.....	273
Performance class monitoring.....	273
Exception class monitoring.....	273
Monitoring control table (MCT).....	273
The DFHMCT TYPE=RECORD Macro.....	274
Monitoring utility program.....	275
Statistical changes with CICS TS for z/VSE 2.1.....	275
Chapter 47. CICS-supplied Utility Programs	279
Changes to the trace formatting utility program, DFHTU430	279
Changes to the dump exit routine, DFHPD430.....	279
Changes to the CICS transaction dump utility program, DFHDU430.....	279
Changes to other internally used programs.....	279
Chapter 48. How to migrate RPG II online applications to CICS TS for z/VSE.....	281
Chapter 49. Changes to the EXEC CICS application programming interface.....	283
Changes to INQUIRE SYSTEM command.....	283
CONVERTTIME.....	283

FORMATTIME.....	285
WAIT JOURNALNUM.....	288

Part 11. CICS System Initialization Changes.....291

Chapter 50. CICS system initialization parameters.....	293
System Initialization Parameters - New with CICS TS for z/VSE 2.2.....	293
System Initialization Parameters - New with CICS TS for z/VSE 2.1.....	293
Specifying system initialization parameters.....	293
Migration considerations.....	294
The DFHSIT macro parameters.....	294
Creating a system initialization table.....	300
Coding more than one system initialization table.....	300
CICS-supplied system initialization tables.....	300
System initialization tables supplied with z/VSE.....	301
Assembling the system initialization table.....	301
Parameters that you cannot code in the DFHSIT macro.....	301
Overriding SIT parameters at system startup.....	302
The CICS parameter manager domain.....	302
System initialization control keywords.....	303
Processing the PARM parameter.....	305
Processing the SYSIPT data set.....	306
Processing the console entries.....	307
Notes on CICS resource table and module keywords.....	307
Selecting versions of CICS programs and tables.....	309
Using a suffix to select the dynamic backout program.....	309
Using an explicit level of function to select programs.....	309
Excluding unwanted programs.....	309
Classes of start and restart.....	310
The global catalog.....	311
The local catalog.....	311
The START system initialization parameter.....	312
CICS startup and the VTAM session.....	315
End of CICS startup.....	315
The system initialization parameter descriptions.....	316

Part 12. Trace entries.....373

Chapter 51. Trace entries.....	375
IRC Commands AP trace points	375
Dispatcher domain trace points.....	375
File control trace points.....	375
Document handler AP trace points.....	376
Socket domain.....	376
Web domain trace points.....	394
Bridge facility management.....	397
Bridge facility management 2.....	401
AP domain recovery trace points.....	403
AP domain transaction initiation trace points.....	403
AP domain bridge facility trace points.....	403
AP domain conversion trace points.....	404
AP domain container data transformation trace points.....	404
Program manager domain trace points.....	408
AP domain event manager trace points.....	411
CMCI domain trace points.....	411
Timer domain trace points.....	478

Miscellaneous Updates.....	479
DFHCNV Macro.....	479
Resource and command check cross-reference.....	479
CICS-supplied transactions.....	479
REXX for CICS TS for z/VSE (REXX/CICS).....	481
Notices.....	483
Programming Interface Information.....	484
Trademarks.....	484
Terms and Conditions for Product Documentation.....	484
Accessibility.....	487
Using Assistive Technologies.....	487
Documentation Format.....	487
Index.....	489

Figures

1. CICS Explorer Workbench.....	4
2. CICS Explorer Workbench illustrating display of a file's attributes.....	5
3. Defining a TCPIP SERVICE for the CICS Explorer.....	19
4. Host Connections window.....	23
5. New Credentials window.....	24
6. Add CMCI Connection window.....	25
7. Selecting a Host Connection to a CICS Transaction Server for z/VSE system.....	27
8. Example of a TCPIP SERVICE resource definition for SSL.....	131
9. How the CICS Transaction Gateway and ECI are used.....	135
10. Example of a TCPIP SERVICE resource definition for ECI.....	138
11. The CEMT INQ TCPIP SERV display.....	138
12. The DEFINE panel for DOCTEMPLATE.....	153
13. The DEFINE panel for PROFILE.....	155
14. The DEFINE panel for the TCPIP SERVICE resource definition.....	161
15. Typical CEBR screen displaying temporary storage queue contents.....	167
16. Typical CEBR display of default temporary storage queue.....	168
17. Sample of the screen following either the INQUIRE or the SET command.....	171
18. CEMT INQUIRE DOCTEMPLATE screen.....	172
19. The expanded display of an individual entry.....	172
20. CEMT INQUIRE TASK screen.....	175
21. The expanded display of an individual entry.....	175
22. CEMT INQUIRE TRANSACTION screen.....	182
23. The expanded display of an individual entry.....	182

24. CEMT INQUIRE TSQUEUE screen.....	185
25. The expanded display of an individual entry.....	185
26. A simple example of a program that creates a channel and passes it to a second program.....	215
27. A simple example of a linked to program that retrieves data from the channel it has been passed...	216
28. A stand-alone program with a single channel.....	217
29. A component—a set of related programs invoked through a single external channel.....	217
30. Multiple external channels to the same component.....	218
31. Multiple components which interact through their channels.....	218
32. Current channel: example with LINK commands.....	221
33. Current channels—example, with XCTL commands.....	223
34. The scope of a channel—example showing LINK commands.....	225
35. The scope of a channel—example showing LINK and XCTL commands.....	226
36. How a client program can construct a channel, pass it to a server program, and retrieve the server's output.....	230
37. How a server program can query the channel it's been passed, retrieve data from the channel's containers, and return output to the caller.....	231
38. Statistic report example.....	277
39. Example of a PARM parameter coded over two lines.....	306

Tables

1. CICS Management Resources supported for CICS TS for z/VSE systems.....	11
2. Configuring CICS Web support.....	39
3. CICS actions for headers on an HTTP request.....	106
4. CICS-written headers for an HTTP response.....	108
5. Status Codes for CICS-generated responses sent to Web clients.....	109
6. Status codes for user-written responses sent to Web clients.....	110
7. Coded character sets.....	114
8. Names of parameter and constants for analyzer programs as defined in files.....	116
9. Abbreviated parameter names.....	116
10. Returned values in wbra_response.....	120
11. The effect of UCTRAN attributes on tranid and data translation.....	160
12. The current channels of interactive programs—example with LINK commands.....	222
13. The current channels of interactive programs—example.....	223
14. The scope of a channel—example with LINK commands.....	225
15. The scope of a channel—example with LINK and XCTL commands.....	227
16. Migrating LINK commands that pass COMMAREAs.....	237
17. Migrating XCTL commands that pass COMMAREAs.....	238
18. Migrating pseudoconversational COMMAREAs on RETURN commands.....	238
19. Migrating START data.....	239
20. Interface Modules.....	246
21. Internal CICS control blocks that were changed for CICS TS for z/VSE 2.1.....	265
22. New fields in the DSECT.....	267

23. New values on INCLUDE and EXCLUDE operands of the DFHMCT TYPE=RECORD introduced with CICS TS for z/VSE 2.1.....	274
24. New fields in A14 record.....	275
25. New System Initialization Parameter for CICS TS for z/VSE 2.2.....	293
26. New System Initialization Parameters.....	293
27. The DFHSIT macro parameters.....	294
28. Summary of resources with a suffix, a dummy load module, or a COLD option.....	307
29. SIT control tables with a NO option.....	310
30. Effect of the START= parameter in conjunction with the catalogs.....	313
31. Effect of the START= parameter on the CICS domains at initialization.....	314
32. Versions of BMS.....	319
33. Levels of protection provided by CICS validation of application-supplied addresses.....	321
34. Effect of CONFDATA system initialization and transaction definition parameters.....	323
35. Results of ESM authorization requests (with SEC=YES).....	351
36. CICS component names and abbreviations.....	353
37. IRC Commands.....	375
38. Dispatcher domain trace points.....	375
39. File control trace points.....	375
40. Document handler.....	376
41. Socket domain trace points.....	376
42. Web domain trace points.....	394
43. Bridge facility management trace points.....	397
44. Bridge facility management trace points.....	401
45. AP domain recovery trace points.....	403
46. AP domain transaction initiation trace points.....	403
47. AP domain bridge facility trace points.....	403

48. AP domain conversion trace points.....	404
49. AP domain container data transformation trace points.....	404
50. Program manager domain trace entries.....	408
51. Event manager trace points.....	411
52. CMCI domain trace points.....	411
53. Timer domain trace points.....	478

About this publication

CICS Transaction Server for z/VSE 2.2 (CICS TS for z/VSE 2.2) replaces CICS Transaction Server for z/VSE 2.1 (CICS TS for z/VSE 2.1) and CICS Transaction Server for VSE/ESA 1.1.1 (CICS TS for VSE/ESA 1.1.1). The functionality of CICS TS for VSE/ESA 1.1.1 and CICS TS for z/VSE 2.1 is contained in CICS TS for z/VSE 2.2. CICS TS for z/VSE 2.2 is the only CICS version that can be used with z/VSE 6.2.

CICS TS for z/VSE 2.2 includes the following components:

- CICS Report Controller
- REXX for CICS

This publication describes:

- Enhancements to the IBM CICS Explorer system management tool.
- Upgrade of CICS Web support to HTTP/1.1.
- Enhancements to the CICS Application Programming Interface (API).

It also addresses migration planning from CICS TS for VSE/ESA 1.1.1 to CICS TS for z/VSE 2.1 or CICS TS for z/VSE 2.2 or from CICS TS for z/VSE 2.1 to CICS TS for z/VSE 2.2. It contains all the information of the SC34-2685-00 edition of this publication.

How to use this publication

Use this publication together with the CICS Transaction Server for VSE/ESA 1.1.1 publications.

What you need to know to understand this publication

This publication assumes that you are familiar with CICS®, either as a system administrator or as a system or application programmer. Some parts of the publication assume additional knowledge about CICS and other products.

Notes on terminology

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

In this publication, the CICS Web interface is split into the Listener support for TCPIP SERVICE, and the protocol support for HTTP. This publication now refers to the HTTP protocol support as "**CICS Web support**". Within the product code, the term "**CICS Web interface**" remains synonymous with "**CICS Web support**".

There are two ways of coding Web application programs:

1. **Commarea**-style applications are those that take a communication area containing an HTTP request as input, and build an HTTP response in the communication area.
2. **Web API** applications use the new WEB and DOCUMENT application programming interface to process the inbound HTTP request and build the response.

Where to find more information

You may occasionally need the following IBM publications:

Publication title	Order number
z/VSE Administration	SC34-2692

Publication title	Order number
z/VSE Diagnosis Tools	SC34-2628
z/VSE Guide for Solving Problems	SC34-2605
z/VSE Guide to System Functions	SC34-2705
z/VSE Installation	SC34-2678
z/VSE Licensed Program Specifications	GC33-8354
z/VSE Messages and Codes Volume 1	SC34-2682
z/VSE Messages and Codes Volume 2	SC34-2683
z/VSE Messages and Codes Volume 3	SC34-2684
z/VSE SNA Networking Support	SC34-2626
z/VSE Operation	SC33-8309
z/VSE Planning	SC34-2681
z/VSE System Control Statements	SC34-2679
z/VSE System Macros Reference	SC34-2708
z/VSE System Macros User's Guide	SC34-2709
z/VSE System Upgrade and Service	SC34-2680
z/VSE System Utilities	SC34-2675
z/VSE TCP/IP Support	SC34-2706
z/VSE e-business Connectors User's Guide	SC34-2693

z/VSE IBM Documentation

IBM Documentation is the new home for IBM's technical information. The z/VSE IBM Documentation can be found here:

<https://www.ibm.com/docs/en/zvse/6.2>

You can also find VSE user examples (in zipped format) at

https://public.dhe.ibm.com/eserver/zseries/zos/vse/pdf3/zVSE_Samples.pdf

Summary of changes

CICS TS for z/VSE V2.2

The current edition of this publication, relating to Version 2 Release 2 Modification Level 0 of the *CICS Transaction Server for z/VSE*, contains these changes:

- CICS TS for z/VSE 2.2 now includes support to enable coding of completely Language Environment (LE) technology-enabled application programs in assembler. A new translator option, LEASM, is provided, which causes Language Environment function to be used to set up the program's environment. This option eases integration of these applications into Language Environment, so that Language Environment services can run more easily. This features was ported from CICS TS for z/OS 3.1. Refer to [Chapter 36, “LE MAIN for Assembler,”](#) on page 245.
- New operands are added to the DFHEIENT and DFHEIRET macros and a new DFHKEBRC macro is added to provide support to use relative addressing instructions in an assembler program. When relative addressing is used, no base registers are needed to address the program instructions, but at least one base register is needed to address static data within the program. The new operand value CODEREG=0 and the new operands STATREG and STATIC are added to the DFHEIENT macro. The new operand LITERALS is added to the DFHEIRET macro. The new DFHKEBRC macro provides relative branch OPSYN definitions for assembler programs. These features were ported from CICS TS for z/OS 4.1 and 5.1. Refer to [Chapter 35, “Relative Addressing Instructions in Assembler Programs,”](#) on page 243.
- Support for the CICS Explorer Client was introduced with CICS TS for VSE/ESA 1.1 (z/VSE 5.1). With this initial support it was possible to *monitor* (read-only) CICS Resources on CICS TS for VSE/ESA. CICS TS for z/VSE 2.1 (shipped with z/VSE 6.1) extended this support and allowed to *control* existing CICS resources and *update* selected resource attributes. The enhancements with CICS TS for z/VSE 2.2 will now allow to define new resources and change or delete existing CICS resource definitions. From the CICS Explorer Client point of view this means that the **Definitions** view can now be used for selected resources. Refer to [Chapter 3, “Supported Definitions Views,”](#) on page 9.
- **The CICS Explorer Operations** view now supports the two new items: **Dynamic Storage Areas** and **Global TS Queue Statistics**. Refer to [Chapter 2, “Supported Operations Views,”](#) on page 7.
- New section describes how to diagnose and fix problems within the CICS Explorer. Refer to [Chapter 10, “Diagnosing problems within the CICS Explorer,”](#) on page 31.
- New APPEND option on the PUT CONTAINER command. Refer to [“PUT CONTAINER \(CHANNEL\)”](#) on page 201.
- New BYTEOFFSET option on the GET CONTAINER command. Refer to [“GET CONTAINER \(CHANNEL\)”](#) on page 193.
- New options DATESTRING, STRINGFORMAT, and RFC1123 of the FORMATTIME command were added along with the whole description of this command. Refer to [“FORMATTIME”](#) on page 285.
- CONVERTIME command was added. Refer to [“CONVERTTIME”](#) on page 283.
- New NOTAUTH condition on the WAIT JOURNALNUM command. Refer to [“WAIT JOURNALNUM”](#) on page 288.
- CICS TS for z/VSE V2.2 now includes support for code page conversion that involve Unicode (UTF-8, UTF-16) data.
- New Web attach transaction CWXU. Refer to [“TRANSACTION definitions for extra alias transactions”](#) on page 42.
- Using Sample Programs for Security. Refer to [“Using Sample Programs for Security”](#) on page 47.
- CICS TS for z/VSE Support of HTTP/1.1. Refer to [Chapter 13, “CICS TS for z/VSE Support of HTTP/1.1,”](#) on page 49.
- New HOST, HOSTLENGTH, SCHEME, PORTNUMBER and VERSIONLEN options on the WEB EXTRACT command. Refer to [“WEB EXTRACT”](#) on page 84.

- New SET option on the WEB READ FORMFIELD command. Refer to [“WEB READ FORMFIELD”](#) on page 86.
- New CHARACTERSET and SERVERCONV options on the WEB RECEIVE command. Refer to [“WEB RECEIVE”](#) on page 91.
- New ACTION, CHARACTERSET, CHUNKING, CLOSESTATUS, FROM, FROMLENGTH, HOSTCODEPAGE, MEDIATYPE, SERVERCONV and STATUSLEN options on the WEB SEND command. Refer to [“WEB SEND”](#) on page 95.
- New topic HTTP header reference for CICS Web support. Refer to [“HTTP header reference for CICS Web support”](#) on page 105.
- New topic HTTP status code reference for CICS Web support. Refer to [“HTTP status code reference for CICS Web support”](#) on page 108.
- New topic IANA media types and character sets. Refer to [“IANA media types and character sets”](#) on page 113.
- New topic HTML coded character sets. Refer to [“HTML coded character sets”](#) on page 114.

CICS TS for z/VSE V2.1 updates are:

- The IBM *CICS Explorer* system management tool. See [Part 1, “CICS Explorer,”](#) on page 1.
- Information about using the CICS *channels* and *containers*. See [Part 8, “CICS channels and containers,”](#) on page 187.
- Migration from CICS TS for VSE/ESA to CICS TS for z/VSE®. See [Part 10, “Migration to CICS Transaction Server for z/VSE 2.2,”](#) on page 251.

Note:

1. This publication includes all of the information that was previously contained in the *CICS Transaction Server for VSE/ESA Version 1 Release 1 Modification Level 1, Enhancements Guide*, GC34–5763–07.
2. With the exception of the *CICS TS Enhancements Guide*, no CICS TS publications are currently updated. Updates or clarifications to information that is contained in CICS TS publications, which are not updated, is collected in [“Miscellaneous Updates”](#) on page 479.

Part 1. CICS Explorer

This part of the publication describes how you can use the CICS Explorer in a z/VSE environment.

Chapter 1. Introduction to the CICS Explorer

The *CICS Explorer* is a *system management tool* for use by system administrators and operators. It is designed to provide a simple, easy-to-use way of managing CICS Transaction Server (CICS TS) systems.

The CICS TS can run in a partition of z/VSE or a region of z/OS.

The CICS Explorer is based on the Eclipse *Rich Client Platform* (RCP), which is an application that uses the windowing and GUI features of the operating system (for example, drag and drop) and is integrated with the operating system's component model. The same CICS Explorer is used for accessing CICS systems running under both z/VSE and z/OS.

The CICS Explorer interacts with its server-part running on z/VSE or z/OS. The *CICS Explorer server-part* on z/VSE consists of:

- A *client interface* which manages incoming requests to the CICS TS, and outgoing responses from the CICS TS.
- A *system interface* that executes the incoming requests.

The main advantages of using the CICS Explorer are:

- You can manage your CICS systems, perform tasks, and present information in an integrated and common way instead of (as previously) having to use various stand-alone graphical and non-graphical user interfaces.
- You can use your own plug-ins and future CICS tools, providing they have been integrated into the Eclipse RCP.

When using the CICS Explorer, the general program flow is as follows:

1. The CICS Explorer uses the *CICS Web Interface* (HTTP format) to connect to a CICS TS and send a request.
2. The CICS TS receives the CICS Explorer request and converts this (HTTP) request into an EXEC CICS command or a Control Information Query.
3. The results from executing the EXEC CICS command or Control Information Query are converted back into HTTP format and are returned to the CICS Explorer.
4. The *CICS Explorer Workbench* displays the results in the *views* (shown in [Figure 1 on page 4](#)). In doing so, the CICS Explorer recognizes whether the connected CICS TS is running under z/VSE or z/OS, and adjusts the views accordingly.

You might also wish to view the information about the CICS Explorer on YouTube:

<http://www.youtube.com/user/CICSExplorer>

Overview of the CICS Explorer Workbench

[Figure 1 on page 4](#) shows the *menu bar*, *view toolbar*, *perspective switcher*, *online help*, *fast view toolbar*, and other details of the CICS Explorer Workbench.

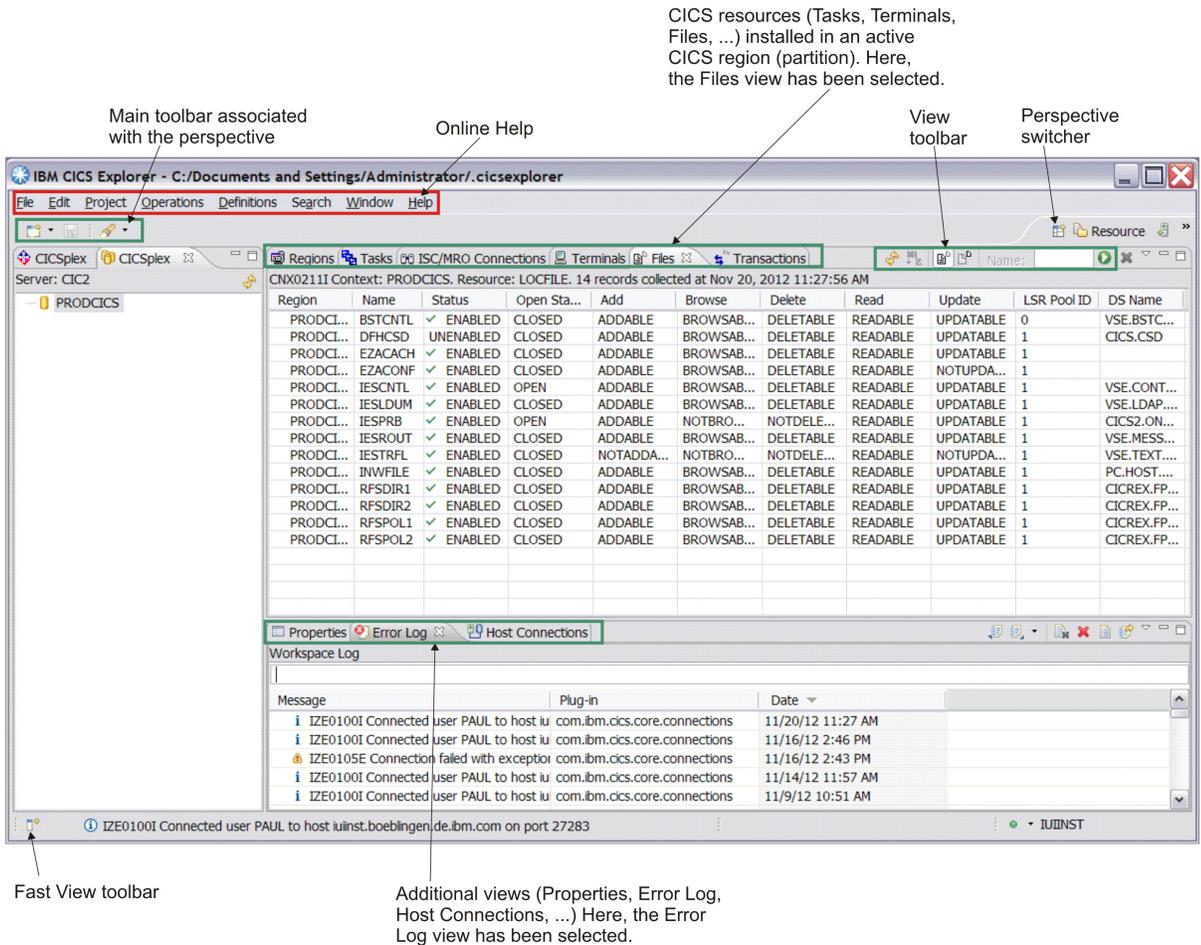


Figure 1. CICS Explorer Workbench

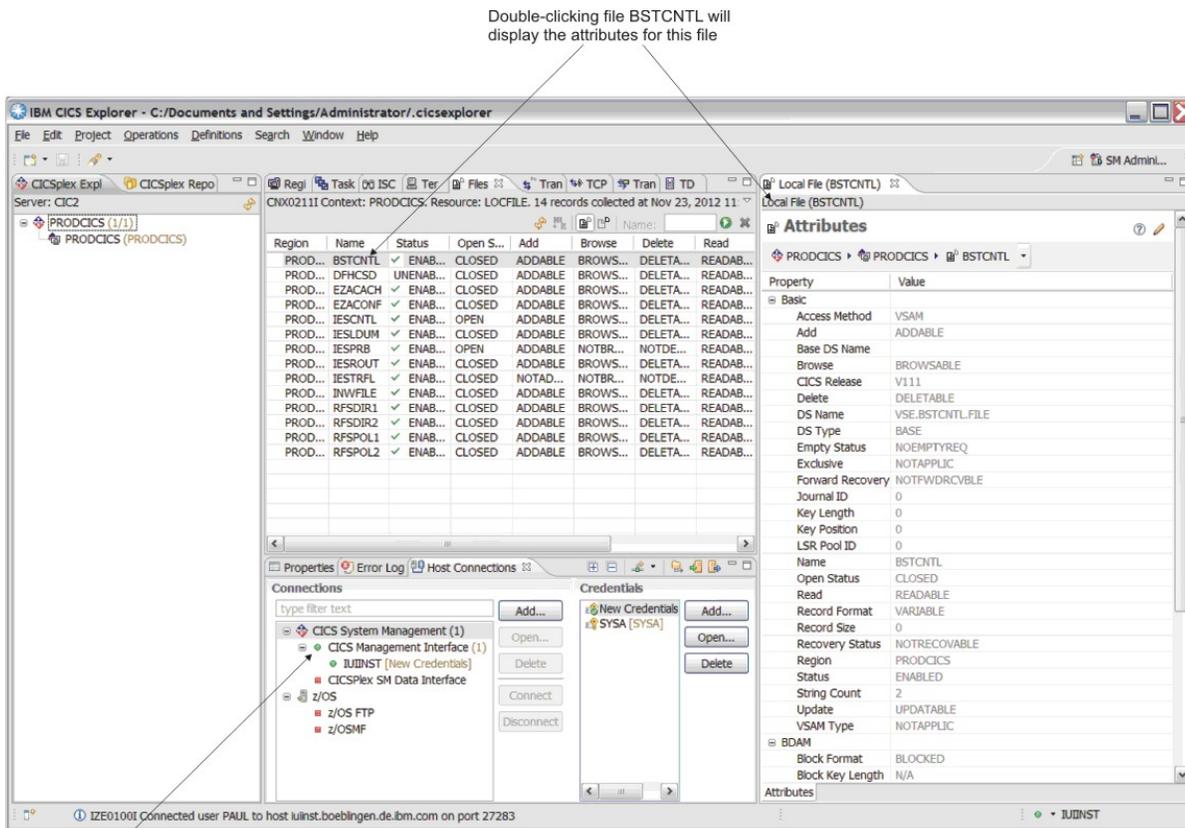
An Eclipse *perspective* is a set of related tools grouped together for a specific task or role. For example, the perspective used by a CICS Systems Programmer is different from that used by a Java™ Application Developer.

- Figure 1 on page 4 shows one perspective (SM Administration) used with the CICS Explorer.
- If you click **Window** → **Open Perspective** → **Other** you can see the list of perspectives supplied with this version of the CICS Explorer.

To get fast access to your commonly-used views, you can add them to the *fast view toolbar* shown in Figure 1 on page 4. To do so, right-click the View tab and select **Fast View**.

If you click **Operations** and then select a resource from the pull-down menu that is supported by the CICS Explorer (for example, Local Files), a resource view will then be created. In the example shown in Figure 1 on page 4, the Local Files view is created with a list of all local files defined to this CICS TS system.

If (for example) file BSTCNTL is double-clicked, the *attributes* for this file are displayed, as shown in Figure 2 on page 5.



When connecting to a CICS TS for z/VSE system, only the CICS Management Interface is supported - the CICSplex SM Data Interface is not supported

Figure 2. CICS Explorer Workbench illustrating display of a file's attributes

In Figure 2 on page 5, we see that the attributes are displayed in tabular form and logically grouped together:

- Every attribute benefits from field-level verification, where the entry is validated in real time.
- Errors are identified by the Error icon, which identifies the field in error and the page on which the field appears.

If an entry in a resource view is right-clicked, the actions that *are possible* are displayed in bold font. For example, in Figure 2 on page 5 if file BSTCNTL is selected and right-clicked, a window is displayed in which:

- **Open, Open File, Close File, Discard, Enable, Disable, Copy,** and **Create System Event** functions can be performed and are shown in bold font.
- Any functions that cannot be performed are shown in "pale" font.

For further information about the CICS Explorer Workbench, such as how to:

- Reposition columns
- Resize columns
- Customize filters
- Create perspectives

refer to the online help provided with the CICS Explorer (obtained by clicking **Help** in Figure 1 on page 4).

Example that uses a selection of CICS Explorer views

This example uses a selection of CICS Explorer views. It illustrates how you can switch between views to obtain useful information that would otherwise require having to use various CICS transactions and graphical user interfaces.

To display information about CICS resources you might use:

- The *Program view* to display the attributes of selected programs. If you were to use the equivalent CICS transaction `CECI INQUIRE PROG(program-name)`, the information would be distributed over a number of pages.
- The *Regions view* (Partitions view) to show more dynamic information, such as maximum tasks allowed and number of times the maximum number of tasks was reached. This type of information, together with the "short on storage" indicators, provide valuable performance indicators.
- The *Task view* to display detailed information about tasks in the relevant CICS partition.
- The *Queues view* (TS queue and TD queue) to determine how the temporary storage and transient data queues were used.

Chapter 2. Supported Operations Views

This topic describes the **Operations** menu item *views* that are supported when you use the CICS Explorer to connect to a *CICS TS for z/VSE* system.

If you select a resource from the **Operations** view, a resource view will be opened that includes a list of all corresponding *resource definitions*. If you right-click an entry in this list, an *action list* for this entry will be displayed:

- Actions that can be performed are shown in **bold** font.
- Actions that cannot be performed are shown in "pale" font.

These are the operations views that are currently supported for a CICS TS for z/VSE system:

Document Templates

The Document Templates (DOCTEMP) view can be used to display/process information about document templates installed in the active CICS system.

Files

The local Files (LOCFILE) and remote Files (REMFIL) view can be used to display/process information about local and remote CICS files in the current context and scope.

Interval Control Requests

The Interval Control Requests (REQID) view can be used to display/process information about outstanding interval control requests in active CICS systems.

ISC/MRO Connections

The Inter-Systems Communication (ISC) / Multi-Region Operation (MRO) Connections (CONNECT) view can be used to display/process information about ISC over IBM Systems Network Architecture (SNA) connections and MRO connections.

Programs

The Programs (PROGRAM) view can be used to display/process information about currently installed programs.

Regions

The Regions (CICSRGN) view can be used to display/process information about the active CICS systems in the current context and scope.

Tasks

The Tasks (TASK) view can be used to display/process information about tasks that are running in the current context and scope.

TCP/IP Services

The TCP/IP Services (TCPIPS) view can be used to display/process information about the TCP/IP Services in an active CICS system.

TD Queues

- The extrapartition Transient Data Queues (EXTRATDQ) view can be used to display/process information about currently installed extrapartition transient data queues.
- The indirect Transient Data Queues (INDTDQ) view can be used to display/process information about currently installed indirect transient data queues.
- The intrapartition Transient Data Queues (INTRATDQ) view can be used to display/process information about currently installed intrapartition transient data queues.
- The remote Transient Data Queues (REMTDQ) view can be used to display/process information about currently installed remote transient data queues.

Terminals

The Terminals (TERMNL) view can be used to display/process information about tasks that are running in the current context and scope.

Transactions

The local Transactions (LOCTRAN) view can be used to display/process information about CICS and user-defined local transactions in the current context and scope. The remote Transactions (REMTRAN) view can be used to display/process information about CICS and user-defined remote transactions in the current context and scope.

Transaction Classes

The Transaction Classes (TRANCLAS) view can be used to display/process information about the transaction classes for each CICS system.

TS Queues

The TS Queues (TSQNAME) view can be used to display information about temporary storage queues in an active CICS system.

Dynamic Storage Areas

The Dynamic Storage Areas (CICSDSA) view displays information about dynamic storage areas (DSAs) within the CICS system.

Global TS Queue Statistics

The TS Queue Statistics (TSQGBL) view displays information about temporary storage queue usage.

Related topic: [Chapter 4, “Supported CICS Management Interface resources,” on page 11.](#)

Chapter 3. Supported Definitions Views

This topic describes the **Definitions** menu item *views* that are supported when you use the CICS Explorer to connect to a *CICS TS for z/VSE* system.

Note: Note that the IBM CICS Explorer Version 5.4 (or higher) is required to support CICS Explorer Definitions with CICS TS for z/VSE.

The CICS Explorer allows to define, change and delete CICS Resources. In order to do this, the CICS Explorer **Definitions** pull-down menu has to be opened. Following CICS resources will be supported for **Definition** by CICS TS for z/VSE:

ISC/MRO Connection

The ISC/MRO Connection Definitions (CONNDEF) view displays remote systems that a CICS system communicates with using intersystem communication (ISC) or multiple region operation (MRO).

Document Template

The Document Template Definitions (DOCDEF) view displays information about document template definitions for use in the CICS system.

File

The File Definitions (FILEDEF) view displays information about the physical and operational characteristics of file definitions.

LSR Pool

The LSR Pool Definitions (LSRDEF) view displays information about the size and characteristics of local shared resource pool definitions that VSAM uses for certain files.

Map Set

The Map Set Definitions (MAPDEF) view displays information about the characteristics of a group of related screen layouts, or map definitions.

Partner

The Partner Definitions (PARTDEF) view displays information about the physical and operational characteristics of partner definitions. Partner definitions enable CICS application programs to communicate via APPC protocols with a partner application program running on a remote logical unit.

Partition Set

The Partition Set Definitions (PRTNDEF) view displays information about the characteristics of display partition configuration definitions.

Profile

The Profile Definitions (PROFDEF) view displays information about the interactions between transactions and terminals or logical units.

Program

The Program Definitions (PROGDEF) view displays information about the control information for a program that is stored in the program library and used to process a transaction.

Session

The Session Definitions (SESSDEF) view displays information about the logical links between systems that communicate using intersystem communication (ISC) or multiple region operation (MRO).

TCP/IP Service

The TCP/IP Service Definitions (TCPDEF) view displays information about the TCP/IP service definitions that use internal sockets support.

Terminal

The Terminal Definitions (TERMDEF) view displays information about the unique characteristics of the terminal device definitions (including visual display units, printers, and operating system consoles) with which CICS communicates.

Transaction Class

The Transaction Class Definitions (TRNCLDEF) view displays information about the way that transactions are to run in CICS systems.

Transaction

The Transaction Definitions (TRANDEF) view displays information about how transactions are to run in CICS systems.

Typeterm

The Typeterm Definitions (TYPTMDEF) view displays information about sets of common attributes for a group of terminals.

Resource Group

The Resource group definitions view displays the CICS groups in the CSD (CSDGROUP).

Group List

The Group List definition view displays the CICS group lists in the CSD (CSDLIST).

Note:

1. With Partition Set, Map Set and Program Definitions the attribute USELPACOPY is used as a synonym for USESVACOPY.
2. The IBM CICS Explorer, when connected to CICS TS for z/VSE does not support
 - the PROTECT attribute for PROFILE definitions
 - the INDOUBT attribute for TRANSACTION definitions
 - the HFSFILE attribute for DOCUMENT TEMPLATE definitions.
3. With a huge number of resource definitions it may happen that a CICS Explorer request to show them fails with message 'EYUXC0020E Cache request exceeds extension size for DAT cache.' If this happens increase the size of the internal cache. This size is defined by the EYUPARM system parameter CACHECMXTND. The default value for this parameter is 768 which, for example, allows for about 4,000 CICS Program Definitions to be retrieved.
Refer to Chapter 9, [“Using Job EYUPARM to enter or modify debugging commands,”](#) on page 29 on how to change EYUPARM parameters.

Chapter 4. Supported CICS Management Interface resources

This topic describes the CICS Management Interface resources that are supported when you use the CICS Explorer (or equivalent HTTP requests) to connect to a *CICS TS for z/VSE* system.

The CICS Explorer uses the HTTP protocol to request resource information from a CICS TS for z/VSE system, and to then obtain the results of the request. You can also use the HTTP protocol to develop HTTP client applications that monitor and control CICS Management Interface resources.

Table 1. CICS Management Resources supported for CICS TS for z/VSE systems

External Resource Name	Internal Resource Name
CICSRegion	APPLID
CICSDocumentTemplate	NAME
CICSLocalTransaction	TRANID
CICSRemoteTransaction	TRANID
CICSTransactionClass	NAME
CICSProgram	PROGRAM
CICSTask	TASK
CICSLocalFile	FILE
CICSRemoteFile	FILE
CICSIJSCMROConnection	NAME
CICSTCPIPService	NAME
CICSTerminal	TERMID
CICSIntervalControlRequest	NAME
CICSExtrapartitionTDQueue	TDQUEUE
CICSIntrapartitionTDQueue	TDQUEUE
CICSRemoteTDQueue	TDQUEUE
CICSIndirectTDQueue	TDQUEUE
CICSTSQueue	NAME
CICSDynamicStorageArea	CICSDSA
CICSGlobalTSQueueStatistics	TSQGBL
CICSDefinitionISCMROConnection	CONNDEF
CICSCSDGroup	CSDGROUP
CICSCSDResource	CSDINGRP
CICSCSDGroupInList	CSDINLST
CICSCSDList	CSDLIST
CICSDefinitionDocumentTemplate	DOCDEF

Table 1. CICS Management Resources supported for CICS TS for z/VSE systems (continued)	
External Resource Name	Internal Resource Name
CICSDefinitionFile	FILEDEF
CICSDefinitionLSRPool	LSRDEF
CICSDefinitionMapSet	MAPDEF
CICSDefinitionPartner	PARTDEF
CICSDefinitionProfile	PROFDEF
CICSDefinitionProgram	PROGDEF
CICSDefinitionPartitionSet	PRTNDEF
CICSDefinitionSession	SESSDEF
CICSDefinitionTCPIPService	TCPDEF
CICSDefinitionTerminal	TERMDEF
CICSDefinitionTransaction	TRANDEF
CICSDefinitionTransactionClass	TRNCLDEF
CICSDefinitionTypeterm	TYPTMDEF

Using an HTTP GET request for information on resources

To request the *resource information*, you can use the following HTTP GET request:

```
get http://hostname:port/CICSSystemManagement/resource/regionname
```

To request information about a *selected resource item*, you can use the following HTTP GET request:

```
get http://hostname:port/CICSSystemManagement/resource/regionname/regionname?CRITERIA=((keyname=='itemname'))
```

For example, to display information about program IESNEWS, you would use:

- keyname = PROGRAM
- itemname = IESNEWS

Option Description

hostname

The TCP/IP host name of your stand-alone CICS region (partition).

port

The port used to access the server. This is the value specified in the PORTNUMBER attribute of the TCPIPService definition created when configuring the CICS management client interface.

resource name

The external resource name as described in the list of resource names (above).

region name

The name of the CICS region (partition) from where you want to retrieve the resource information.

To retrieve (E)DSA information, you can use an HTTP GET request with the following URL:

```
http://hostname:port/CICSSystemManagement/CICSDynamicStorageArea/cicsname/cicsname?CRITERIA=((NAME=='dsaname'))
```

where *dsaname* can be CDSA, RDSA, SDSA, UDSA, CDSA, ERDSA, ESDSA or EUDSA.

To retrieve *TSQGBL* information, you can use an HTTP GET request with the following URL:

```
http://hostname:port/CICSSystemManagement/CICSGlobalTSQueueStatistics/  
cicsname?
```

Related topic: [Chapter 2, “Supported Operations Views,” on page 7.](#)

Using an HTTP PUT request to process a single resource

To update or perform an action on a CICS Explorer resource using the HTTP protocol, an HTTP PUT request is required. PUT requests include an XML *request body* which contains either:

- Details of the changes to be made to resource attributes.
- The action to be performed on the targeted resources.

The syntax of an HTTP PUT request is:

```
put http://hostname:port/CICSSystemManagement/ResourceName/regionname/  
regionname?CRITERIA=((keyname=name))
```

To perform an action on a resource, the request body is:

```
<request>  
<action name='action' />  
</request>
```

Here is an example on how to open a File:

```
http://hostname:port/CICSSystemManagement/  
CICSLocalFile/PRODCICS/PRODCICS?CRITERIA=((FILE=='FILE001'))
```

The request body for the above example is:

```
<request>  
<action name="OPEN" />  
</request>
```

Some action requests require an additional parameter. For example, here a FILE is closed:

```
http://hostname:port/CICSSystemManagement/  
CICSLocalFile/PRODCICS/PRODCICS?CRITERIA=((FILE=='FILE001'))
```

The request body for the above example is:

```
<request>  
<action name='CLOSE' >  
<parameter name='BUSY' value='NOWAIT' />  
</action>  
</request>
```

To update one or more attributes, the request body is:

```
<request>  
<update>  
<attributes attribute_name="new_value" [attribute_name = "new_value" ...] />  
</update>  
</request>
```

Here is an example that updates value of MAXTASKS value to 60:

```
put http://hostname:port/CICSSystemManagement/CICSRegion/PRODCICS/  
PRODCICS?CRITERIA=((APPLID=='PRODCICS'))
```

The request body for this example is:

```
<REQUEST>
  <UPDATE>
    <ATTRIBUTES MAXTASKS="60" />
  </UPDATE>
</REQUEST>
```

Using HTTP requests to process resource definitions

To retrieve information on defined *CICS Resources*, use an HTTP GET request with the following URL:

```
http://hostname:port/CICSSystemManagement/ResourceName/cicsname/cicsname?
CRITERIA=((NAME=='progrname'))&PARAMETER=CSDGROUP(group)
```

To define a *CICS Resource*, use an HTTP POST request with URL:

```
http://hostname:port/CICSSystemManagement/CICSDefinitionProgram/cicsname/cicsname/
```

and XML body example:

```
<request>
  <create>
    <parameter name="CSD"/>
    <attributes name="PROGT1" csdgroup="TEST"/>
  </create>
</request>
```

To alter a *CICS Resource*, use an HTTP PUT request with URL:

```
http://hostname:port/CICSSystemManagement/CICSDefinitionProgram/cicsname/cicsname/
?CRITERIA=((NAME=='progrname'))&PARAMETER=CSDGROUP(group)
```

and XML body example:

```
<request>
  <update>
    <attributes language="COBOL"/>
  </update>
</request>
```

Chapter 5. Restrictions when connecting to a CICS TS for z/VSE system

This topic describes the restrictions when using the CICS Explorer to connect to a *CICS TS for z/VSE* system.

- When defining a connection you can only select the **CICS Management Interface**. Other types of connection (for example, the CICSplex[®] SM Data Interface) are not supported.
- Within the pull-down menus for the **Operations** and **Definitions** items, not all views are supported. The views that *are* supported are described in Chapter 2, “Supported Operations Views,” on page 7 and Chapter 3, “Supported Definitions Views,” on page 9. If you select a view (for example, Atom Services) that is not supported, the message "CNX0203I This resource is not available" will be displayed.
- Not all CICS Management Interface resources are supported. The resources that *are* supported are listed in Chapter 4, “Supported CICS Management Interface resources,” on page 11.
- The IPv4 protocol only is supported. The IPv6 protocol is *not* supported.
- When using an SSL connection to a z/VSE system that uses TCP/IP for z/VSE it is currently required for the CICS Explorer Client to use the TLSv1 protocol. This can be requested by Window -> Preferences -> Explorer -> Certificate Management.

Chapter 6. Installing and configuring the CICS Explorer

This topic describes the tasks you should follow in order to install and configure the CICS Explorer. This description differentiates between the CICS Explorer client-part (running on a local workstation, a remote network drive, or a shared Linux® server) and the CICS Explorer server-part (running on z/VSE).

Configuring the z/VSE host for use with the CICS Explorer

Before you start the installation of the client-part of the CICS Explorer, you must prepare your z/VSE host system so that the CICS Explorer can connect to a CICS Transaction Server running in a z/VSE partition. The following procedure assumes that the CICS Explorer is used with the second, predefined CICS TS (CICS2) (for a general description on how to setup a second CICS, please refer to [z/VSE Administration, SC34-2692](#)):

1. Skeleton CEXPLCSD is stored in VSE/ICCF Library 59 and *contains the instructions described here*. Change the entry for file EYUPARM in CEXPLCSD so that EYUPARM will be defined as a BAM (Basic Access Method) file. These are the relevant statements in CEXPLCSD:

```
* --V001--  VALID WHERE THE FILE RESIDES                *
* --V002--  START TRACK/BLOCK                          *
* --V003--  NUMBER OF TRACKS/BLOCKS (1 TRACK OR 64 BLOCKS) *
* VALID INPUT COULD BE ANY TRACK, TRAP, OR TRACE COMMAND, AN *
* EXAMPLE WOULD BE (YOU HAVE TO REPLACE THE BLANK LINE *
* AFTER THE $$DITTO CSQ COMMAND):                      *
*          TRAP(NQCR,12)                                *
*          TRACK(XQPQ, FROM, NQLT, SPEC)                *
*          DATTRACE(18)                                *
* YOU WILL BE INSTRUCTED BY IBM PERSONNEL ON SPECIFIC *
* COMMANDS TO BE USED.                                *
```

Now run the skeleton CEXPLCSD, which performs:

- An EXEC DFHCSDUP to define groups SMSSEYU and CMCI. These groups contain the profiles, programs, and transactions that are used with the CICS Explorer. Here is an example for transaction CORM:

```
DEFINE TRANSACTION(CORM) GROUP(SMSSEYU)
PROGRAM(EYU9NXRM) TWASIZE(512) PROFILE(DFHCICST) STATUS(ENABLED)
TASKDATALOC(ANY) TASKDATAKEY(CICS) STORAGECLEAR(NO)
RUNAWAY(50000) SHUTDOWN(DISABLED) DYNAMIC(NO) PRIORITY(255)
TRANCLASS(DFHTCL00) DTIMOUT(NO) INDOUBT(BACKOUT) RESTART(NO)
SPURGE(NO) TPURGE(NO) DUMP(YES) TRACE(YES) CONFDATA(NO)
RESSEC(NO) CMDSEC(NO)
```

- An EXEC BSTADMIN to define the BSM security for the CICS transactions. Here is an example for transaction CORM:

```
ADD TCICSTRN 'CORM' UACC(NONE) DATA('IBM SUPPLIED')
PERMIT TCICSTRN 'CORM' ID(GROUP01) ACCESS(READ)
```

- An EXEC DITTO to define and initialize file EYUPARM. This file is used in job EYUPARM, as described in [Chapter 9, “Using Job EYUPARM to enter or modify debugging commands,”](#) on page 29.

2. Use skeleton DFHDCTC2 to activate EYUPARM and COPR. These are the entries in DFHDCTC2:

```
* EYUPARM DFHDCT TYPE=SDSCI,          CICS EXPLORER PARAMETER
          DSCNAME=EYUPARM,
          BLKSIZE=80,
          RECSIZE=80,
          RECFORM=FIXUNB,
          TYPEFLE=INPUT,
          DEVADDR=SYS075,
```

```

                DEVICE=DISK,
                BUFNO=1

...
* COPR    DFHDCT TYPE=EXTRA,          CICS EXPLORER INPUT PARAMETER
          DESTID=COPR,
          DSCNAME=EYUPARM,
          RECFORM=FIXBLK,
          TYPEFLE=INPUT,
          OPEN=INITIAL

```

3. Add the following DLBL and EXTENT statements to the job DTRCICS2 that defines the labels for a second CICS. Job DTRCICS2 is contained within skeleton SKPREPC2 (stored in VSE/ICCF library 59).

```

// DLBL EYUPARM, 'EYUPARM.FILE', 0, SD
// EXTENT SYS075, --V001--, 1, 0, --V002--, --V003--

```

4. Add the following ASSGN statement for the EYUPARM file to the CICS startup job DTRCICS2:

```

// ASSGN SYS075, DISK, VOL=--V001--, SHR

```

5. In your CICS System Initialization Table (for example DFHSITC2, stored in VSE/ICCF library 59) ensure that:

- SEC=YES
- TCPIP=YES

6. In some environments, the CICS Explorer requires additional 31-bit partition storage.

- If you select Environment C, the amount of partition storage for PRODCICS is sufficient to run the CICS Explorer.
- For other DBDCCICS and Environments A and B, you should increase the partition size (ALLOC procedure) and PASIZE (Tailor IPL). As a result, VSIZE will probably have to be increased.

For Environment A, you should also change the ELIM parameter in skeletons SKCICS2 or SKCICS as follows:

```

// IF XENVNR = A THEN
// SETPARM ELIM=25M

```

(For Environments B and C, the ELIM parameter must not be modified).

7. Create a conversion table using the skeleton DFHCNV (stored in VSE/ICCF library 59) and then perform a CEMT SET PRO (DFHCNV) NEW in the CICS system you are using (either DBDCCICS or PRODCICS).

8. Using the CEDA transaction, install groups SMSSEYU and CMCI in your PRODCICS.

9. Define a TCPIPSERVICE in z/VSE as follows:

```

DEF TC
OVERTYPE TO MODIFY
CEDA DEFINE TCPIP SERVICE(          )
TCPIP SERVICE ==> CMCIT
Group         ==> CMCI
Description   ==> CICS Explorer
Urm          ==> dfhwbody
Portnumber    ==> 27283          1-65535
Certificate   ==>
STatus       ==> Open           Open | Closed
SSL          ==> No             Yes | No | Clientauth
Attachsec    ==> Verify        Local | Verify
TRansaction  ==> cwxn
Backlog      ==> 00001         0-32767
TSqprefix    ==>
Ippaddress   ==> 9.152.88.28
S0cketclose  ==> 0             No | 0-240000

MESSAGES: 2 SEVERE

SYSID=CIC1 APPLID=DBDCCICS

PF 1 HELP 2 COM 3 END          6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
MR b
21/018

```

Figure 3. Defining a TCPIP SERVICE for the CICS Explorer

- In [Figure 3 on page 19](#), you must specify your own IP address.
 - If TCP/IP is running, you can open this TCPIP SERVICE using the command `CEMT SET TCPIPS`.
 - If you wish to use the CICS Explorer for multiple CICS Transaction Server for z/VSE systems, you must use different ports.
 - Per default, the CICS Explorer attempts to connect to a CICS system using the *Security Sockets Layer (SSL)* protocol. If the SSL connection is not successful, the connection will be reattempted using basic authentication. If you wish to use SSL, you must select YES in the field SSL of [Figure 3 on page 19](#). To use SSL connections, you must enter a different port number. In addition, you must define valid certificates.
10. Ensure that the TCP/IP Server is identified by a name for its IP address, otherwise CICS WEB Support will not initialize successfully (and the TCPIP SERVICE can not be opened). Use the TCP/IP command `DEFINE NAME` to do so.
 11. Ensure that the CICS resource definition group `DFH$WBSN` is activated for the CICS Transaction Server for z/VSE system being used with the CICS Explorer.

Start/Stop the CICS Explorer server-part

After completing the installation of the *CICS Explorer client-part* as described in [“Obtaining a copy of the CICS Explorer” on page 20](#) and [“Performing the installation of the CICS Explorer” on page 20](#), you can use the CORM transaction (entered at a 3270 terminal or the z/VSE Console) to start the server-part of the CICS Explorer.

You can use the COSH transaction to stop the server-part of the CICS Explorer.

Obtaining a copy of the CICS Explorer

You obtain a copy of the client-part of the CICS Explorer by downloading it from the CICS Explorer Web page.

1. Start your Web browser and go to this z/VSE Web page:

```
https://www.ibm.com/support/pages/ibm-zvse-service-and-support-corrective-cics-transaction-server
```

2. From within the **CICS Explorer**[®] section, click **Free Product Download** and you will be redirected to the *IBM CICS Explorer Downloads* Web page.
3. Click **Download site** and you will be redirected to the *IBM CICS Explorer Family* Web page.
4. After entering your IBM ID and password, you can proceed to download the CICS Explorer.
5. From the downloaded package, run the "zosexplorer.exe" which starts as the "IBM Explorer for z/OS Aqua".

Note:

- The CICS Explorer that you download from the IBM CICS Explorer Family Web page is used for connecting to *both* CICS TS for z/VSE and CICS TS for z/OS systems. The CICS Explorer recognizes whether the connected CICS TS is running under z/VSE or z/OS and adjusts the views accordingly.
- To use the CICS Explorer with z/VSE 6.2, the CICS Explorer 5.4 (or later) client is required.
- The *server part* of the CICS Explorer is automatically installed into z/VSE sublibrary PRD1 . BASE during the installation of z/VSE.

Performing the installation of the CICS Explorer

You can install the client-part of the CICS Explorer on a local workstation, a remote network drive, or a shared Linux server.

Installing the CICS Explorer on a local workstation:

1. Ensure you have completed the task described in [“Obtaining a copy of the CICS Explorer” on page 20](#) to download the CICS Explorer .zip file (or a .tar.gz file on Linux) from the download site to your local workstation.
2. Extract the contents of the .zip or .tar.gz file to a new directory on your local workstation. For example, to C:\Program Files\Explorer\ on a Windows operating system, or ~/Explorer/ on a Linux operating system.
3. When the extract has completed, open the CICS Explorer directory in your new Explorer directory.
4. Locate the cicsexplorer.exe file (cicsexplorer on Linux) and create a shortcut on the desktop.

You can now double-click the shortcut icon to start the CICS Explorer.

Installing the CICS Explorer on a remote network drive:

1. Ensure you have completed the task described in [“Obtaining a copy of the CICS Explorer” on page 20](#) to download the CICS Explorer .zip file from the download site to your local workstation.
2. Extract the contents of the .zip file to a new directory on the remote network drive.
3. When the extract has completed, open the CICS Explorer directory in the new directory on the remote network drive.
4. Locate the cicsexplorer.exe file and create a shortcut on the desktop.
5. Right-click the shortcut and click Properties. The Target field displays the path to the CICS Explorer executable file on the remote network drive. You must distribute the shortcut to all users who will run the CICS Explorer client-part. If the path from their workstations to the remote server is different from the one already there, you must change the path in the shortcut.

The software can now be shared by multiple users and can be centrally managed. The users start the CICS Explorer by double-clicking the shortcut icon that you distributed.

Installing the CICS Explorer on a shared Linux server:

1. Ensure you have completed the task described in [“Obtaining a copy of the CICS Explorer” on page 20](#) to download the CICS Explorer .tar.gz file from the download site to your local workstation.
2. Log in to the Linux server and create a new directory for the CICS Explorer, for example; /opt/Explorer
3. Extract the contents of the .tar.gz file to the new directory.

The CICS Explorer is now installed on the shared server. Users can use SSH tunneling to access the CICS Explorer and display the output on the local terminal. The CICS Explorer executable file is `cicsexplorer` located in the `CICS_Explorer` directory. For example, the file path is then `/opt/Explorer/CICS_Explorer/cicsexplorer`.

Chapter 7. Configuring a new connection to a CICS system

This topic describes how you can configure a new connection to a CICS system (a CICS Transaction Server for z/VSE or a CICS Transaction Server for z/OS system).

Before you attempt to connect to a CICS system, you might wish to check that the *server-part* of the CICS Explorer has been started using the CORM transaction. You can use the COSH transaction to stop the server-part of the CICS Explorer.

Adding a set of CICS Explorer credentials

Before you can use the CICS Explorer to connect to a CICS system (a CICS Transaction Server for z/VSE or a CICS Transaction Server for z/OS), you must have *at least one* set of credentials. Each time connect to a CICS system, your *credentials* (user-ID and password/passphrase) are sent to the CICS system for authentication.

After you have defined a set of credentials, you can use them on all CICS systems that share the credentials (you are not required to re-enter your details each time).

Before starting this procedure, ensure that you have the correct level of authorization to connect to your CICS system.

To add a set of credentials, you must:

1. Click **Window -> Manage Connections** on the menu bar shown in [Figure 1 on page 4](#). The *Host Connections* window shown in [Figure 4 on page 23](#) then opens.

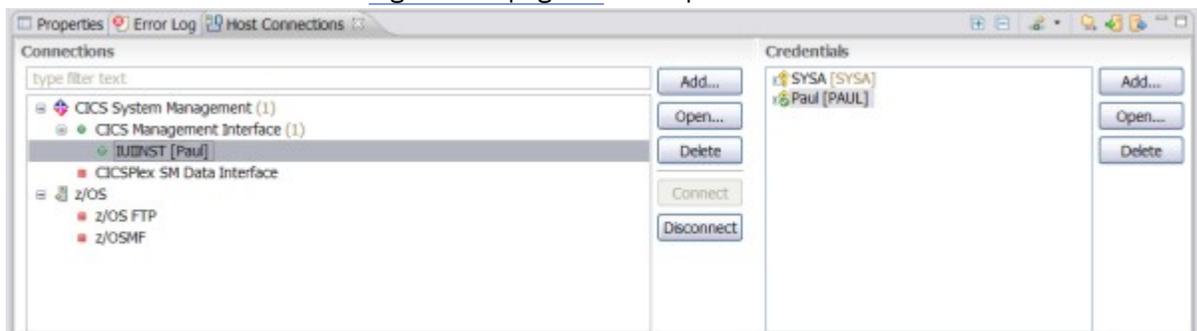


Figure 4. Host Connections window

2. Click **Add** in the Credentials section of [Figure 4 on page 23](#) and a *New Credentials* window opens, shown in [Figure 5 on page 24](#). Enter your credential details and provide a credential name.

The name can be anything you like, and is used only to help you distinguish between different credentials. If you do not type a Credentials Name, the default name used is the same as the User ID.

Enter a password or passphrase, and click the **Save password** check box to save the password.

Note: If you do not enter a password or passphrase, you will be prompted to enter a password or passphrase with *each* connection request!



Figure 5. New Credentials window

Finally, click **OK**.

If you now wish to add a connection to a CICS system, proceed to [“Adding a CICS Explorer connection”](#) on page 24.

Adding a CICS Explorer connection

This topic describes how to add a CICS Explorer connection that can be used for connecting to a CICS system.

Before you attempt to connect to a CICS system, you might wish to check that the *server-part* of the CICS Explorer has been started using the CORM transaction. You can use the COSH transaction to stop the server-part of the CICS Explorer.

1. If a *Host Connections* window is not already open, on the menu bar shown in [Figure 1](#) on page 4 in topic [“Overview of the CICS Explorer Workbench”](#) on page 3, click **Window -> Manage Connections**. A *Host Connections* window then opens.
2. Click **Add** in the Connections section and a pull-down menu is displayed offering a choice of CICS systems:
 - CMCI (CICS Management Interface)
 - CICSplex SM Data Interface
 - z/OS® FTP
 - z/OSMF
 - z/OS Remote System
3. Your selection now depends upon whether the connection is to a CICS Transaction Server for z/VSE system or to a CICS Transaction Server for z/OS system:
 - If you wish to connect to a CICS Transaction Server for z/VSE system, select **CMCI (CICS Management Interface)**. All other options are not supported.
 - If you wish to connect to a CICS Transaction Server for z/OS system, you can select any of the four options shown in (2.) above.

The *Add CMCI Connection* window opens, as shown in [Figure 6](#) on page 25.

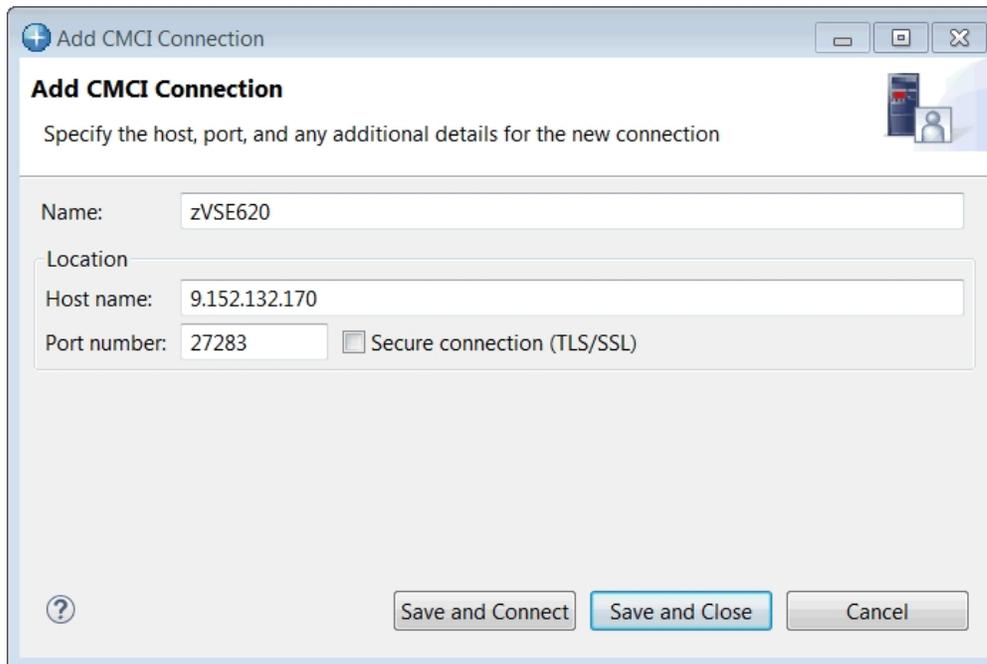


Figure 6. Add CMCI Connection window

4. Now enter:

Name

The local name used to identify this connection. The name can be anything you choose and is used only to help you distinguish between different connections.

Host name

The TCP/IP host name or IP address of the host system running your CICS region (partition).

Port number

The port used to access the server. This is the value specified in the PORTNUMBER attribute of the TCPIPSERVICE definition that was created during the configuration of the CICS management interface.

5. If you click **Save and Connect** in Figure 6 on page 25, the CICS Explorer attempts to connect to the CICS system you have configured. The connection will also be stored for your future use. The *Signon* dialog is now displayed. Enter your User ID and password, and click **OK**.

- *If the connection is successful*, the connection name appears in the connection status bar in the lower right corner of the Workbench window (shown in Figure 1 on page 4), next to either a:
 - *green icon* which indicates a *non-SSL connection*.
 - *padlock* which indicates an *SSL connection*.
- *If the connection is not successful*, a *red icon* is displayed in the connection status bar in the lower right corner of the Workbench window (shown in Figure 1 on page 4), next to the connection name. An error message is displayed at the top of the Connections Preferences view, giving the reason for the failure. Check the values in the fields, correct any errors, and click **Connect** to test your corrections.

6. The first time you use the CICS Explorer to connect to a CICS system, a user workspace is automatically created in a default location. If you wish to change this default location, see [“Changing a CICS Explorer user workspace”](#) on page 28.

Chapter 8. Using an existing connection to a CICS system

This topic describes how to use an *existing* CICS Explorer connection to connect to a *CICS system* (a CICS Transaction Server for z/VSE or a CICS Transaction Server for z/OS).

Before you attempt to connect to a CICS system, you might wish to check that the *server-part* of the CICS Explorer has been started using the CORM transaction. You can use the COSH transaction to stop the server-part of the CICS Explorer.

To use an existing CICS Explorer connection:

1. If a *Host Connections* window is not already open, on the menu bar shown in [Figure 1 on page 4](#), click **Window -> Manage Connections**. A *Host Connections* window then opens.
2. Your selection now depends upon whether the connection is to a CICS Transaction Server for z/VSE system or to a CICS Transaction Server for z/OS system. If you wish to connect to a CICS Transaction Server for z/VSE system:
 - a. Click the box next to **CICS System Management** and **CICS Management Interface** is displayed.
 - b. Click the box next to **CICS Management Interface** and the CICS Transaction Server for z/VSE systems to which you can connect are displayed.
 - c. Select a CICS Transaction Server for z/VSE system together with a set of Credentials, as shown in the [Figure 7 on page 27](#) example.

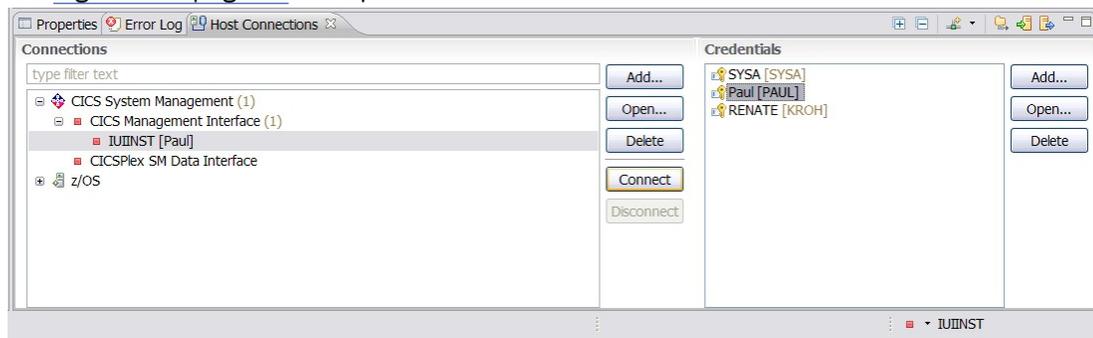


Figure 7. Selecting a Host Connection to a CICS Transaction Server for z/VSE system

- d. Click **Connect**.

If you wish to connect to a CICS Transaction Server for z/OS system:

- a. Click the box next to **z/OS** and **z/OS FTP** and **z/OSMF** are displayed.
 - b. Select one of these CICS Transaction Server for z/OS systems together with a set of Credentials.
 - c. Click **Connect**.
3. The CICS Explorer attempts to connect to the CICS system you have selected.
 - *If the connection is successful*, the connection name appears in the connection status bar in the lower right corner of the Workbench window (shown in [Figure 7 on page 27](#)), next to either a:
 - *green icon* which indicates a *non-SSL connection*.
 - *padlock* which indicates an *SSL connection*.
 - *If the connection is not successful*, a *red icon* is displayed in the connection status bar in the lower right corner of the Workbench window (shown in [Figure 7 on page 27](#)), next to the connection name. An error message is displayed at the top of the Connections Preferences view, giving the reason for the failure. Check the values in the fields, correct any errors, and click **Connect** to test your corrections.

Changing a CICS Explorer user workspace

This topic describes how to change a user's CICS Explorer workspace from the default location that is created automatically by the CICS Explorer when a user starts the CICS Explorer for the first time.

The CICS Explorer workspace contains connection and configuration information. Because the workspace contains user IDs and passwords, you should ensure that the workspace can only be accessed by the *owning user*.

Per default, the CICS Explorer creates a workspace on the local file system.

Under Linux, the default user workspace is created on:

```
<drive letter>/home/<username>/.cicsexplorer
```

Under Windows, the default user workspace is created on:

```
<drive letter>\Documents and Settings\<username>\.cicsexplorer
```

To change the location of a user's workspace:

1. Click **Help** in the CICS Explorer Workbench window shown in [Figure 1 on page 4](#).
2. Select **Help Contents** from the pull-down menu. The *IBM CICS Explorer User Guide* is then opened.
3. Search for *changing the default workspace location* and the information you require is then displayed.

Chapter 9. Using Job EYUPARM to enter or modify debugging commands

Using job EYUPARM, you can enter CPSM system parameter or modify the debugging commands and their parameters that are used for debugging the CICS Explorer.

You can then use CICS transaction COD0 to start debugging from the moment the CICS Explorer is started. Per default, debugging is *not* activated.



Warning: You should only use the CICS Explorer debugging transactions COD0 and CODB when requested to do so by IBM Customer Support Personnel!

You can use the EYUPARM job to change or enter the TRACE, TRAP and TRACK commands. In the sample below:

- The value of TRAP for NQCR has been set at level 12.
- The TRACK statement activates trace flags for the XQPQ method only when it is called from the NQLT method.
- The value of DATTRACE has been set to level 18.

```
// JOB EYUPARM
// DLBL EYUPARM, 'EYUPARM.FILE', 0, SD
// EXTENT SYS001, SYSWK1, 1, 0, 1577060, 64
// UPSI 1
// EXEC DITTO
$$DITTO SET EOD=@@@
$$DITTO CSQ BLKFACTOR=1, FILEOUT=EYUPARM, RECSIZE=80
TRAP(NQCR, 12)
TRACK(XQPQ, FROM, NQLT, SPEC)
DATTRACE(18)
CACHEMXTND(1024)
@@@
/*
/ &
```

The CACHEMXTND(1024) statement (above) increases the size of the internal cache. This might be required, for example, if the Program view fails with message EYUXC0020E "Cache request exceeds extension size for compid cache".

The *default settings* for the EYUPARM job are "blanks" (no settings):

```
// JOB EYUPARM
// DLBL EYUPARM, 'EYUPARM.FILE', 0, SD
// EXTENT SYS001, SYSWK1, 1, 0, 1577060, 64
// UPSI 1
// EXEC DITTO
$$DITTO SET EOD=@@@
$$DITTO CSQ BLKFACTOR=1, FILEOUT=EYUPARM, RECSIZE=80
@@@
/*
/ &
```

Chapter 10. Diagnosing problems within the CICS Explorer

This section describes the tasks that you perform to resolve problems when the CICS Explorer is used with CICS TS for z/VSE.

As mentioned earlier, the CICS Explorer basically consists of a *client part* (running on a local or remote workstation) and a *server part* (running on CICS TS for z/VSE). Both parts have their own procedures to diagnose and resolve problems. For a description of these procedures within the CICS Explorer *client part*, refer to the appropriate information in the CICS Explorer pull-down menu **Help** → **Help Contents**, in section **CICS Explorer User Guide – Troubleshooting**.

Below we describe procedures to diagnose problems within the CICS Explorer *server part* on CICS TS for z/VSE.

The information is intended for system programmers and others who are responsible for debugging systems that use the CICS Explorer.

Debugging transactions COD0 and CODB

The CICS Explorer *Server part* on CICS TS for z/VSE provides two interactive debugging transactions (COD0 and CODB). They can be used to format and manipulate the internal data structures of the CPSM runtime environment.

Note: CPSM is an abbreviation for CICSplex System Manager, an element of CICS TS for z/OS. Please note that the CICSplex System Manager is not available with CICS TS for z/VSE. Nevertheless, for simplicity we will use the term CPSM when talking about the CICS Explorer *Server part* on CICS TS for z/VSE.

To run these debugging transactions, log on to the CICS system and enter COD0 or CODB.

- COD0

This method-level debugging transaction provides access to CPSM objects, methods, message argument lists (MALs), and outstanding requests. To exit this transaction, type EXIT on the command line.

- CODB

This system-level debugging transaction provides access to address space and data space storage, major control blocks, data queues, and CPSM entries in the CICS trace table. To exit this transaction, press PF3 or type END on the command line.

The following usage rules apply to the COD0 and CODB transactions:

1. You issue a COD0 command by typing the command name on the command line. You issue a CODB command by typing its option number on the command line.
2. The standard END and CANCEL commands are recognized. END completes the task in progress and returns you to the previous screen, while CANCEL cancels the task before returning.
3. You can scroll a display by using the commands DOWN, UP, TOP, and BOT. With COD0, you can also enter a default scrolling amount in the Scroll==> field.
4. On a selection list, any character that is not a blank or an underscore can be used to select an option.
5. These transactions support only 3270 model 2 screens that is, 24x80 and 32x80 type screens.

Note: You should only use the debugging transactions COD0 and CODB when requested to do so by IBM Customer Support Personnel!

Using traces in the CICS Explorer Server part

The CICS Explorer *Server part* on CICS TS for z/VSE includes a special trace facility, the so-called CPSM trace.

There are two ways to activate and control this CPSM trace:

1. Provide trace-related commands and parameters into the EYUPARM file.

Refer to Chapter 9, “Using Job EYUPARM to enter or modify debugging commands,” on page 29 for a description on how to do it.

2. Use the COD0 transaction TRACE flag command.

You can change the trace flag settings of one or more CPSM components by over-typing the component's bit setting.

The CPSM trace entries are written to the CICS internal trace or to the external CICS AUXTRACE data set.

How to format CPSM trace entries from an external CICS AUXTRACE data set

The EYU9XZUT batch utility is provided to format CPSM trace entries from an external CICS AUXTRACE data set. EYU9XZUT provides several parameters that allow you to select the specific trace records to be formatted:

ABBREV

Provides an abbreviated trace, which has one line per trace record with a sequence number at the far right. Use the sequence number to select full trace formatting of specific records. This formatting option requires that logical unit SYS041 is ASSIGNED to SYSLST.

FULL

Provides full trace formatting of trace records meeting all selection criteria. This formatting option requires that logical unit SYS040 is ASSIGNED to SYSLST.

RECOVERY=ONLY|ALL

ONLY formats only abend trace records, regardless of any other criteria that may be specified. ALL formats all abend trace records, as well as all trace records that match any other specified criteria.

SEQ=

Specify one or more sequence numbers to select specific trace records. The sequence number for each trace record appears at the far right of the formatted trace heading. Sequence numbers can be from 1 to 9 characters in length. A sequence number of zero is not valid. Sequence numbers can be specified as a single entry or as a range of entries separated by a hyphen. For example:

```
SEQ=1-99,103,12345-12399
```

If you rerun the trace format utility using SEQ=, in order to get the same trace records you must specify all of the same options that you specified on the first run.

Note:

1. Filename CICSTRC is used by EYU9XZUT for the auxiliary trace data set.
2. EYU9XZUT requires the SORT program to be installed!

This is a sample job for running EYU9XZUT on z/VSE:

```
* $$ JOB JNM=EYU9XZUT,CLASS=0
* $$ LST RBS=2000
// JOB EYU9XZUT          FORMAT CPSM TRACE FROM AUX TRACE DATASET
// DLBL SORTWK1,'SORT1.WORK',0,SD      DFSORT work file
// EXTENT SYS003,DOSRES,1,0,19830,900
// ASSGN SYS003,DISK,VOL=DOSRES,SHR
// DLBL CICSTRC,'CICS2.AUXTRACE',0,VSAM,
//                               CAT=VSESPUC,DISP=(OLD,KEEP)
// ASSGN SYS041,SYSLST          Abbreviated trace output
// EXEC EYU9XZUT,OS390
ABBREVIATED
```

```

/*
// PAUSE Press ENTER to continue
// ASSGN SYS040,SYSLST           Full trace output
// EXEC EYU9XZUT,OS390
FULL
SEQ=1-99,500-520
/*
/&
* $$$ E0J

```

How to format CPSM trace entries from an internal CICS trace within a system dump

To properly format CPSM trace entries perform the following steps:

1. Run the dump exit module DFHPD430 with parameter CPSMTOAUX.

This will copy the internal trace area to an external file with filename DFHAUXT. DFHAUXT must be a file with a fixed recordsize of 4096.

Note: CPSMTOAUX is only accepted when TRACE Domain output is also requested (TR=n with n=1,2,3). With TR=0 (either by request or by default) CPSMTOAUX is ignored with message:

```

*** Parameter CPSMTOAUX ignored because TRACE domain output
        is not requested

```

2. Run batch utility EYU9XZUT for this just created file.

Sample job:

```

* $$$ JOB JNM=DMPACD2N,DISP=D,PRI=8,           C
* $$$ LST DISP=H,RBS=3000
// JOB DMPACD2N ANALYZE CICS/TS DUMP
// EXEC PROC=DTRINFOA
// DLBL DFHAUXT,'CPSM.AUXTRACE',0,VSAM,        X
        CAT=VSESPUC,RECSIZE=4096,            X
        DISP=(NEW,KEEP),RECORDS=(1000,1000)
// EXEC INFOANA,SIZE=INFOANA,OS390
SELECT DUMP MANAGEMENT
    DUMP NAME SYSDUMP.F8.DF800004
    RETURN
SELECT DUMP VIEWING
    CALL DFHPD430 DATA AP=3,KE=3,DS=3,TR=1,LD=3,CPSMTOAUX
    RETURN
    DUMP NAME SYSDUMP.F8.DF800004
SELECT DUMP VIEWING
    PRINT FORMAT
    RETURN
SELECT END
/*
// PAUSE PRESS ENTER TO FORMAT CPSM TRACE
// ASSGN SYS041,SYSLST           Assignment for ABBreviated trace
// DLBL SORTWK1,'SORT1.WORK',0,SD   SORT work file
// EXTENT SYS003,DOSRES,1,0,19830,900
// ASSGN SYS003,DISK,VOL=DOSRES,SHR
// DLBL CICSTRC,'CPSM.AUXTRACE',0,VSAM,        X
        CAT=VSESPUC,DISP=(OLD,KEEP)
// EXEC EYU9XZUT,OS390
ABBREVIATED
//
// ASSGN SYS040,SYSLST           Assignment for FULL trace
// DLBL CICSTRC,'CPSM.AUXTRACE',0,VSAM,        X
        CAT=VSESPUC,DISP=(OLD,DELETE)
// EXEC EYU9XZUT,OS390
FULL
/*
/&
* $$$
E0J

```

Chapter 11. Messages generated when using the CICS Explorer

This topic provides a summary of the message prefixes that you might receive when using the CICS Explorer and where you can obtain explanations of these messages.

For an explanation of messages that have the prefix:

- **CNX** (for example, CNX0222W), you should start the CICS Explorer and then:
 - From the tool bar, select **Help** and then **Help Contents**. The *Help - IBM CICS Explorer for z/OS* window then opens.
 - Click on the plus sign to the left of the **CICS Explorer User Guide**.
 - Click on the plus sign to the left of **Reference**.
 - Click on the plus sign to the left of **Messages** and then **CICS Explorer (CNX) Messages**. All current messages with prefix CNX will be listed. You can now select the message whose details you require.
- **EYU** (for example, EYUAR0001E). Most of the EYU prefix messages that you might receive with z/VSE 6.2 are included in the *z/VSE Messages and Codes Volume 1, SC34-2683*. If a message is missed, you can search for **CICSplex SM Messages and Codes** at the CICS TS Knowledge Center:

<https://www.ibm.com/support/knowledgecenter/SSGMGV/welcome.html>

- **DFHWU** (for example, DFHWU4002) you can search for **CICS TS Messages and Codes Vol 2** at the CICS TS Knowledge Center:

<https://www.ibm.com/support/knowledgecenter/SSGMGV/welcome.html>

- **IZE** (for example, IZE0106E) you should start the CICS Explorer and then:
 - From the tool bar, select **Help** and **Help Contents**. The *Help - IBM Explorer for z/OS* window then opens
 - Click on the plus sign to the left of **IBM Explorer for z/OS User's Guide**
 - Click on the plus sign to the left of **z/OS Explorer (IZE) Messages**

All current messages with prefix IZE will be listed. You can now select the message whose details you require.

Part 2. CICS Web support

Chapter 12. Configuring CICS Web support

Note! This topic is originally taken from the [CICS Internet Guide](#), SC34-5765 (September 2000). It has been updated for Secure Sockets Layer (SSL) and HTTP/1.1 support. You might find useful information in the other topics of the [CICS Internet Guide](#) publication, such as:

- "Introduction to CICS Web support"
- "Planning for CICS Web support"
- "Writing CICS programs to process HTTP requests"
- "Security for CICS Web support"

This topic provides an overview of how to configure CICS Web support. [Table 2 on page 39](#) is a checklist of what you need to do.

Task	See...	Task completed?
Specifying the appropriate system initialization (SIT) parameters.	"Controlling web support with system initialization parameters" on page 39	YES / NO
Creating the necessary resource definitions.	"Defining resources to CICS" on page 40	YES / NO
Reserving ports for CICS Web support	"Reserving ports for CICS Web support" on page 45	YES / NO
Specifying a name server (optional).	"Specifying a name server" on page 46	YES / NO
Enabling lightpen support (optional).	"Enabling lightpen support" on page 46	YES / NO
Running the sample application to test CICS Web support.	"Running the sample application" on page 46	YES / NO
Use SSL support (optional). Refer to Chapter 13, "CICS TS for z/VSE Support of HTTP/1.1," on page 49 for a description of the HTTP/1.1 support in CICS TS for z/VSE .	Chapter 21, "Configuring CICS to use SSL," on page 129	YES / NO

Controlling web support with system initialization parameters

CICS Web support is controlled initially by *system initialization parameters*. When CICS is running, you can make changes using CEMT and CEDA.

There are four CICS system initialization parameters relating to CICS Web support, and a further three parameters relating to SSL:

- The TCPIP parameter specifies whether CICS TCPIP services are to be activated at CICS startup. The default is NO, meaning that HTTP services cannot be enabled, and you cannot use any TCPIP SERVICE resources defined with CEDA. If TCPIP is set to YES, HTTP services can be enabled and can then process work.
- The MAXSOCKETS parameter specifies the maximum number of IP sockets that can be managed by the CICS socket domain.

- If you are using Web 3270 support, you can use the WEBDELAY parameter to fix:
 - The length of time, in minutes, after which a Web task and its associated data is marked for deletion if no activity takes place on it.
 - The frequency, in minutes, with which the garbage collection transaction CWBG is run to delete the marked tasks and their data.
- If you intend to use Secure Sockets Layer (SSL) as described in Chapter 21, “Configuring CICS to use SSL,” on page 129, there are *three* further CICS system initialization parameters that you must use:
 - the ENCRYPTION parameter to specify the level of encryption you want for TCP/IP connections using SSL.
 - the KEYFILE parameter to specify the name of the VSE Keyring Library on the z/VSE host that you wish to use with CICS.
 - the SSLDELAY parameter to specify the length of time for which CICS retains session IDs for secure socket connections.

(You must also set parameter TCPIP to YES).

For details of all the system initialization parameters described here, see “[The system initialization parameter descriptions](#)” on page 316.

Defining resources to CICS

This section describes the resources needed to configure CICS Web support.

It contains these topics:

- “[CICS supplied resource definitions](#)” on page 40
- “[DOCTEMPLATE definitions](#)” on page 40
- “[TCPIPSERVICE definitions](#)” on page 42
- “[TRANSACTION definitions for extra alias transactions](#)” on page 42
- “[PROGRAM definitions for user-replaceable programs](#)” on page 43
- “[Setting up a sublibrary for the template manager](#)” on page 43
- “[Defining a conversion table](#)” on page 43

CICS supplied resource definitions

CICS Web support provides an RDO group defining the CICS resources used by the interface. The following definitions are in the locked group DFHWEB:

- Transactions required by CICS Web support (for example, CWBA and CWXN).
- Programs supplied with the CICS Web support.

To change these definitions, you must copy them to your own RDO group and modify them there.

Sample CICS Web TCPIPSERVICE definitions are provided in the locked group DFH\$SOT. To change these definitions, you must copy them to your own group and change them there.

The group DFH\$WBSN contains the resource definitions for the security sample programs described in the section “Security for CICS Web support” of the [CICS Internet Guide, SC34-5765](#).

DOCTEMPLATE definitions

DOCTEMPLATE definitions allow you to perform variable substitution on documents in a manner similar to that done by BMS for 3270 screens. Templates can contain HTML, or binary data such as images. The data within the template is retrieved whenever a call is made for the template by means of an EXEC CICS DOCUMENT CREATE or EXEC CICS DOCUMENT INSERT command. The template can reside in any of the following places:

- z/VSE sublibrary.
- CICS auxiliary temporary storage.
- CICS extrapartition transient data.
- CICS load module.
- CICS file.
- Exit program.

See the Chapter 26, “CEDA DEFINE DOCTEMPLATE,” on page 153 for details of how to define a DOCTEMPLATE, and information about programming with documents and the associated EXEC CICS DOCUMENT commands.

z/VSE sublibrary

You can use any editor you like to create the templates as members of this sublibrary. The record format must be F (fixed). The templates can contain sequence numbers as follows:

- F format, and LRECL 80: the sequence numbers must be in record positions 73 through 80.

In any other case, there must be no sequence numbers in the records. The template manager decides whether there are sequence numbers by looking at the first logical record of a member of the z/VSE sublibrary, so members that are only partially sequenced might be interpreted incorrectly. The default library name is DFHHTML. The sublibrary name must be DFHDOC. The members must have a member type of HTML.

CICS temporary storage

Define one TSQUEUE for each template. The document handler domain returns an error if a request for a template is made to a non-existent TSQUEUE.

CICS transient data

Define an extrapartition TDQUEUE for each template. If you use an intrapartition transient data queue, your data is lost as soon as it has been read. If you use an extrapartition data queue, you must reset the queue after reading it.

CICS load module

Compile and link-edit a data-only load module. For example, an Assembler CSECT could contain a PROLOG containing your own control information, an ENTRY statement, any number of DC statements containing the HTML you want to output (you must put your own linefeeds in), and an END statement. CICS assumes that the entry point of the load module delimits the start of the template.

CICS file

This can be any CICS-controlled file.

Exit program

This is called whenever a request is made for the template. CICS passes a commarea to the exit program which is mapped by the following copybooks:

- DFHDHTXD (Assembler)
- DFHDHTXH (C)
- DFHDHTXL (PL/I)
- DFHDHTXO (COBOL)

The commarea contains the address (`dhtx_buffer_ptr`) and length (`dhtx_buffer_len`) of a CICS-supplied buffer in which the EXITPGM must return the template. The actual length of the template must be returned in `dhtx_template_len`. If the template to be returned is longer than `dhtx_buffer_len`,

the template must be truncated to length `dhtx_buffer_len` and the EXITPGM must set the length required in `dhtx_template_len`. The EXITPGM is then called again with a larger buffer.

TCPIP SERVICE definitions

For HTTP requests to be submitted directly to CICS, you need one or more TCPIP SERVICE resources to be installed.

The TCPIP SERVICE definition allows you to define which TCP/IP services are to use CICS internal Sockets support. The internal CICS service that can be defined is CICS Web support.

The TCPIP SERVICE definition allows you to manage these internal CICS interfaces, with CICS listening on multiple ports, with different flavors of CICS Web support on different ports.

You must install and open a TCPIP SERVICE definition for each port on which CICS is to listen for incoming HTTP requests. You can create your own TCPIP SERVICE definition, or copy the HTTPNSL definitions from the DFH\$SOT group into your own group and modify them to meet your system requirements.

The important attributes for a Web TCPIP SERVICE are:

- The STATUS must be OPEN
- The TRANSACTION to be attached by CICS when new work arrives on the specified port must be CWXN or a user-defined alias of CWXN, which must invoke DFHWBXN as the initial program.
- The port on which CICS is to listen
- The backlog of requests to be processed which TCP/IP for z/VSE is to allow
- The name of the analyzer user-replaceable module to be driven for TCPIP SERVICE
- The IP address on which CICS is to listen for incoming requests.
- The TS queue name. This is the 6-character prefix of TS queue names generated by CICS Web support when writing inbound and outbound data to temporary storage. If no prefix is supplied on the definitions, the default name of DFHWEB is used to generate TS queue names.
- The name of a certificate contained in the VSE Keyring Library, that TCP/IP services should use.
- The level of SSL to be used (either no SSL support, SSL support with server authentication only, or SSL support with server authentication *and* client authentication).

For more information on defining Web TCPIP SERVICES, see [Chapter 28, “CEDA DEFINE TCPIP SERVICE,” on page 161](#).

TRANSACTION definitions for extra alias transactions

Three CICS transactions are provided with CICS Web support:

Web attach transactions (CWXN and CWXU).

The CICS-supplied transactions CWXN and CWXU are the default transactions that are attached to process new requests received for a TCP/IP Service. CWXN is the default for a TCP/IP Service with PROTOCOL(HTTP), and CWXU is the default with PROTOCOL(USER). Both transactions execute program DFHWBXN. They are started by the Sockets listener task CSOL when a new connection request is received on the port specified on the TCPIP SERVICE definition. They establish the context in which the alias transaction CWBA is to be run, and issue the appropriate ATTACH command. Before CICS Transaction Server for z/VSE, Version 2 Release 2, if a Web client and CICS had a persistent ("keep-alive") connection, the CWXN transaction would remain in the system for the duration of the persistent connection. Now, the CWXN transaction terminates after each request from the Web client has been passed to the alias transaction (CWBA or another transaction), or after the static response has been delivered. The Sockets listener task monitors the socket and initiates a new instance of CWXN for each request on the persistent connection.

Alias transaction CWBA.

An alias transaction is a CICS-supplied transaction that is started by the Web attach transaction (CWXN) to process a single request. Many instances of the alias transaction can be active in a CICS system at the same time, each processing a different request. The alias transaction runs the

CICS-supplied alias program that calls the CICS program. If you wish, you may set up additional transaction definitions for alias transactions, each using the CICS-supplied alias program.

You may want to use other alias transaction names for various reasons:

- Auditing
- Resource and command checking
- Allocating initiation priorities
- Allocating database plan selection
- Assigning different runaway values to different CICS programs

If you do want to use other alias transaction names, you must copy the definition of CWBA, making the necessary changes. The definition of CWBA is as follows:

```
DEFINE TRANSACTION(CWBA)    GROUP(DFHWEB)
PROGRAM(DFWBA)             TWASIZE(0)
PROFILE(DFHICST)           STATUS(ENABLED)
TASKDATALOC(ANY)          TASKDATAKEY(USER)
RUNAWAY(SYSTEM)           SHUTDOWN(DISABLED)
PRIORITY(1)                TRANCLASS(DFHTCL00)
DTIMOUT(NO)               INDOUBT(BACKOUT)
SPURGE(YES)               TPURGE(NO)
RESSEC(NO)                 CMDSEC(NO)
```

You cannot change the program name in this definition. Only the CICS-supplied alias program DFWBA can be used. All the extra alias transactions must be local transactions.

PROGRAM definitions for user-replaceable programs

Each incoming request is serviced by a CICS program that provides transaction processing services, and by two other user-replaceable programs, an analyzer (required) and a converter (optional).

If you are not using autoinstall for programs, you must define all the user-replaceable programs you use. If you are using autoinstall for programs, you do not need to define the converters. In any case analyzers must be defined with EXECKEY(CICS). All the user-replaceable programs must be local to the system in which CICS Web support is operating.

Setting up a sublibrary for the template manager

If you use the HTML template manager for constructing HTTP responses, you may provide a z/VSE sublibrary to hold the templates. You can use any editor you like to create the templates as members of this data set. The record format must be F (fixed). The templates can contain sequence numbers as follows:

- F format, and LRECL 80: the sequence numbers must be in record positions 73 through 80.

In any other case, there must be no sequence numbers in the records. The template manager decides whether there are sequence numbers by looking at the first logical record of a member of the z/VSE sublibrary, so members that are only partially sequenced might be interpreted incorrectly.

Any z/VSE library can be used to specify sublibrary member templates, as specified in the DOCTEMPLATE definition. If you are using the template manager (DFHWBTL) or the Web bridge (DFHWBTTA), references to templates that are not defined and installed as DOCTEMPLATE definitions are resolved as members of the library specified as DFHHTML. The sublibrary name must be DFHDOC. The members must have a member type of HTML.

Defining a conversion table

When CICS exchanges messages with a Web client, character data in the messages normally needs to undergo code page conversion on entering and leaving the CICS environment. Code page conversion for text in messages is required for two reasons:

- CICS and user-written applications for CICS typically use an EBCDIC encoding, but Web clients typically use an ASCII encoding.
- Within each encoding, a number of different code pages are used to support national languages.

Non-text content of messages, such as images or application data, does not require code page conversion.

In releases of CICS before CICS Transaction Server for z/VSE, Version 2 Release 2, code page conversion for CICS Web support was handled using a code page conversion table (DFHCNV).

In CICS Transaction Server for z/VSE, Version 2 Release 2, a code page conversion table is normally not required for CICS Web support.

However, if you have existing request processing structures that include an analyzer program which references entries in the conversion table, and you do not want to change the analyzer program, you can continue to provide DFHCNV entries. As an alternative, you can change the analyzer program. Changing the analyzer program involves coding two new output parameters to specify the client and server code pages, in place of the output parameter that specified the name of a DFHCNV entry. As a pair of parameters specifying the character set used by the Web client (wbra_characterset), and the host code page suitable for the application program (wbra_hostcodepage).

Specifying the parameters in this way means that an entry in the code page conversion table (DFHCNV) is not required.

If you have commarea-style Web applications which do not use the Web API, or you are using CICS Web support to run a terminal-oriented transaction, you need to create or modify a DFHCNV table for data conversion to allow CICS to deal with incoming requests. The use of the DFHCNV macro for defining the table is described in [CICS Family: Communicating from CICS on System/390](#). There are two kinds of data conversion performed in CICS Web support:

Conversion of the HTTP header information. This information is always transmitted as ASCII data using the ISO 8859-1 (Latin-1) character set.

This is the base character set for HTTP and HTML. This data has to be translated into EBCDIC. The conversion template name that the server controller supplies to the DFHCNV program, which does the translation, is DFHWBHH.

Conversion of the HTTP user data.

This information is transmitted in the code page of the HTTP client, and can be translated into EBCDIC if required. The conversion template name is supplied by the analyzer. If the request is not an HTTP request, all the request is translated using the name supplied by the analyzer.

You might also refer to the sample conversion table DFHCNV contained in VSE Library 59.

For data conversion of the HTTP headers, you need to create a conversion template as follows:

```
DFHCNV TYPE=ENTRY, *
      RTYPE=PC, *
      CLINTCP=8859-1, *
      SRVERCP=037, *
      RNAME=DFHWBHH, *
      USREXIT=NO
DFHCNV TYPE=SELECT, OPTION=DEFAULT
DFHCNV TYPE=FIELD, OFFSET=0, DATATYP=CHARACTER, DATALEN=32767, *
      LAST=YES
```

In the TYPE=ENTRY macro, the RNAME parameter must be DFHWBHH. The code page specifications CLINTCP and SRVERCP will get the HTTP request headers translated from ASCII to EBCDIC, and the HTTP response headers translated from EBCDIC to ASCII. The TYPE=SELECT and TYPE=FIELD macros must be coded exactly as shown.

For each name that the analyzer might specify for translating user data in the request from the client code page into EBCDIC, and for translating the user data in the response from EBCDIC to the client code page, you need to create a conversion template as follows:

```
DFHCNV TYPE=ENTRY, *
      RTYPE=PC, *
      CLINTCP=8859-1, *
```

```

SRVERCP=037, *
RNAME=DFHWBUD, *
USREXIT=NO
DFHCNV TYPE=SELECT,OPTION=DEFAULT
DFHCNV TYPE=FIELD,OFFSET=0,DATATYP=CHARACTER,DATALEN=32767, *
LAST=YES

```

In the TYPE=ENTRY macro, the CLINTCP parameter must specify the code page of the client, and the RNAME parameter must specify the name that the analyzer will supply. The sample entry above supports translation of user data in the request from ASCII to EBCDIC, and of the user data in the response from EBCDIC to ASCII, for the default analyzer, which uses the name DFHWBUD. You may code the TYPE=SELECT and TYPE=FIELD macros in any way that is appropriate to the format of the user data that the client sends.

You may use the TYPE=INITIAL macro to set defaults for some of the values specified in these samples, as explained in [CICS Family: Communicating from CICS on System/390](#).

The following sample shows a complete definition of the conversion templates for use with a Web browser using a Japanese double-byte character set. The code page 932 is one of several code pages for Japanese Web browsers, and 931 is one of the corresponding System/390® code pages. This sample can be used with the default analyzer. The sample must be assembled using the High-Level Assembler before being used.

```

DFHCNV TYPE=INITIAL
DFHCNV TYPE=ENTRY,RTYPE=PC,RNAME=DFHQBHH,USREXIT=NO, *
SRVERCP=037,CLINTCP=8859-1
DFHCNV TYPE=SELECT,OPTION=DEFAULT
DFHCNV TYPE=FIELD,OFFSET=0,DATATYP=CHARACTER,DATALEN=32767, *
LAST=YES
DFHCNV TYPE=ENTRY,RTYPE=PC,RNAME=DFHQBUD,USREXIT=NO, *
CLINTCP=932,SRVERCP=931
DFHCNV TYPE=SELECT,OPTION=DEFAULT
DFHCNV TYPE=FIELD,OFFSET=0,DATATYP=CHARACTER,DATALEN=32767, *
LAST=YES
DFHCNV TYPE=FINAL
END

```

Configuring TCP/IP for z/VSE

This section describes the changes you must make to TCP/IP for z/VSE as part of configuring CICS Web support.

Reserving ports for CICS Web support

You are recommended to reserve as many ports as you need for CICS Web support, and to ensure that CICS Web support has exclusive use of those ports.

Application programmers may use port numbers from 256 to 32 767 for nonstandard servers. For z/VSE, be careful when choosing port numbers less than 1024 — the "well-known" ports.

The maximum length of any queue of requests for a TCP/IP port on which a program is listening is controlled by TCP/IP for z/VSE, and is currently defined to be equivalent to a maximum of 10. This cannot be modified.

Identifying the TCP/IP server

You must identify the TCP/IP for z/VSE server by specifying a name for its IP address, otherwise CICS Web Support will not initialize successfully. You use the TCP command DEFINE NAME to do so. For example:

```
DEFINE NAME,NAME=WINVSE.HURSLEY.IBM.COM,IPADDR=9.20.101.97
```

where NAME is a meaningful host name, and IPADDR specifies your server IP address.

If you do not do this, you will see the following error message during CICS start-up:

```
DFHS00117 applid Unable to determine the TCP/IP host name.  
Language Environment return code X'00000458', return code X'00000000'.  
TCP/IP services are unavailable.
```

Specifying a name server

If you want full CICS function (that is, if you want to use DFH\$WBSN and DFH\$WBENV), CICS Web support needs to access a name server during its operation. You set the name server in TCP/IP for z/VSE using the TCP command SET DNS1. For example:

```
SET DNS1=n.n.n.n
```

where *n.n.n.n* is the dotted decimal address of the name server.

To override the default search, you can code individual name, IP address pairs using the TCP command DEFINE NAME. For example:

```
DEFINE NAME,NAME=name,IPaddr=n.n.n.n
```

If the name server lookup fails when CICS runs:

- The security sample program DFH\$WBSN does not execute correctly.
- The environment variables program DFH\$WBENV does not return a connection name in SERVER_NAME, but the dotted decimal address of the connection, and it also returns a null string for REMOTE_HOST.

Enabling lightpen support

To enable selector pen processing over the CICS Web support 3270 bridge, you must define a bridge facility with lightpen support enabled. To do this, follow these steps:

1. Copy the following definitions to a new group. Unless all applications running on the CICS system require lightpen support, you should also rename both definitions:
 - The CICS-supplied bridge facility CBRF, in group DFHTERM.
 - Its default TYPETERM, DFH\$LU2, in group DFH\$TYPE.
2. In the TYPETERM definition, change the LIGHTPEN option under "DEVICE PROPERTIES" to YES.
3. In the TERMINAL definition, change the TYPETERM parameter to point to the new TYPETERM.
4. Install the definitions in the CICS region.
5. If you have created a new bridge facility definition, update the PROFILE definition of the 3270 transaction which you are going to run with CICS Web support, so that the bridge facility will be modelled on the new TERMINAL/TYPETERM definition:
 - a. Identify the PROFILE that the transaction uses by using CEDA to view the PROFILE parameter of the TRANSACTION definition.
 - b. If the profile is a CICS-supplied profile, make a copy of it to your own group and rename it.
 - c. Alter the new PROFILE and enter the name of your new bridge facility in the FACILITYLIKE parameter.
 - d. Alter your TRANSACTION definition to use the new PROFILE definition.

Running the sample application

A sample application DFH\$WB1A is provided to help you test the operation of CICS Web support. From a suitable Web browser, enter a URL that connects to CICS Web support with absolute path /CICS/CWBA/DFH\$WB1A. The response displays the message "DFH\$WB1A on system xxxxxxxx successfully invoked through the CICS Web support." with xxxxxxxx replaced by the application ID of the CICS system in which CICS Web support is running.

Using Sample Programs for Security

The CICS Internet Guide, SC34-5765, section "Using sample programs for security", points to a number of security sample programs. Note that sample programs DFH\$WBSC and DFH\$WBSN include the CICS translator option statement *ASM XOPTS(SP NOPROLOG NOEPILOG). Any compile skeleton or job that is used to compile these sample programs (for example as generated by the IUI option 8 = Compile) must not overrule the CICS translator option statement in the source files.

Chapter 13. CICS TS for z/VSE Support of HTTP/1.1

This section describes the CICS TS for z/VSE Support of HTTP/1.1

HTTP/1.1 compliance for CICS as an HTTP server

Releases of CICS before CICS Transaction Server for z/VSE, Version 2 Release 2, supported HTTP/1.0.

Starting with CICS TS for z/VSE 2.2, CICS Web support has been enhanced to handle and provide features of the HTTP/1.1 specification, including chunked transfer-coding, pipelining, and persistent connections.

CICS Web support is conditionally compliant with the HTTP/1.1 specification, as described in the Internet Society and IETF (Internet Engineering Task Force) Request for Comments document RFC 2616, *Hypertext Transfer Protocol -HTTP/1.1* (<http://www.ietf.org/rfc/rfc2616.txt>).

Conditional compliance with the HTTP/1.1 specification means that CICS satisfies all the "MUST" level requirements, but not all the "SHOULD" level requirements, that are detailed in the HTTP/1.1 specification, where these requirements are relevant to the functions actually provided by CICS itself. An implementation that satisfies all the MUST or REQUIRED level and all the SHOULD level requirements for its protocols is said to be unconditionally compliant. An implementation is not compliant if it fails to satisfy one or more of the MUST or REQUIRED level requirements for the protocols it implements.

There are three aspects to CICS Web support compliance with the HTTP/1.1 specification.

- CICS Web support performs actions that are required from an HTTP server. For example, CICS Web support receives inbound requests, maintains persistent connections, writes certain HTTP headers, and transmits responses. The behavior of CICS Web support during these actions is conditionally compliant with the HTTP/1.1 specification. Where necessary, this represents a change in behavior from earlier releases of CICS. [“CICS Web support behavior in compliance with HTTP/1.1” on page 50](#) describes how the behavior of CICS Web support complies with HTTP/1.1, and where this differs from releases of CICS before CICS Transaction Server for z/OS, Version 2 Release 2.
- Some parts of the HTTP/1.1 specification are not relevant to CICS Web support. For example, CICS Web support does not act as a proxy server or provide a caching facility. [“HTTP functions not supported by CICS Web support” on page 51](#) notes these areas, so that you know the areas in which HTTP/1.1 compliance is not a concern for CICS Web support and for your user application program.
- Web-aware user application programs in CICS can be used to create application-generated HTTP responses and instruct CICS Web support how to serve the responses. If you want your CICS Web support implementation to be compliant with the HTTP protocol specifications, in particular HTTP/1.1, your user application programs share the responsibility for compliance in the actions that they perform. Some basic guidance information is provided in this documentation, but it is important to check the HTTP specification to which you are working for more detailed information. In practical terms, the different levels of requirement in the HTTP/1.1 specification (MUST, SHOULD, or MAY) should be handled by your application program as follows:
 - MUST level requirements must be implemented to maintain compliance. CICS Web support is designed to handle, or to assist you to comply with, all the MUST level requirements that apply to straightforward activities. Some additional MUST level requirements might apply if you choose to fulfill an optional requirement that CICS Web support does not already handle. Also, some MUST level requirements relate to actions that your application must *not* perform in certain circumstances.
 - SHOULD level requirements are not necessary for conditional compliance with the HTTP specifications. CICS does not comply with all the SHOULD level requirements in the HTTP/1.1 specification. However, if your application can comply with a SHOULD level requirement without inconvenience, it is advisable to do so.
 - MAY actions are optional for any HTTP implementation, whatever its level of compliance. They should be treated as suggestions or recommendations.

CICS Web support behavior in compliance with HTTP/1.1

CICS Web support complies on your behalf with many of the requirements in the HTTP/1.1 specification. Most of the behavior described here applies for CICS as an HTTP server, but there are a few exceptions.

New behavior with CICS TS for z/VSE Version 2 Release 2

- **CICS checks inbound messages for compliance with HTTP/1.1, and handles or rejects non-compliant messages.** The checks are made immediately on receipt of the request, before an analyzer program is involved. A variety of basic acceptance checks are made, and if the message is unacceptable and it is not appropriate for CICS to handle the problem itself, an error response is returned to the Web client where possible. These basic acceptance checks are not carried out for HTTP/1.0 messages, nor are they carried out if the USER protocol (instead of the HTTP protocol) is specified on the TCPIP SERVICE definition.

Note: CICS requires the Content-Length header on all inbound HTTP/1.1 messages that have a message body. If a message body is present but the header is not provided, or its value is inaccurate, the socket receive for the faulty message or for a subsequent message can produce unpredictable results. For HTTP/1.0 messages that have a message body, the Content-Length header is optional.

- **CICS follows the HTTP/1.1 rules for comparison of URLs.** Scheme names and host names are compared case-insensitively, but paths are compared case-sensitively. All components are unescaped before comparison. CICS also handles the absolute URI form in requests (where the host name is specified in the request line). Note that when you use an analyzer program to handle an inbound HTTP request, if you need to achieve compliance in this respect, you must code your analyzer program to perform URL comparison according to the rules stated in the HTTP/1.1 specification.
- **CICS provides a suitable HTTP version number in the start line of outbound messages.** CICS normally identifies the message as HTTP/1.1, unless it knows that the Web client is at HTTP/1.0 level. In that case, CICS identifies the message as HTTP/1.0. Requests by a Web client to upgrade from one HTTP version to another, or to a different security protocol, are not supported by CICS.
- **On outbound HTTP/1.1 messages, CICS supplies the HTTP headers that should normally be present for the message to be compliant with HTTP/1.1.** Some headers are also produced by CICS which are not required for compliance, but support actions that the application program has requested (such as the Expect header). [“HTTP header reference for CICS Web support” on page 105](#) describes the headers that CICS writes and the circumstances in which these headers are created. The headers for compliance are supplied for messages sent by both Web-aware applications and non-Web-aware applications. They are not supplied if the USER protocol (instead of the HTTP protocol) is specified for the TCPIP SERVICE definition. For HTTP/1.0 messages, only the Connection: Keep-Alive, Content-Length, Date and Server headers are supplied.
- **CICS takes action on the Expect header on an inbound requests.** When CICS as an HTTP server receives a request with the Expect header, it sends a 100-Continue response to Web client and waits for the remainder of the request.
- **CICS handles OPTIONS requests from Web clients and makes an appropriate response.** OPTIONS * (an inquiry on the whole server, rather than a specific resource) is the only format accepted. The response contains basic information about CICS as an HTTP server (the HTTP version and server software description). No user application program is involved.
- **CICS handles TRACE requests from Web clients and makes an appropriate response.** When CICS Web support receives a correctly formed request with the TRACE method, it returns a response containing the request with its original headers, plus any headers it acquired (such as the Via header). No user application program is involved.
- **CICS accepts inbound messages with chunked transfer-coding and assembles them for you, and supports your use of chunked transfer-coding to send outbound messages.** Trailing headers for chunked messages can be manipulated through the HTTP header read, write and browse commands. This means that applications can receive and handle chunked messages as they would normal messages. CICS also supports sending chunked messages out from a user application, but you must ensure that you follow the correct procedure to make the chunked message acceptable to the recipient.

[“How CICS Web support handles chunked transfer-coding”](#) on page 51 explains CICS Web support behavior in this respect.

- **CICS supports pipelining.** CICS lets you receive pipelined requests from a Web client. CICS passes each request to the application program in turn, to ensure that the application complies with HTTP/1.1 by responding to the requests in the order they were received. [“How CICS Web support handles pipelining”](#) on page 52 explains CICS Web support behavior in this respect.

Changed behavior, compared to CICS TS Version 2 Release 1

- **Connections are now persistent by default.** If CICS is communicating with a Web client at HTTP/1.1 level, it keeps connections open unless the Web client or the user application in CICS specifically request closure, or the task ends. If CICS is communicating with a Web client at HTTP/1.0 level, it sends Connection: Keep-Alive headers when a persistent connection is supported. [“How CICS Web support handles persistent connections”](#) on page 52 explains CICS Web support behavior in this respect, and how this differs from earlier CICS releases.

- **CICS handles a wider range of error situations and unsupported messages.**

CICS Web support is designed to react to many error situations, or situations where an inbound message might cause problems for a user application. A wider range of error responses are provided, suited to the requirements of HTTP/1.1 or HTTP/1.0 clients. [“HTTP status code reference for CICS Web support”](#) on page 108 explains the situations in which CICS provides a response to a Web client. The user-replaceable Web error programs can be tailored to modify CICS-supplied responses. See the [CICS Internet Guide](#) for more information.

HTTP functions not supported by CICS Web support

The HTTP/1.1 specification defines various roles for the parties that make use of the HTTP protocol. CICS Web support carries out many of the functions that are appropriate for an origin server, for a client, and for a user agent (although a human user might not be involved for every HTTP client request). The HTTP/1.1 specification also includes requirements that relate to proxies, gateways, tunnels and caches, and these requirements are not relevant to CICS Web support and can be ignored.

- **CICS does not act as a proxy.** You can ignore all the requirements in the HTTP/1.1 specification that relate to the behavior of proxies.
- **CICS does not act as a gateway (an intermediary for another server) or a tunnel (a relay between HTTP connections).** You can ignore all the requirements in the HTTP/1.1 specification that relate to the behavior of gateways and tunnels.
- **CICS does not provide caching facilities, or provide support for user-written caching facilities.** You can ignore all the requirements in the HTTP/1.1 specification that relate to the behavior of caches. Although you may store any information you receive from a server, you should be careful that you do not deliver the stored information to a user who is making a request in the expectation of receiving current information from the server.
- **CICS is not designed for use as a Web browser.** CICS does not provide history lists, lists of favorites or other features of a Web browser, so any requirements relating to these can be ignored.

How CICS Web support handles chunked transfer-coding

Messages using chunked transfer-coding can be sent and received by CICS.

CICS as an HTTP server can receive a chunked message as a request, or send one as a response.

- When CICS as an HTTP server receives a chunked message as an HTTP request, CICS Web support recognizes the chunked encoding. It waits until all the chunks are received (indicated by the receipt of a chunk with zero length), and assembles the chunks to form a complete message. The assembled message body can be received by a user application program using the WEB RECEIVE command.
 - You can limit the total amount of data that CICS accepts for a single chunked message, using the MAXDATALEN option on the TCPIP SERVICE resource definition that relates to the port on which the request arrives.

- The timeout value for receiving a chunked message is set by the SOCKETCLOSE attribute of the TCPIP SERVICE definition.
- Trailing headers from the chunked message can be read using the HTTP header commands. The Trailer header identifies the names of the headers that were present as trailing headers. If you are using an analyzer program in the processing path for the request, note that trailing headers are not passed to the analyzer program along with the main request headers.
- When CICS sends a chunked message, the application program can specify chunked transfer-coding by using the CHUNKING(CHUNKYES) option on the WEB SEND command for each chunk of the message. The message can be divided up in whatever way is most convenient for the application program. CICS sends each chunk of the message, adding appropriate HTTP headers to indicate to the recipient that chunked transfer-coding is being used. The application program issues WEB SEND with CHUNKING(CHUNKEND), to indicate the end of the message. CICS then sends an empty chunk (containing a blank line) to end the chunked message, along with any trailing headers that are wanted.

“Using chunked transfer-coding to send an HTTP request or response” on page 53 explains the process to use for chunked transfer-coding when sending an HTTP message from CICS. This procedure should be followed in order for your chunked message to be acceptable to the recipient.

How CICS Web support handles pipelining

A pipelined request sequence can be received by CICS.

CICS as an HTTP server can receive a pipelined request sequence from a Web client.

When CICS receives a pipelined sequence of HTTP requests, the requests are processed serially. This is to ensure that the responses are returned in the same order that the requests were sent. CICS treats each message in the pipelined sequence as a separate transaction, passing the message to an application program and waiting for the application program to produce a response. Each transaction handles a single request and provides a response. The remaining requests in the pipelined message sequence are held by CICS until the response to the previous request is sent, and then a new transaction is initiated to process each further request.

How CICS Web support handles persistent connections

Persistent connections are the default behavior for CICS Web support.

Before CICS TS 2.2, the connection behavior was that CICS would normally close the connection when data had been received from the Web client, unless the Web client sent a Connection: Keep-Alive header. Now, when a connection is made between a Web client and CICS as an HTTP server, CICS attempts to keep the connection open by default.

When CICS is the HTTP server, the persistent connection is closed if:

- The user-written application that is handling the request closes the connection (by specifying the CLOSESTATUS(CLOSE) option on the WEB SEND command).
- The Web client closes the connection (notified by a Connection: close header).
- The Web client is an HTTP/1.0 client that does not send a Connection: Keep-Alive header.
- The timeout period is reached (indicating that the connection has failed, or that the Web client has deliberately exited the connection).

Otherwise, CICS leaves the persistent connection open for the Web client to send further requests. If there is a persistent connection with the client, CICS keeps the connection open after an error response is sent through a Web error program. The exception is where CICS selects the 501 (Method Not Implemented) status code for the response, in which case the connection is closed by CICS.

With a TCPIP SERVICE resource definition for CICS Web support, the SOCKETCLOSE attribute of the TCPIP SERVICE definition should not be specified as zero. A zero setting for SOCKETCLOSE means that CICS as an HTTP server closes the connection immediately after receiving data from the Web client, unless further data is waiting. This means that persistent connections cannot be maintained.

Using chunked transfer-coding to send an HTTP request or response

This topic tells you how to set up chunked transfer-coding for an HTTP response from CICS as an HTTP server.

Before setting up chunked transfer-coding, you need to plan the following attributes of the item that you want to send:

1. The HTTP headers that should be used at the beginning of the message. CICS supplies its usual message headers, which are listed in [“HTTP header reference for CICS Web support”](#) on page 105. For a chunked message, CICS supplies the proper headers for chunked transfer-coding, including the Transfer-Encoding: chunked header. If any additional headers are required at the beginning of the message, the application can write them before the first WEB SEND command.
2. Any headers that should be sent in the trailer at the end of the message. These headers are known as trailing headers. Note that the HTTP/1.1 specification sets requirements for the use of trailing headers, including that it should not matter if the recipient ignores them.
3. How the message should be divided up. This can be done in whatever way is most convenient for the application program. For example, the output from a number of other application programs could be sent as it is produced, or data from each row of a table could be read and sent individually.
4. The length of each chunk of data that will be sent. Do not include the length of any trailing headers.

The procedure described in this topic enables you to create a correctly constructed chunked message, as defined in the HTTP/1.1 specification. If the chunked message is not correctly constructed, the recipient may discard it.

The body of a chunked message cannot be formed directly from CICS documents (so the DOCTOKEN option cannot be used). The FROM option must be used to specify data to form the body of a chunked message.

When you have begun sending the parts of a chunked message, you cannot send any different messages or receive any items, until the final empty chunk is sent and the chunked message is complete.

1. Before beginning a chunked message, verify that the Web client is at HTTP/1.1 version. All HTTP/1.1 applications are required to understand chunked transfer-coding. A chunked message cannot be sent to an HTTP/1.0 recipient. Use the WEB EXTRACT command to check the HTTP version specified for the Web client's request.
2. Use the WRITE HTTPHEADER command as many times as necessary to write any HTTP headers that should be sent *before* the body of the message. Do not write the headers for chunked transfer-coding; CICS writes these itself, using the chunk length information supplied by the application program.
3. If you want to include trailing headers (headers sent out *after* the body of the message) with the chunked message, use the WRITE HTTPHEADER command to write a Trailer header. Specify the names of all the HTTP headers you plan to send in the trailer, as the value of the Trailer header. You may send any headers as trailing headers, except the Transfer-Encoding, Trailer and Content-Length headers. You need to ensure that the Web client sent a TE: trailers header on its request. This header shows that the client understands trailing headers. CICS returns an INVREQ response with a RESP2 value of 6 to the WRITE HTTPHEADER command if you attempt to write the Trailer header when the client did not send TE: trailers. Alternatively, you can use the READ HTTPHEADER command to check for the presence of the TE: trailers header. The trailing headers themselves are written during the chunked sending process.
4. Use the WEB SEND command to send the first chunk of the message.
 - a. Specify CHUNKING(CHUNKYES) to tell CICS that this is a chunk of a message.
 - b. Use the FROM option to specify the first chunk of data from the body of the message.
 - c. Use the FROMLENGTH option to specify the length of the chunk.

- d. Specify any other options that apply to both chunked and non-chunked messages, as given in our main set of instructions. For example, if this chunked message is the final message that you want to send to this Web client, specify the CLOSESTATUS(CLOSE) option.
5. Use the WEB SEND command as many times as necessary to send each of the remaining chunks of the message. On each WEB SEND command, just specify the following items:
 - a. CHUNKING(CHUNKYES).
 - b. The FROM option, giving the chunk of data.
 - c. The FROMLENGTH option, giving the length of the chunk.

Do not specify any of the other options for the command. CICS sends each chunk as you issue the command.

6. Optional: At any time after issuing the WEB SEND command for the first chunk, but before issuing the WEB SEND command for the final empty chunk (see the next step), use the WRITE HTTPHEADER command to create further HTTP headers that should be sent as trailing headers. Provided that a Trailer header was written on the first chunk of the message, the HTTP headers written during the chunked sending process are treated by CICS as trailing headers, and they are sent out with the final empty chunk. (If the Trailer header was not written, CICS does not allow any trailing headers to be written.) Note that CICS does not check whether your trailer headers match the names that you specified in the initial Trailer header on the first chunk of the message.
7. When you have sent the last chunk of the data, specify a further WEB SEND command with CHUNKING(CHUNKEND) and no FROM or FROMLENGTH option. CICS then generates and sends an empty chunk to the recipient to end the chunked message. The empty chunk is sent along with the trailer containing any trailing headers that you wrote.
8. Errors are handled as follows:
 - a. If one of the WEB SEND commands fails during the sequence, an error response is returned, and subsequent sends will also fail. The application should handle this situation appropriately.
 - b. If all of the chunks are sent successfully but the application does not issue the final WEB SEND command with CHUNKING(CHUNKEND), the transaction is abended with abend code AWBP. This is necessary because CICS cannot guarantee that the chunked message is complete and correct, and so cannot issue the final empty chunk on behalf of the application.

An incomplete chunked message should be ignored and discarded by the recipient. The Web client will decide whether or not to retry the request.

Application programming for non-HTTP requests

Application programs for non-HTTP requests can use certain elements of the EXEC CICS WEB programming interface. They may also be non-Web-aware applications, and produce output that is encoded by a converter program. A pseudoconversational programming model is not suitable for non-HTTP requests. Applications should be designed to receive a single request and provide a single response.

Web-aware applications

If you want to use a Web-aware application to respond to non-HTTP requests, you can use the following CICS API commands:

- The WEB RECEIVE command can be used to receive a non-HTTP request. If the application is used to respond to both HTTP and non-HTTP requests, you can use the TYPE option on the WEB RECEIVE command to distinguish between the two request types. Note that CICS does not carry out any parsing for a non-HTTP message, and requests that have been divided up for transmission across the network are not automatically assembled. If an analyzer program assembles the request, the results of this are not visible to the CICS WEB API commands.
- The EXEC CICS DOCUMENT commands can be used to compose a CICS document to form the body of a response.

- The WEB SEND command can be used to send a response to a non-HTTP client. However, the following options that relate to HTTP-specific actions are not suitable:
 - STATUSCODE and STATUSTEXT. If specified, these are ignored.
 - CLOSESTATUS. If specified, this is ignored.
 - CHUNKING. If specified, this causes an error on the command.
- The WEB RETRIEVE command can be used to retrieve a CICS document sent in an earlier EXEC CICS WEB SEND command.

Other EXEC CICS WEB commands relate to HTTP requests only, and can result in an INVREQ condition if used with non-HTTP requests. An application program can specify code page conversion of non-HTTP requests using the WEB RECEIVE command.

Non-Web-aware applications with converter programs

With non-Web-aware applications, you can use a converter program to convert the input from the Web client into a suitable COMMAREA for the application, and to convert the output from the application into HTML to provide the response. If an analyzer program has reconstructed the request after it was divided up for transmission across the network, the results of this can be passed to a converter program.

The following input fields which relate to HTTP requests are undefined in a converter program for non-HTTP requests:

- The HTTP version
- The method
- The path component of the URL
- The request headers

Other changes

The communication area passed to the WEB error program DFHWBEP has been changed.

Changed field:

- `wbep_abend_code` (input)
 - The four character abend code associated with this exception.

New field:

- `wbep_close_conn` (output)
 - One byte field that indicates that the connection is to be closed after the response is sent to the client. „Y“ requests the connection to be closed.

BINARY(data-area)

specifies a buffer of data which is to be used as the contents of the new document being created. The data is copied unchanged to the document content and no attempt is made to parse the data for symbol substitution. The purpose of the BINARY option is to allow the application to insert blocks of data that must not undergo conversion to a client code page when the data is sent.

DELIMITER(data-value)

specifies an optional 1-byte value used to delimit symbol name-value pairs in the SYMBOLLIST buffer. If this option is not specified, the value defaults to an ampersand. There are several disallowed DELIMITER values, all of which cause an INVREQ condition if used. The disallowed values are:

- null (binary X'00')
- shift in (binary X'0E')
- shift out (binary X'0F')
- space (binary X'40')
- plus sign (binary X'4E')
- colon (binary X'7A')
- equals (binary X'7E')
- percent sign (binary X'6C')
- backslash (binary X'E0')

If this option is used, the application must ensure that the DELIMITER does not appear in any symbol value in the SYMBOLLIST buffer. For this reason, the application should not use alphanumeric and other printable characters as the DELIMITER value.

DOCSIZE(data-area)

specifies a binary fullword area that will be updated with the current size of the document in bytes. This is the maximum size of the buffer needed to contain a copy of the document when a RETRIEVE command is issued.

DOCTOKEN(data-area)

specifies a data area to contain the symbolic name of the document. The area must be 16 bytes in length and will be set to a CICS-generated name by which the document can be referred to in later commands.

FROM(data-area)

specifies that data supplied by the application is to be used to create the contents of the new document. The data content could be a template or a document which was created and retrieved. If the data is a template, symbol substitution will take place where the symbols exist in the symbol table. If the data is a previously retrieved document, the conversion and bookmark tags which were inserted during retrieval will be removed from the content and stored in the internal format required by the API commands. Note that symbol substitution will not be attempted on any unresolved symbols contained in a retrieved document.

FROMDOC(data-area)

specifies the symbolic name (see the **DOCTOKEN** option) of a document whose contents are to be copied to the new document being created. All bookmark and conversion tags are copied to the new document. The symbol table will be not be copied.

HOSTCODEPAGE(name)

specifies the name of the host codepage that the data being added is encoded in. This option applies to the TEXT, SYMBOL and TEMPLATE options only. The name must be eight characters long; if it is shorter than eight characters it must be padded on the right with blanks.

LENGTH(data-value)

specifies the length, as a fullword binary value, of the application supplied buffer.

LISTLENGTH(data-value)

specifies the length, as a fullword binary value, of the symbol list.

SYMBOLLIST(data-area)

specifies a buffer which contains a symbol list. A symbol list is a character string consisting of one or more symbol definitions separated by ampersands. Each symbol definition consists of a name, an equals sign, and a value.

The name is case-sensitive. It can contain only uppercase and lowercase letters, numbers, and underscores ("_"). Unlike the symbols in the template, the names in the symbol list have neither an ampersand at the beginning, nor a semicolon at the end. For example, the symbol &mytitle; in the template corresponds to the name *mytitle* in the symbol list. Symbols in the symbol list are separated by the & character. For example:

```
applid=IYCQ&jobname=test
```

The following restrictions apply to the characters allowed in the value in the symbol list:

- A percent sign may be followed by two hexadecimal digits. When the value is put into the symbol table, the percent sign and the two hexadecimal digits following it are interpreted as the EBCDIC equivalent of the single ASCII character denoted by the two digits. If you want a percent sign in the value in the symbol table, you may put %25 in the symbol list. If the characters following the percent sign are not two valid hexadecimal digits, the percent sign and the following characters are put into the symbol table as they appear in the symbol list. Note that this restriction does not apply if the UNESCAPED keyword has been specified.
- An ampersand is not allowed. If you want an ampersand in the value in the symbol table, you must put %26 in the value in the symbol list. Note that this restriction does not apply if the DELIMITER keyword is used to specify a delimiter other than ampersand.
- When the value is put into the symbol table, a plus sign is interpreted as a space. If you want a plus sign in the value in the symbol table, you must put %2B in the value in the symbol list. Note that this restriction does not apply if the UNESCAPED keyword has been specified.
- If you want a space in the value in the symbol table, the value in your symbol list may contain a space, a plus sign, or %20. Note that this restriction does not apply if the UNESCAPED keyword has been specified.

TEMPLATE(name)

specifies the 48-byte name of a template. The template must be defined to CICS using RDO. If the name is shorter than 48 bytes, it must be padded on the right with blanks.

TEXT(data-area)

specifies a buffer of data which is to be used as the contents of the new document being created. The data is copied unchanged to the document content and no attempt is made to parse the data for symbol substitution. The data will be marked as requiring conversion to the client code page when the document is sent.

UNESCAPED

prevents CICS from unescaping symbol values contained in the SYMBOLLIST buffer. If this option is used, plus signs are not converted to spaces, and sequences such %2B are not converted to single byte values.

Conditions**INVREQ**

RESP2 values are:

1

The retrieved document specified on the FROM option is not in a valid RETRIEVE format.

8

The value specified for DELIMITER is not valid.

NOTFND

RESP2 values:

DOCUMENT INSERT

2

The document specified on the FROMDOC option could not be found or was named incorrectly.

3

The template specified on the TEMPLATE option could not be found, or was named incorrectly, or was in an invalid file format.

7

The host codepage specified on the HOSTCODEPAGE option could not be found or was named incorrectly.

SYMBOLERR

a symbol specified in the symbol list does not conform to the naming rules for symbols. RESP2 contains the offset of the symbol in the list.

TEMPLATERR

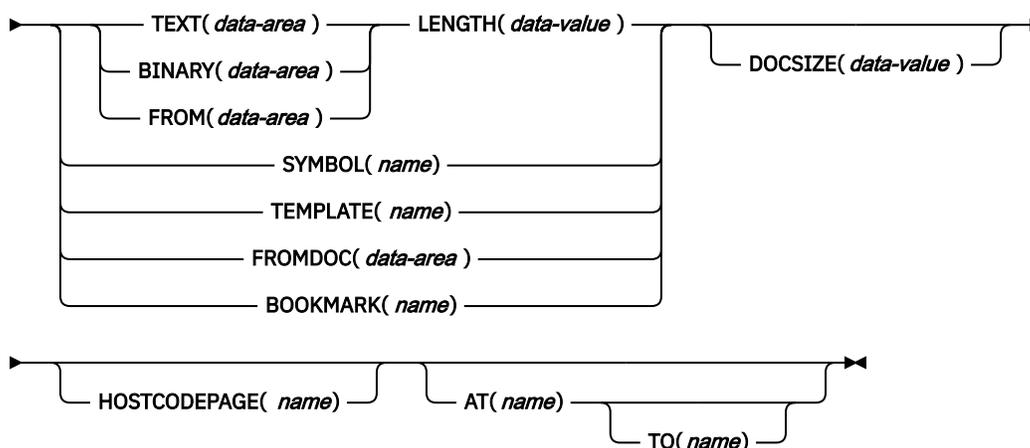
an invalid #set, #include or #echo command has been encountered while processing the supplied template data. RESP2 contains the offset of the invalid command. This condition can also occur for unmatched Shift-Out Shift-In pairs.

DOCUMENT INSERT

Insert document objects.

DOCUMENT INSERT

➔ DOCUMENT — INSERT — DOCTOKEN(*data-area*) ➔



Conditions: DUPREC, INVREQ, NOTFND, TEMPLATERR

Description

DOCUMENT INSERT allows the application to insert document objects at insertion points within the document. The insertion points (bookmarks) define relative positions within the document. Bookmarks must be defined before being referenced. Data is always inserted after the position identified by the bookmark.

Options

AT(*name*)

specifies the 16-byte symbolic name of a bookmark which identifies the position of the insertion point in the document. Data is inserted after the bookmark, and any data following the bookmark is shifted down. The application can use a combination of the AT and TO options to perform an overlay operation. If the AT operand is not specified, the data is inserted at the end of the document. A

pre-defined bookmark of TOP is provided to allow the application to insert data at the beginning of the document.

BINARY(*data-area*)

specifies a buffer of data to be inserted into the document. The data is copied unchanged to the insertion point in the document, and no attempt is made to parse the data for symbol substitution. The BINARY option allows the application to insert blocks of data that must not undergo conversion to a client code page when the data is sent.

BOOKMARK(*name*)

specifies a bookmark to be inserted into the document. A bookmark is a symbolic name which identifies an insertion point in the document. The name can be up to 16 characters in length, and must not contain any imbedded spaces.

DOCSIZE(*data-value*)

specifies a binary fullword area to be updated with the current size of the document in bytes. This is the maximum size of the buffer needed to contain a copy of the document when a RETRIEVE command is issued.

DOCTOKEN(*data-area*)

specifies the 16-byte symbolic name of the document into which data is to be inserted.

FROM(*data-area*)

specifies that a buffer of data supplied by the application is to be inserted into the document. The data content can be a template or a document that was previously created and retrieved. If the data is a template, symbol substitution takes place where the symbols exist in the symbol table. If the data is a previously retrieved document, the conversion and bookmark tags which were inserted during the retrieval will be removed from the content and stored in the internal form required by the API commands. Note that symbol substitution will not be attempted on any unresolved symbols contained in a retrieved document.

FROMDOC(*data-area*)

specifies the symbolic name of a document (see the DOCTOKEN option) whose contents are copied to the insertion point of the target document. All bookmarks and conversion tags are copied to the target document. The symbol table is not copied.

HOSTCODEPAGE(*name*)

specifies the symbolic name (see the DOCTOKEN option) of the host codepage that the data being added is encoded in. This option applies to the TEXT, SYMBOL and TEMPLATE options only. The name must be eight characters long; if it is shorter than eight characters, it must be padded on the right with blanks.

LENGTH(*data-value*)

specifies the length, as a fullword binary value, of the buffer containing the TEXT, BINARY or FROM data.

When the DOCUMENT INSERT command follows a DOCUMENT RETRIEVE command, without the use of the DATAONLY option, and the retrieved document is being inserted using the FROM option, the LENGTH specified must be equal to the length of the retrieved document.

SYMBOL(*name*)

specifies the 32-byte name of a symbol in the symbol table. The data associated with the symbol in the symbol table is inserted, but not the symbol itself.

TEMPLATE(*name*)

specifies the 48-byte name of a template. The template must be defined to CICS using RDO. If the name is less than 48 bytes, it must be padded on the right with blanks.

TEXT(*data-area*)

specifies a buffer of data to be inserted into the document. The data is copied unchanged to the insertion point in the document, and no attempt is made to parse the data for symbol substitution. When the document is sent, it is marked as requiring conversion to the client code page.

TO(*name*)

specifies the symbolic name of a bookmark identifying the end position of an overlay operation. Data between the bookmarks identified by the AT and TO operands is deleted, and new data is inserted in

its place. It is possible to delete data between two bookmarks by specifying a null string on the TEXT or BINARY option with a LENGTH of zero.

Conditions

DUPREC

the bookmark has already been defined.

INVREQ

RESP2 values are:

0

The bookmark specified on the TO option appears before the bookmark specified on the AT bookmark.

1

The retrieved document specified on the FROM option is not in a valid RETRIEVE format.

2

The bookmark name on the BOOKMARK option is invalid.

NOTFND

one of the following documents or templates could not be found, or its name was incorrect.

RESP2 values:

1

The document specified on the DOCUMENT option.

2

The document specified on the FROMDOC option.

3

The template specified on the TEMPLATE option.

4

The document specified on the SYMBOL option.

5

The document specified on the AT option.

6

The document specified on the TO option.

7

The document specified on the HOSTCODEPAGE option.

TEMPLATERR

an invalid #set, #include or #echo command has been encountered while processing the supplied template data. RESP2 contains either a zero (if the maximum of 32 levels of embedded templates is exceeded), or the offset of the invalid command.

DOCUMENT RETRIEVE

Copy a document into the application's own buffer.

DOCUMENT RETRIEVE

➔ DOCUMENT — RETRIEVE — DOCTOKEN(*data-area*) — INTO(*data-area*) — LENGTH(*data-value*) ➔



Conditions: LENGERR, NOTFND

Description

DOCUMENT RETRIEVE allows the application to obtain a copy of the document in its own buffer, which it can then manipulate directly. The document is managed by CICS, and the application does not have direct access to the buffer containing the contents of the document. The document exists only for the duration of the current transaction, so the application must retrieve the document and store it if the document is to exist over transaction boundaries. The retrieved document can be used as a basis for a new document by using the FROM option of the DOCUMENT CREATE command.

When the document is retrieved, CICS inserts tags into the document contents to identify the bookmarks and to delimit the blocks that do not require codepage conversion. To request a copy without tags, specify DATAONLY. The extracted document can also be converted into a single client codepage by using the CLNTCODEPAGE option.

Options

CLNTCODEPAGE(*name*)

specifies the name of the client codepage to which the data should be converted. The name can be up to 40 characters in length; if it is shorter than 40 characters, it must be padded on the right with blanks.

DATAONLY

specifies that the data should be retrieved without any imbedded tags.

DOCTOKEN(*data-area*)

specifies the 16-byte symbolic name of the document to be retrieved.

INTO(*data-area*)

specifies the buffer that is to contain the copy of the document content.

LENGTH(*data-value*)

specifies the length, as a fullword binary value, of the amount of data being returned to the application.

MAXLENGTH(*data-value*)

specifies the length, as a fullword binary value, of the maximum amount of data the buffer can receive.

Conditions

LENGERR

RESP2 values:

- 1** MAXLENGTH is less than or equal to zero. The document is truncated.
- 2** The length of the receiving buffer is zero, or is too short to contain the document contents. The document is truncated.

NOTFND

RESP2 values:

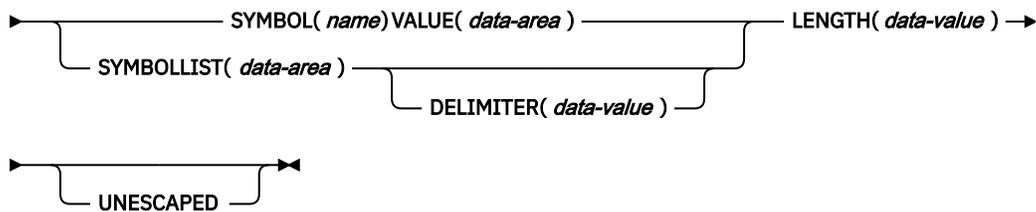
- 1** The document has not been created, or the name is incorrectly specified.
- 7** The specified client codepage can not be found.

DOCUMENT SET

Add symbols and values to symbol table.

DOCUMENT SET

►► DOCUMENT — SET — DOCTOKEN(*data-area*) →



Conditions: NOTFND, SYMBOLERR

Description

DOCUMENT SET allows the application to add symbols and their associated values to the symbol table. If the symbol being added already exists in the table, it is replaced by the new definition.

Options**DELIMITER(*data-value*)**

specifies an optional 1-byte value used to delimit symbol name-value pairs in the SYMBOLLIST buffer. If this option is not specified, the value defaults to an ampersand. There are several disallowed DELIMITER values, all of which cause an INVREQ condition if used. The disallowed values are:

- null (binary X'00')
- shift in (binary X'0E')
- shift out (binary X'0F')
- space (binary X'40')
- plus sign (binary X'4E')
- colon (binary X'7A')
- equals (binary X'7E')
- percent sign (binary X'6C')
- backslash (binary X'E0')

If this option is used, the application must ensure that the DELIMITER does not appear in any symbol value in the SYMBOLLIST buffer. For this reason, the application should not use alphanumeric and other printable characters as the DELIMITER value.

DOCTOKEN(*data-area*)

specifies the 16-byte symbolic name of the document that owns the symbol table.

LENGTH(*data-value*)

specifies the length, as a fullword binary value, of the buffer containing the data value associated with the symbol or the length of the buffer containing the symbol list when the SYMBOLLIST option is used.

SYMBOL(*name*)

specifies the name of the symbol that is to be added to the table. The name can be 1 to 32 characters in length with no embedded spaces.

SYMBOLLIST(*data-area*)

specifies a buffer which contains a symbol list. A symbol list is a character string consisting of one or more symbol definitions separated by ampersands. Each symbol definition consists of a name, an equals sign, and a value.

The name is case-sensitive. It can contain only uppercase and lowercase letters, numbers, and underscores ("_"). Unlike the symbols in the template, the names in the symbol list have neither an ampersand at the beginning, nor a semicolon at the end (for example, the symbol :&mytitle; in the

template corresponds to the name *mytitle* in the symbol list). Symbols in the symbol list are separated by the & character. For example:

```
applid=IYCQ&jobname=test
```

The following restrictions apply to the characters allowed in the value in the symbol list:

- A percent sign may be followed by two hexadecimal digits. When the value is put into the symbol table, the percent sign and the two hexadecimal digits following it are interpreted as the EBCDIC equivalent of the single ASCII character denoted by the two digits. If you want a percent sign in the value in the symbol table, you may put %25 in the symbol list. If the characters following the percent sign are not two valid hexadecimal digits, the percent sign and the following characters are put into the symbol table as they appear in the symbol list. Note that this restriction does not apply if the UNESCAPED keyword has been specified.
- An ampersand is not allowed. If you want an ampersand in the value in the symbol table, you must put %26 in the value in the symbol list. Note that this restriction does not apply if the DELIMITER keyword is used to specify a delimiter other than ampersand.
- When the value is put into the symbol table, a plus sign is interpreted as a space. If you want a plus sign in the value in the symbol table, you must put %2B in the value in the symbol list. Note that this restriction does not apply if the UNESCAPED keyword has been specified.
- If you want a space in the value in the symbol table, the value in your symbol list may contain a space, a plus sign, or %20. Note that this restriction does not apply if the UNESCAPED keyword has been specified.

UNESCAPED

prevents CICS from unescaping symbol values contained in the SYMBOLLIST buffer. If this option is used, plus signs are not converted to spaces, and sequences such as %2B are not converted to single byte values.

VALUE(data-area)

specifies an area containing the value to be associated with the SYMBOL.

Conditions

INVREQ

RESP2 values:

8

The value specified for DELIMITER is not valid.

NOTFND

RESP2 values:

1

The document has not been created or the name is incorrectly specified.

SYMBOLERR

a symbol name is invalid. RESP2 values:

0

SYMBOLLIST was not used.

offset

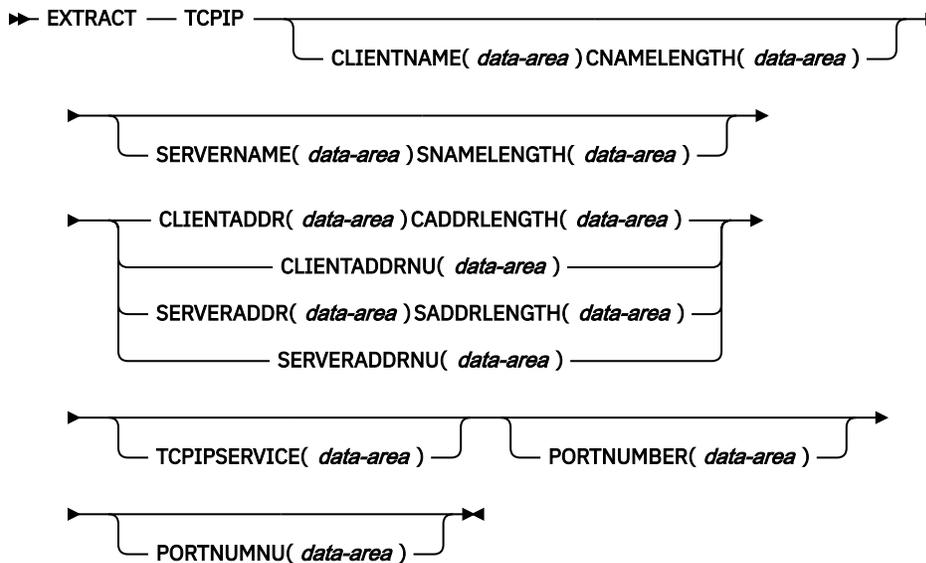
RESP2 contains the offset of the invalid symbol in the list.

Chapter 15. EXEC CICS EXTRACT

EXTRACT TCPIP

Obtain information about TCPIP characteristics of the current transaction.

EXTRACT TCPIP



Conditions: INVREQ, LENGERR

Options

CADDRLENGTH(data-area)

specifies the length of the buffer supplied on the CLIENTADDR option, and is set to the actual length of the data returned to the application. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

CLIENTADDR(data-area)

specifies a buffer to contain the client's TCP/IP address.

CLIENTADDRNU(data-area)

specifies a fullword binary field containing the client's TCP/IP address in binary form.

CLIENTNAME(data-area)

specifies a buffer to contain the client's name as known by the Domain Name Server.

CNAMELENGTH(data-area)

specifies the length of the buffer supplied on the CLIENTNAME option, and is set to the actual length of the data returned to the application. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

PORTNUMBER(data-area)

Specifies a 5-character field to contain the port number associated with this transaction in character form. This is the port on which the incoming data that initiated this transaction was received.

EXTRACT CERTIFICATE

PORTNUMNU(data-area)

Fullword field to contain the port number associated with this transaction in binary form. This is the port on which the incoming data that initiated this transaction was received.

SADDRENGTH(data-area)

specifies the length of the buffer supplied on the SERVERADDR option, and is set to the actual length of the data returned to the application. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

SERVERADDR(data-area)

specifies a buffer to contain the server's TCP/IP address in dotted decimal character form (nnn.nnn.nnn.nnn).

SERVERADDRNU(data-area)

specifies a fullword binary field containing the server's TCP/IP address in binary form.

SERVERNAME(data-area)

specifies a buffer to contain the server's name as known by the Domain Name Server.

SNAMELENGTH(data-area)

specifies the length of the buffer supplied on the SERVERNAME option, and is set to the actual length of the data returned to the application. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

TCPIPSERVICE(data-area)

An 8-byte field to contain the name of the TCPIPSERVICE associated with this transaction.

Conditions

INVREQ

RESP2 values:

2

An invalid socket response was received.

5

The command was issued from a non-TCP/IP application.

LENGERR

RESP2 values:

3

CLIENTADDR is too small to contain the string extracted.

4

SERVERADDR is too small to contain the string extracted.

6

CLIENTNAME is too small to contain the string extracted.

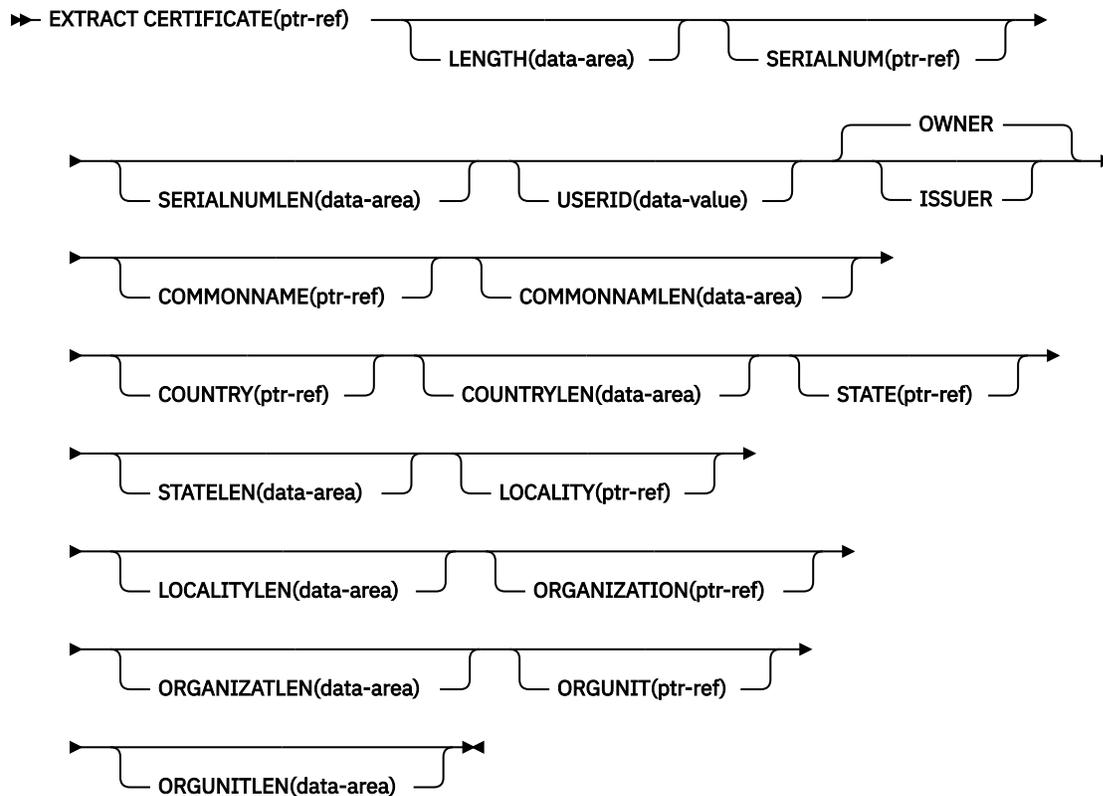
7

SERVERNAME is too small to contain the string extracted.

EXTRACT CERTIFICATE

Obtain information from the client certificate. The client certificate was received over a TCP/IP service that specified client authentication.

Syntax



Conditions: INVREQ, LENGERR

Description

EXTRACT CERTIFICATE allows the application to obtain information from the X.509 certificate that was received from a client during a Secure Sockets Layer (SSL) handshake over a TCPIP SERVICE that specified SSL(CLIENAUTH). The certificate contains fields that identify the owner (or subject) of the certificate, and fields that identify the Certificate Authority that issued the certificate. You can select the fields that you require by specifying the OWNER or ISSUER option. You cannot retrieve both OWNER and ISSUER fields with one command.

Options

CERTIFICATE(ptr-ref)

specifies a pointer reference to be set to the address of the full binary certificate received from the client. The pointer reference is valid until the next CICS command or the end of task.

COMMONNAME(ptr-ref)

specifies a pointer reference to be set to the common name from the client certificate. The pointer reference is valid until the next CICS command or the end of task.

COMMONNAMLEN(data-area)

specifies a fullword binary data area to be set to the length of the common name from the client certificate.

COUNTRY(ptr-ref)

specifies a pointer reference to be set to the address of the country from the client certificate. The pointer reference is valid until the next CICS command or the end of task.

COUNTRYLEN(data-area)

specifies a halfword binary data area to be set to the length of the country from the client certificate.

ISSUER

indicates that the values returned by this command refer to the Certificate Authority that issued this certificate.

LENGTH(data-area)

specifies a fullword binary data area to be set to the length of the body of the client certificate.

LOCALITY(ptr-ref)

specifies a pointer reference to be set to the address of the locality from the client certificate. The pointer reference is valid until the next CICS command or the end of task.

LOCALITYLEN(data-area)

specifies a halfword binary data area to be set to the length of the locality from the client certificate.

ORGANIZATION(ptr-ref)

specifies a pointer reference to be set to the address of the organization from the client certificate. The pointer reference is valid until the next CICS command or the end of task.

ORGANIZATLEN(data-area)

specifies a halfword binary data area to be set to the length of the organization from the client certificate.

ORGUNIT(ptr-ref)

specifies a pointer reference to be set to the address of the organization unit from the client certificate. The pointer reference is valid until the next CICS command or the end of task.

ORGUNITLEN(data-area)

specifies a halfword binary data area to be set to the length of the organization unit from the client certificate.

OWNER

indicates that the values returned by this command refer to the owner of the certificate.

SERIALNUM(ptr-ref)

specifies a pointer reference to be set to the address of the serial number of the certificate assigned by the certificate issuer. The pointer reference is valid until the next CICS command or the end of task.

SERIALNUMLEN(data-area)

specifies a halfword binary data area to be set to the length of the serial number.

STATE(ptr-ref)

specifies a pointer reference to be set to the address of the state or province from the client certificate. The pointer reference is valid until the next CICS command or the end of task.

STATELEN(data-area)

specifies a halfword binary data area to be set to the length of the state or province from the client certificate.

USERID(data-value)

specifies a pointer reference to be set to the userid connected with the client certificate. The pointer reference is valid until the next CICS command or the end of task.

Conditions

INVREQ

occurs for the following conditions:

- the command is being issued in a non-CICS Web Interface application.
- the command is being issued for a non-HTTP request.
- if an error occurs retrieving the certificate data from CICS intermediate storage.

LENGERR

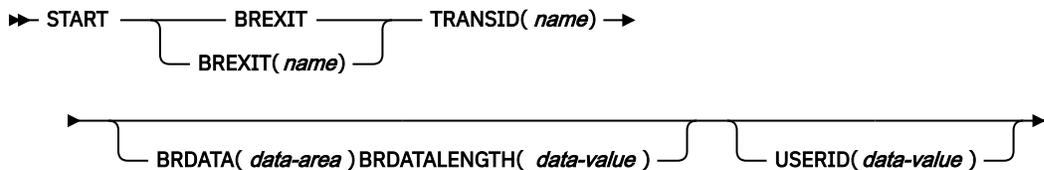
the string being extracted is longer than the length specified for one of the options.

Chapter 16. EXEC CICS START

START BREXIT

Start a task in the 3270 bridge environment and associate it with the named bridge exit.

START BREXIT



Conditions: INVREQ, LENGERR, NOTAUTH, PGMIDERR, TRANSIDERR, USERIDERR

Description

START BREXIT starts a task immediately in the local CICS region, and initializes the specified transaction (TRANSID) and bridge exit (BREXIT).

In the 3270 bridge environment, all 3270 terminal requests issued by the transaction specified by TRANSID, are intercepted and passed to the user-replaceable program (the bridge exit) specified by BREXIT.

The bridge exit (BREXIT) emulates the 3270 interface by passing the terminal requests to a client application that may be executing inside or outside of CICS.

See the [CICS External Interfaces Guide](#) for more information about the 3270 bridge and its interfaces.

The attached task cannot be CANCELLED; its STARTCODE is defined by the bridge exit.

Options

BREXIT(name)

specifies the name (1-8 characters) of the bridge exit to be associated with the started task. If no name is specified, the value of BREXIT on the TRANSACTION resource definition for TRANSID is used.

BRDATA(data-area)

specifies the data to be passed to the bridge exit specified by BREXIT when the task is started.

BRDATALENGTH(data-value)

specifies a fullword binary data value that is the length of the BRDATA to be passed to the bridge exit specified by BREXIT when the task is started.

TRANSID(name)

specifies the symbolic identifier (1–4 characters) of the transaction to be executed by a task started as the result of a START BREXIT command. The transaction will be started in the 3270 bridge environment, and will execute in association with the bridge exit specified in BREXIT.

USERID(data-value)

Specifies the userid under whose authority the started transaction is to run.

Conditions

INVREQ

RESP2 values:

11

An attempt was made to route a START BREXIT request.

12

A START BREXIT request has failed..

18

A USERID is specified and the CICS external security manager interface is not initialized.

Default action: terminate the task abnormally.

LENGERR

occurs if BRDATALENGTH is not greater than zero.

Default action: terminate the task abnormally.

NOTAUTH

RESP2 values:

7

A resource security check fails on TRANSID (name).

9

A surrogate user security check fails on USERID (name).

The security access capabilities of the transaction that issued the command do not allow the command to be performed with the value specified in the USERID option.

Default action: terminate the task abnormally.

PGMIDERR

occurs if no name is supplied by the BREXIT option and the transaction definition for TRANSID does not provide a default BREXIT name.

Default action: terminate the task abnormally.

TRANSIDERR

occurs if the transaction identifier specified in a START BREXIT command has not been defined to CICS.

Default action: terminate the task abnormally.

USERIDERR

RESP2 values:

8

The specified USERID is not known to the external security manager.

10

The external security manager is in a state such that CICS cannot determine whether a specified USERID is valid.

Default action: terminate the task abnormally.

Passing data to the bridge exit

Data can be passed to the bridge exit using the BRDATA and BRDATALENGTH options.

The following example shows how to start a specified task, in the 3270 bridge environment and pass data to its bridge exit:

```
EXEC CICS START BREXIT('DFH0CBRE')  
      TRANSID('TRNL')  
      BRDATA(BRSD)  
      BRDATALENGTH(72)  
      :
```


LOCKED

occurs when the request cannot be performed because use of the queue has been restricted owing to a unit of work failing in-doubt.

Default action: terminate the task abnormally.

NOTAUTH

occurs when a resource security check has failed on QUEUE(name).

Default action: terminate the task abnormally.

QIDERR

occurs when the specified queue cannot be found in either main or auxiliary storage.

Default action: terminate the task abnormally.

SYSIDERR

occurs in any of the following situations:

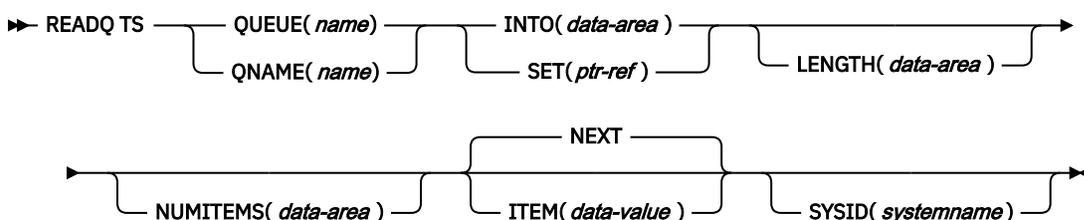
- When the SYSID option specifies a name that is neither the local system nor a remote system (made known to CICS by defining a CONNECTION)
- When the link to the remote system is closed

Default action: terminate the task abnormally.

READQ TS

Read data from temporary storage queue.

READQ TS



Conditions: INVREQ, IOERR, ISCINVREQ, ITEMERR, LENGERR, NOTAUTH, QIDERR, SYSIDERR

Note for dynamic transaction routing: Using this command could create inter-transaction affinities that adversely affect the use of dynamic transaction routing. See the [CICS Application Programming Guide](#) for more information about transaction affinities.

Description

READQ TS retrieves data from a temporary storage queue in main or auxiliary storage.

Options

INTO(*data-area*)

specifies the data area into which the data is to be written. The data area can be any variable, array, or structure.

ITEM(*data-value*)

provides a halfword binary value that specifies the item number of the logical record to be retrieved from the queue.

LENGTH(*data-area*)

specifies the length, as a halfword binary value, of the record to be read.

If you specify the INTO option, LENGTH need not be specified if the length can be generated by the compiler from the INTO variable.

If you specify INTO, LENGTH defines the maximum length of data that the program accepts. If the value specified is less than zero, zero is assumed. If the length of the data exceeds the value specified, the data is truncated to that value and the LENGERR condition occurs.

On completion of the retrieval operation, the data area is set to the original length of the data record read from the queue.

If you specify the SET option, the LENGTH must be specified.

NEXT

specifies retrieval for the next sequential logical record following the last record retrieved (by any task), or the first record if no previous record has been retrieved.

NUMITEMS(*data-area*)

specifies a halfword binary field into which CICS stores a number indicating how many items there are in the queue. This only occurs if the command completes normally.

QUEUE(*name*)

specifies the symbolic name (1–8 characters) of the queue to be read from.

QNAME(*name*)

an alternative to QUEUE, QNAME specifies the symbolic name (1–16 characters) of the queue to be read from.

SET(*ptr-ref*)

specifies the pointer reference that is set to the address of the retrieved data. The pointer reference, unless changed by other commands or statements, is valid until the next READQ TS command or the end of task.

If the application program is defined with DATALOCATION(ANY), the address of the data can be above or below the 16MB line. If the application program is defined with DATALOCATION(BELOW), the address of the data is below the 16MB line.

If TASKDATAKEY(USER) is specified for the running task, and storage protection is active, the data returned is in a user-key. If TASKDATAKEY(CICS) is specified and storage protection is active, the data returned is in a CICS-key.

SYSID(*systemname*)

(remote and shared queues only) specifies the system name (1–4 characters) identifying the remote system or shared queue pool to which the request is directed.

Conditions**INVREQ**

occurs in either of the following situations:

- the queue was created by CICS internal code.
- the queue name specified consists solely of binary zeroes.

Default action: terminate the task abnormally.

IOERR

occurs when there is an irrecoverable input/output error.

Default action: terminate the task abnormally.

ISCINVREQ

occurs when the remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

WRITEQ TS

ITEMERR

occurs in any of the following situations:

- The item number specified is invalid (that is, outside the range of item numbers written to the queue).
- An attempt is made to read beyond the end of the queue using the NEXT (default) option.

Default action: terminate the task abnormally.

LENGERR

occurs when the length of the stored data is greater than the value specified by the LENGTH option.

This condition only applies to the INTO option and cannot occur with SET.

Default action: terminate the task abnormally.

NOTAUTH

occurs when a resource security check has failed on QUEUE(name).

Default action: terminate the task abnormally.

QIDERR

occurs when the queue specified cannot be found, either in main or in auxiliary storage.

Default action: terminate the task abnormally.

SYSIDERR

occurs in any of the following situations:

- When the SYSID option specifies a name that is neither the local system nor a remote system (made known to CICS by defining a CONNECTION).
- When the link to the remote system is closed.

Default action: terminate the task abnormally.

Examples

The following example shows how to read the first (or only) record from a temporary storage queue into a data area specified in the request; the LENGTH data area is given the value of the length of the record.

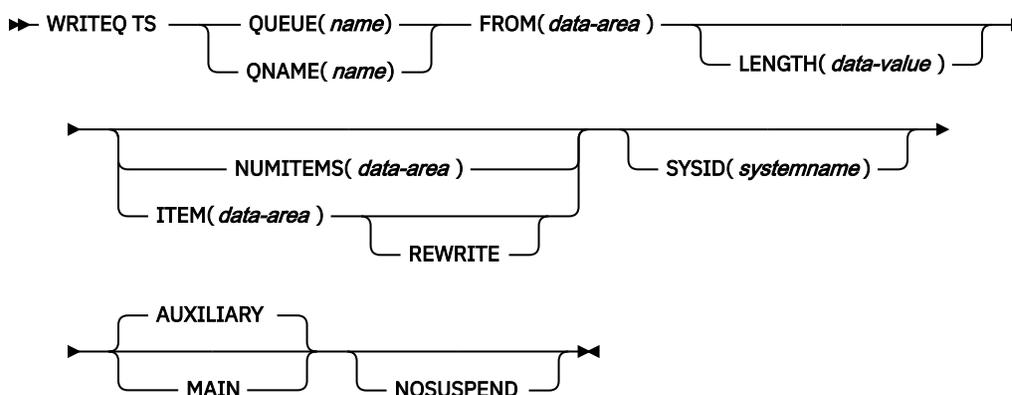
```
EXEC CICS READQ TS
      ITEM(1)
      QUEUE(UNIQNAME)
      INTO(DATA)
      LENGTH(LDATA)
```

The following example shows how to read the next record from a temporary storage queue into a data area provided by CICS; the pointer reference specified by the SET option is set to the address of the storage area reserved for the data record, and the LENGTH data area is given the value of the length of the record.

```
EXEC CICS READQ TS
      QUEUE(DESCRQ)
      SET(PREF)
      LENGTH(LENG)
      NEXT
```

WRITEQ TS

Write data to a temporary storage queue.

WRITEQ TS

Conditions: INVREQ, IOERR, ISCINVREQ, ITEMERR, LENGERR, NOSPACE, NOTAUTH, QIDERR, SYSIDERR

Note for dynamic transaction routing: Using this command could create inter-transaction affinities that adversely affect the use of dynamic transaction routing. See the [CICS Application Programming Guide](#) for more information about transaction affinities.

Description

WRITEQ TS stores temporary data (records) in a temporary storage queue in main or auxiliary storage.

If a queue has been defined as recoverable, the program must not issue a WRITEQ TS if a DELETEQ TS has previously been issued within the same logical unit of work. In other words, following a DELETEQ TS, no WRITEQ TS can be issued until after a syncpoint has occurred.

If there is insufficient space available in the temporary storage data set to satisfy the WRITEQ TS request, the task is suspended until space does become available. (Space may be released by other tasks in the system.) If, however, space is not available in the temporary storage data set, and the NOSUSPEND option has been specified, or there is an active HANDLE CONDITION for NOSPACE, the NOSPACE condition is raised.

Options**AUXILIARY**

specifies that the temporary storage queue is on a direct access storage device in auxiliary storage. This is the default value for the first write.

This option is ignored for an existing queue.

FROM(data-area)

specifies the data to be written to temporary storage.

ITEM(data-area)

specifies, as a halfword binary value, the item number of the logical record to be replaced in the queue (REWRITE option also specified).

ITEM can be both an input and output field to CICS. As such, programmers should ensure that the ITEM field is not defined within protected storage when issuing a WRITEQ command. If the ITEM value were a literal (for example), command checking (CMDPROT=YES) would result in an AEYD abend occurring.

Note: In earlier releases, ITEM on a WRITEQ TS without REWRITE would perform a similar function to NUMITEMS. This function is retained for compatibility.

LENGTH(data-value)

specifies the length, as a halfword binary value, of the data to be written.

You must specify this option if you are using SYSID.

The maximum length is 32763.

MAIN

specifies that the temporary storage queue is in main storage.

This option is ignored for an existing queue.:

If you use the MAIN option to write data to a temporary storage queue on a remote system, the data is stored in main storage if the remote system is accessed by the CICS multiregion operation (MRO) facility. If these conditions are not met, the data is stored in auxiliary storage.

If the system is MRO and MAIN is specified, the queue is not recoverable and SYNCPOINT ROLLBACK does not function.

NOSUSPEND

specifies that the application program is not to be suspended if there is insufficient space in the temporary storage data set to satisfy the WRITEQ TS request. Instead, the NOSPACE condition is raised.

Note, however, that if a HANDLE CONDITION for NOSPACE is active when the command is executed, this also overrides the default action, and control is passed to the user label supplied in the HANDLE CONDITION. This takes precedence over the NOSUSPEND option but is, of course, negated by either NOHANDLE or RESP.

This does not apply to temporary storage queues in main storage (MAIN option).

NUMITEMS(*data-area*)

specifies a halfword binary field into which CICS stores a number indicating how many items there are now in the queue, after the WRITEQ TS command is executed.

If the record starts a new queue, the item number assigned is 1; subsequent item numbers follow on sequentially. NUMITEMS is not valid if REWRITE is specified.

QUEUE(*name*)

specifies the symbolic name (1–8 characters) of the queue to be written to. If the queue name appears in the TST, and the entry is marked as remote, the request is shipped to a remote system. The name must be unique within the CICS system. Do not use X'FA' through X'FF', or **, or \$\$, or DF, as the first character of the name; these characters are reserved for CICS use. The name cannot consist solely of binary zeros.

QNAME(*name*)

an alternative to QUEUE, QNAME specifies the symbolic name (1–16 characters) of the queue to be written to. If the queue name appears in the TST, and the entry is marked as remote, the request is shipped to a remote system. The name must be unique within the CICS system. Do not use X'FA' through X'FF', or **, or \$\$, or DF, as the first character of the name; these characters are reserved for CICS use. The name cannot consist solely of binary zeros.

REWRITE

specifies that the existing record in the queue is to be overwritten with the data provided. If the REWRITE option is specified, the ITEM option must also be specified. If the specified queue does not exist, the QIDERR condition occurs. If the correct item within an existing queue cannot be found, the ITEMERR condition occurs and the data is not stored.

SYSID(*systemname*)

specifies the system name (1–4 characters) identifying the remote system to which the request is directed.

Conditions

INVREQ

occurs in any of the following situations:

- A WRITEQ TS command specifies a queue name that consists solely of binary zeros.
- A WRITEQ TS command specifies a queue that is locked and awaiting ISC session recovery.

- The queue was created by CICS internal code.

Default action: terminate the task abnormally.

IOERR

occurs when there is an irrecoverable input/output error.

Default action: terminate the task abnormally.

ISCINVREQ

occurs when the remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

ITEMERR

occurs in any of the following situations:

- The item number specified in a WRITEQ TS command with the REWRITE option, is not valid (that is, it is outside the range of entry numbers assigned for the queue).
- The maximum number of items (32 767) is exceeded.

Default action: terminate the task abnormally.

LENGERR

occurs in any of the following situations:

- The length of the stored data is zero or negative.
- The length of the stored data is greater than 32763.

Default action: terminate the task abnormally.

NOSPACE

occurs when the NOSUSPEND option is specified and there is no space for the data in the auxiliary temporary storage data set.

Also occurs if there is no space and there is an active HANDLE CONDITION for NOSPACE.

Default action: ignore the condition.

NOTAUTH

occurs when a resource security check has failed on QUEUE(name), or QNAME(name).

Default action: terminate the task abnormally.

QIDERR

occurs when the queue specified by a WRITEQ TS command with the REWRITE option cannot be found, either in:

- Main storage
- Auxiliary storage

Default action: terminate the task abnormally.

SYSIDERR

occurs in any of the following situations:

- When the SYSID option specifies a name that is neither the local system nor a remote system (made known to CICS by defining a CONNECTION).
- The link to the remote system is closed.

Default action: terminate the task abnormally.

Chapter 18. EXEC CICS WEB

WEB ENDBROWSE FORMFIELD

Signals the end of a FORMFIELD browse.

WEB ENDBROWSE FORMFIELD

➤ WEB — ENDBROWSE — FORMFIELD ➤

Conditions: INVREQ

Description

WEB ENDBROWSE FORMFIELD terminates the browse of a set of name-value pairs in an HTML form. The form is part of the body of an HTTP request being processed by the current CICS task. No information is returned on the ENDBROWSE.

Conditions

INVREQ

occurs for the following conditions. RESP2 values are:

- 1** The command is being issued in a non-CICS Web application.
- 3** The command is being issued for a non-HTTP request.
- 4** The command is being issued before a WEB STARTBROWSE command is issued.

WEB ENDBROWSE HTTPHEADER

Signals the end of an HTTP header browse.

WEB ENDBROWSE HTTPHEADER

➤ WEB — ENDBROWSE — HTTPHEADER ➤

Conditions: INVREQ

Description

WEB ENDBROWSE HTTPHEADER terminates the browse of HTTP headers. No information is returned on the ENDBROWSE.

Conditions

INVREQ

occurs for the following conditions. RESP2 values are:

- 1** The command is being issued in a non-CICS Web interface application.

WEB EXTRACT

3

The command is being issued for a non-HTTP request.

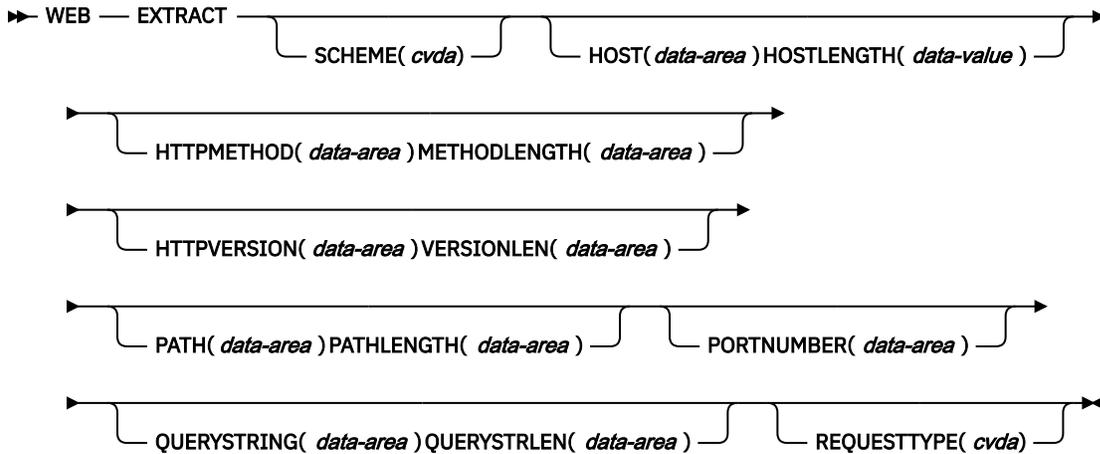
4

The command is being issued before a WEB STARTBROWSE command is issued.

WEB EXTRACT

Obtain information about an inbound client HTTP request that has been made to CICS.

WEB EXTRACT



Conditions: ILLOGIC, INVREQ, LENGERR

Description

WEB EXTRACT allows the application to obtain additional information about the inbound request from the client.

Note: This command can also be specified as EXTRACT WEB.

Options

HOST(*data-area*)

specifies a buffer to contain the host component of the URL, as specified either in the Host header field for the request, or in the request line (if an absolute URI was used for the request). The port number is presented separately using the PORTNUMBER option.

HOSTLENGTH(*data-area*)

specifies the length of the buffer supplied on the HOST option, as a fullword binary variable, and is set to the actual length of the data returned to the application. 116 characters is an appropriate size to specify for this data-area. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

HTTPMETHOD(*data-area*)

specifies a buffer to contain the HTTP method string on the request line of the inbound message.

HTTPVERSION(*data-area*)

specifies a buffer to contain the HTTP version from the Web client, as stated on its request. "1.1" indicates HTTP/1.1, and "1.0" indicates HTTP/1.0.

Note: CICS does not make any special provision for a Web client that is below HTTP/1.0 level. CICS behaves as though they were at HTTP/1.0 level, and returns "1.0" as the HTTP version. If your application program writes HTTP headers that might be unsuitable for a Web client at HTTP/1.0 level, or if you intend to send chunked information to the Web client (which cannot be received by a client at HTTP/1.0 level), your application program should check the HTTP version information.

METHODLENGTH(data-value)

specifies the length of the buffer supplied on the HTTPMETHOD option, and is set to the actual length of the data returned to the application. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated. The actual length of the HTTPMETHOD is returned in METHODLENGTH.

PATH(data-area)

specifies a buffer to contain the PATH specified in the request line of the inbound message.

PATHLENGTH(data-value)

specifies the length of the buffer supplied on the PATH option, and is set to the actual length of the data returned to the application. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated. The actual length of the PATH is returned in PATHLENGTH.

PORTNUMBER(data-area)

returns a data area containing the port number specified in the request line of the message. The value returned in the data area is a fullword binary value. Well-known port numbers for a service are normally omitted from the URL. If the port number is not present in the URL, the WEB EXTRACT command identifies and returns it based on the scheme. For HTTP, the well-known port number is 80, and for HTTPS, the well-known port number is 443. If a port number is returned which is not the default for the scheme, you need to specify the port number explicitly to gain access to the URL (for example, if you are using this information in a WEB OPEN command).

QUERYSTRING(data-area)

specifies a buffer to contain the query string on the request line of the HTTP request. The query string is the value or values encoded after the question mark (?) delimiting the end of the path on the request line of an HTTP request. The data is passed to the application in its escaped form.

QUERYSTRLEN(data-area)

specifies the length of the buffer supplied on the QUERYSTRING option, as a fullword binary variable, and is set to the actual length of the data returned to the application (the query string). 256 characters is an appropriate size to specify for this data-area. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

REQUESTTYPE(cvda)

This option specifies the type of request received.

CVDA values are:**HTTPYES**

indicates an HTTP request.

HTTPNO

indicates a non-HTTP request.

SCHEME(cvda)

returns the scheme used for the connection between CICS and the Web client or server.

CVDA values are:**HTTP**

is the HTTP protocol, without SSL.

HTTPS

is the HTTPS protocol, which is HTTP with SSL.

VERSIONLEN(data-area)

specifies the length of the buffer supplied on the HTTPVERSION option, as a fullword binary variable, and is set to the actual length of the data returned to the application.

Conditions**ILLOGIC**

occurs for the following conditions. RESP2 values are:

9

CICS logic error.

WEB READ FORMFIELD

INVREQ

occurs for the following conditions. RESP2 values are:

- 1 The command is being issued in a non-CICS Web interface application.
- 3 The command is being issued for a non-HTTP request (this is set only if one or more of HTTPMETHOD, HTTPVERSION, or PATH is specified and the request is a non-HTTP request).

LENGERR

occurs for the following conditions. RESP2 values are:

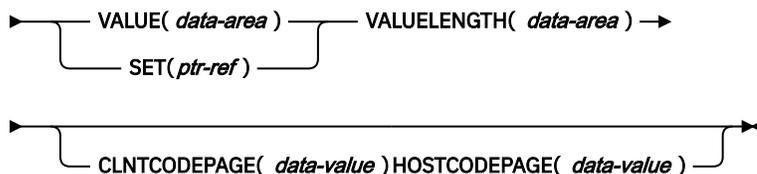
- 4 The method exceeds the length specified (METHODLENGTH option).
- 5 The PATHLENGTH option value was not greater than zero.
- 6 The HTTP version exceeds the length specified (VERSIONLEN option).
- 7 The VERSIONLEN option value was not greater than zero.
- 8 The query string exceeds the length specified (QUERYSTRLEN option).
- 21 The HOSTLENGTH option value was not greater than zero.
- 29 The host name exceeds the length specified (HOSTLENGTH option).
- 30 The path exceeds the length specified (PATHLENGTH option).

WEB READ FORMFIELD

Retrieve the value of a field from an HTML form.

WEB READ FORMFIELD

➔ WEB — READ — FORMFIELD(*data-value*) — NAMELENGTH(*data-value*) —➔



Conditions: INVREQ, LENGERR, NOTFND

Description

WEB READ FORMFIELD retrieves the value of a specific field from an HTML form, the name of which is given on the request in the FORMFIELD parameter. The form is part of the body of an HTTP request being processed by the current CICS task.

Options

CLNTCODEPAGE(name)

specifies the 40-character name of the codepage to be used when data is converted from the client codepage. If this is not specified, CICS obtains it from the charset parameter on the Content-Type header of the HTTP request for the current CICS task. If the Content-Type header is not present,

CICS uses codepage 819 (ISO-8859-1). If you specify CLNTCODEPAGE you must also specify HOSTCODEPAGE. Valid values for CLNTCODEPAGE are the CICS-supported CLINTCP and SRVERCP values described in [CICS Family: Communicating from CICS on System/390](#). Mapping is performed between the IANA Coded Character Set values used in HTTP requests to the CLINTCP values known to CICS; these values are listed in the [CICS Internet Guide, SC34-5765](#).

FORMFIELD(data-value)

specifies the name of the form field to extract. It is a string of text containing the name of the requested field. The string of text supplied is not case sensitive.

Note: (for COBOL programmers) If you are not using the COBOL3 translator option, you must use a data area instead of a data value.

HOSTCODEPAGE(name)

specifies the 8-character name of the host codepage to be used when the forms data is converted from the ASCII codepage, it was received in, into the EBCDIC, CICS processes it in. If this is not specified the default (037) is used. If you specify HOSTCODEPAGE you must also specify CLNTCODEPAGE.

NAMELENGTH(data-value)

specifies the length, as a fullword binary value, of the form field name.

SET(ptr-ref)

specifies a pointer reference that is to be set to the address of data received. The pointer reference is valid until the end of the task.

VALUE(data-area)

specifies the buffer to contain the value of the named form field. CICS unescapes any escaped characters before placing them in the buffer.

VALUELENGTH(data-area)

specifies the length, as a fullword binary value, of the buffer that is to contain the form field value. If the value exceeds the length of the buffer, it is truncated. If the length of the form field value is less than the size of the buffer, the form field value is placed in the leftmost byte positions. VALUELENGTH contains the actual length of the data returned.

Conditions

INVREQ

RESP2 values are:

- 1** The command is being issued in a non-CICS Web application.
- 3** The command is being issued for a non-HTTP request.
- 11** The client codepage cannot be found.
- 12** The server codepage cannot be found.
- 13** No forms data has been supplied in the body of the HTTP request.
- 14** The codepage combination for client and server is invalid.

LENGERR

RESP2 values are:

- 1** The length in VALUELENGTH is less than or equal to zero.
- 5** The form field value has been truncated during a read operation because the receiving buffer is too small.

WEB READ HTTPHEADER

NOTFND

RESP2 values are:

1

The form field with the given name cannot be found.

WEB READ HTTPHEADER

Extract HTTP header information.

WEB READ HTTPHEADER

► WEB — READ — HTTPHEADER(*data-value*) — NAMELENGTH(*data-value*) — VALUE(*data-area*) →
 ◄— VALUELENGTH(*data-area*) —►

Conditions: INVREQ, LENGERR, NOTFND

Description

WEB READ HTTPHEADER extracts HTTP header information.

Options

HTTPHEADER(*data-value*)

specifies the name of the HTTP header to be extracted.

NAMELENGTH(*data-area*)

specifies the length, as a fullword binary value, of the HTTP header name.

VALUE(*data-area*)

specifies the buffer to contain the value of the HTTP header being extracted.

VALUELENGTH(*data-area*)

specifies the length of the buffer supplied on the VALUE option, and is set to the actual length of the data returned to the application. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated. The actual length of the VALUE is returned in VALUELENGTH.

Conditions

INVREQ

RESP2 values are:

1

The command is being issued in a non-CICS Web interface application.

3

The command is being issued for a non-HTTP request.

LENGERR

RESP2 values are:

1

The length in VALUELENGTH is less than or equal to zero.

2

The header value has been truncated because the receiving buffer is too small.

NOTFND

RESP2 values are:

1

The header with the given name could not be found.

WEB READNEXT FORMFIELD

Retrieve next name-value pair in an HTML form.

WEB READNEXT FORMFIELD

```
►► WEB — READNEXT — FORMFIELD( data-area ) — NAMELENGTH( data-area ) — VALUE( data-area ) —►
      ◀— VALUELENGTH( data-area ) —▶
```

Conditions: ENDFILE, INVREQ, LENGERR

Description

WEB READNEXT FORMFIELD retrieves the next name-value pair in an HTML form.

Options

FORMFIELD(*data-area*)

specifies the buffer that contains the form field. The case of the name is the same as it is stored in the form.

NAMELENGTH(*data-area*)

specifies the length, as a fullword binary value, of the form field name. If the length of the form field value is less than the size of the buffer, the form field value is placed in the leftmost byte positions. NAMELENGTH contains the actual length of the data returned.

VALUE(*data-area*)

specifies the buffer that contains the value corresponding to the name returned in the FORMFIELD data area. CICS unescapes any escaped characters before placing them in the buffer.

VALUELENGTH(*data-area*)

specifies the length, as a fullword binary value, of the buffer that contains the form field value. If the value exceeds the buffer length, it is truncated. If the length of the form field value is less than the size of the buffer, the form field value is placed in the leftmost byte positions. VALUELENGTH contains the actual length of the data returned.

Conditions

ENDFILE

The end of the list of name/value pairs has been reached.

INVREQ

RESP2 values are:

- 1** The command is being issued in a non-CICS Web application.
- 3** The command is being issued for a non-HTTP request.
- 4** The command is being issued before a WEB STARTBROWSE FORMFIELD has been issued.
- 6** A form field has been found that is not in the expected format.

LENGERR

RESP2 values are:

- 1** NAMELENGTH or VALUELENGTH is less than or equal to zero.

WEB READNEXT HTTPHEADER

4

The form field name has been truncated during a browse operation because the receiving buffer is too small.

5

The form field value has been truncated because the receiving buffer is too small.

WEB READNEXT HTTPHEADER

Retrieve next HTTP header.

WEB READNEXT HTTPHEADER

► WEB — READNEXT — HTTPHEADER(*data-area*) — NAMELENGTH(*data-value*) →

 ► VALUE(*data-area*) — VALUELENGTH(*data-value*) →◄

Conditions: ENDFILE, INVREQ, LENGERR

Description

WEB READNEXT HTTPHEADER retrieves the next HTTP header in the list of headers.

Options

HTTPHEADER(*data-area*)

specifies the buffer to contain the name of the HTTP header being extracted.

NAMELENGTH(*data-area*)

specifies the length of the buffer supplied on the HTTPHEADER option, and is set to the actual length of the data returned to the application. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated. The actual length of the HTTPHEADER name is returned in NAMELENGTH.

VALUE(*data-area*)

specifies the buffer to contain the value of the HTTP header being extracted.

VALUELENGTH(*data-area*)

specifies the length of the buffer supplied on the VALUE option, and is set to the actual length of the data returned to the application. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated. The actual length of the VALUE is returned in VALUELENGTH.

Conditions

ENDFILE

The end of the list of HTTPHEADERS has been reached.

INVREQ

RESP2 values are:

1

The command is being issued in a non-CICS Web interface application.

3

The command is being issued for a non-HTTP request.

4

The command is being issued before a WEB STARTBROWSE has been issued.

6

A header has been found which is not in the format NAME:VALUE.

LENGERR

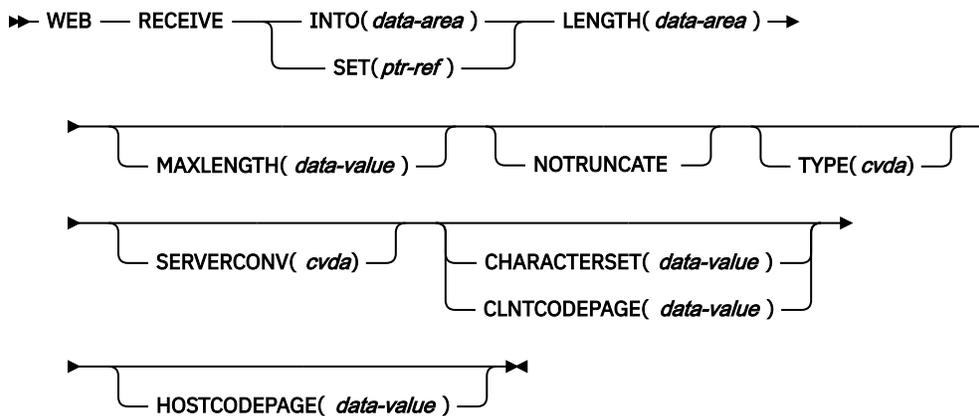
RESP2 values are:

- 1 NAMELENGTH or VALUELENGTH is less than or equal to zero.
- 4 The header name has been truncated during a browse operation because the receiving buffer is too small.
- 5 The header value has been truncated because the receiving buffer is too small.

WEB RECEIVE

Receive an HTTP request, or a non-HTTP message.

WEB RECEIVE



Conditions: INVREQ, LENGERR, NOTFND

Description

WEB RECEIVE receives the body of an HTTP request, or all the data for a non-HTTP message, into an application-supplied buffer. The headers for an HTTP request can be examined separately using the WEB HTTPHEADER commands. The item received by the WEB RECEIVE command can be:

- The body of an HTTP request that a Web client has made to CICS as an HTTP server.
- A non-HTTP message handled by CICS Web support facilities, with the user-defined (USER) protocol on the TCPIP SERVICE definition.

The data is returned in its escaped form. The type of code page conversion used for incoming data received by the CICS application program can be specified on this command.

Options

CHARACTERSET(data-value)

specifies the character set that was used by the Web client for the entity body of the received item. The name of the character set can consist of up to 40 alphanumeric characters, including appropriate punctuation. CICS does not support all the character sets named by IANA. The [“HTML coded character sets”](#) on page 114 lists the IANA character sets that are supported by CICS for code page conversion. When the CHARACTERSET option is specified, SERVERCONV(SRVCONVERT) is assumed, so code page conversion of the entity body takes place. As an alternative to identifying the character set yourself, specifying either SERVERCONV(SRVCONVERT), or HOSTCODEPAGE, or both, and omitting CHARACTERSET, lets CICS identify the character set for the message body. The description for the SERVERCONV option tells you what happens in this case. If you omit all of the code page conversion options, no code page conversion takes place.

CLNTCODEPAGE(data-value)

is supported for migration purposes only. CHARACTERSET replaces it. The action taken by CICS is the same for both keywords. This means that code page conversion does take place when CLNTCODEPAGE or HOSTCODEPAGE is specified, even if the SERVERCONV option is not specified. Code page conversion does not take place if all the code page conversion options are omitted. The CECI transaction will not offer CLNTCODEPAGE but accept it as synonym for CHARACTERSET. CICS Translator messages will refer to CHARACTERSET instead of CLNTCLODEPAGE when needed.

HOSTCODEPAGE(data-value)

specifies the 8-character name of the CICS (host) code page used by the application program, into which the entity body of the received item should be converted from the character set in which it was received from the Web client. When the HOSTCODEPAGE option is specified, SERVERCONV(SRVCONVERT) is assumed, so code page conversion of the entity body takes place. Specifying either SERVERCONV(SRVCONVERT), or CHARACTERSET, or both, and omitting HOSTCODEPAGE, lets CICS determine the host code page. The default if this option is not specified is the default code page for the local CICS. region, as specified in the LOCALCCSID system initialization parameter. If you omit all of the code page conversion options, no code page conversion takes place.

INTO(data-area)

specifies the buffer that is to contain the data being received.

LENGTH(data-area)

specifies a fullword binary variable which is set to the amount of data that CICS has returned to the application. Note that this might be slightly less than the limit that you set using the MAXLENGTH option, especially if a double-byte or multi-byte character set is involved, because CICS does not return a partial character at the end of the data.

- If the NOTRUNCATE option **is not** specified, any further data present in the message has now been discarded. A LENGERR response with a RESP2 value of 57 is returned if further data was present.
- If the NOTRUNCATE option **is** specified, any additional data is retained. A LENGERR response with a RESP2 value of 36 is returned if additional data is available. The description for the NOTRUNCATE option tells you what to do in this case.

MAXLENGTH(data-value)

specifies the maximum amount, as a fullword binary value, of data that CICS is to pass to the application. The MAXLENGTH option applies whether the INTO or the SET option is specified for receiving data. If the data has been sent using chunked transfer-coding, CICS assembles the chunks into a single message before passing it to the application, so the MAXLENGTH option applies to the total length of the chunked message, rather than to each individual chunk. The data is measured after any code page conversion has taken place.

If the length of data exceeds the value specified and the NOTRUNCATE option **is not** specified, the data is truncated to that value, and the remainder of the data is discarded.

If the length of data exceeds the value specified and the NOTRUNCATE option **is** specified, CICS retains the remaining data and can use it to satisfy subsequent RECEIVE commands.

NOTRUNCATE

specifies that when the data available exceeds the length requested on the MAXLENGTH option, the remaining data is not to be discarded immediately but is to be retained for retrieval by subsequent RECEIVE commands. (If no further RECEIVE commands are issued, the data is discarded during transaction termination.)

A single RECEIVE command using the SET option and without the MAXLENGTH option receives all the remaining data, whatever its length. Alternatively, you can use a series of RECEIVE commands with the NOTRUNCATE option to receive the remaining data in appropriate chunks. Keep issuing the RECEIVE command until you are no longer getting a LENGERR response. Bear in mind that if you receive less than the length requested on the MAXLENGTH option, this does not necessarily indicate the end of the data; this could happen if CICS needs to avoid returning a partial character at the end of the data.

SERVERCONV(cvda)

specifies whether or not CICS translates the entity body of the item received, from the character set used by the Web client, to a code page suitable for the application. You can use the CHARACTERSET and HOSTCODEPAGE options on this command to specify the character set and code page that are used. If you specify either of these options, code page conversion (SRVCONVERT) is assumed. Alternatively, you can omit either or both of these options, specify SERVERCONV(SRVCONVERT) and let CICS determine a suitable character set and code page.

- **SRVCONVERT** CICS converts the entity body of the message. When you specify SRVCONVERT without CHARACTERSET, CICS identifies the character set as follows:

1. If the Web client's request has a Content-Type header naming a character set supported by CICS, that character set is used.
2. If the Web client's request has no Content-Type header or the named character set is unsupported, the ISO-8859-1 character set is used.
3. For non-HTTP messages (sent using the USER protocol), the ISO-8859-1 character set is used.

When you specify SRVCONVERT without HOSTCODEPAGE, CICS determines the host code page as the default code page for the local CICS region, as specified in the LOCALCCSID system initialization parameter. If you specify SRVCONVERT alone, note that for code page conversion to take place, the media type for the message must specify a type of data content that can be identified as text according to the IANA definitions. For messages where no media type is given but SRVCONVERT is specified, code page conversion also takes place. If a non-text media type is present, CICS does not convert the message body. However, for compatibility with Web-aware applications coded in earlier releases, if you specify either of the CHARACTERSET or HOSTCODEPAGE options or omit the SERVERCONV option, the media type for the message does not influence code page conversion.

- **NOSRVCONVERT** CICS does not convert the entity body of the item, and it is passed to the application in the character set used by the Web client. If you specify NOSRVCONVERT, you cannot specify the CHARACTERSET or HOSTCODEPAGE options.

SET(ptr-ref)

specifies a pointer reference that is to be set to the address of data received. The pointer reference is valid until the next receive command or the end of task.

TYPE(cvda)

returns the type of request received. CVDA values are:

- **HTTPYES** indicates an HTTP request.
- **HTTPNO** indicates a non-HTTP request.

HTTP requests and non-HTTP requests use different protocols, which are specified on TCPIP SERVICE definitions, and must therefore use different ports. Non-HTTP requests use the user-defined (USER) protocol. You might use the TYPE option to distinguish between the request types if you specify the same user-written application program for responding to both HTTP and non-HTTP requests.

Note: If you omit all of the code page conversion options (SERVERCONV, CLNTCODEPAGE, CHARACTERSET, HOSTCODEPAGE), no code page conversion takes place.

Conditions**INVREQ**

RESP2 values:

- 1** The command is being issued in a non-CICS Web support application.
- 14** Invalid code page combination.
- 46** The SERVERCONV option is invalid.

WEB RETRIEVE

80

CHARACTERSET cannot be specified with SERVERCONV(NOSRVCONVERT).

81

HOSTCODEPAGE cannot be specified with SERVERCONV(NOSRVCONVERT).

84

Body incomplete.

LENGERR

RESP2 values:

1

The MAXLENGTH option value was not greater than zero.

36

Partial response body returned. Use additional RECEIVES to obtain remainder.

57

The response body exceeds the length specified, and the remainder of the body has been discarded.

NOTFND

RESP2 values:

7

Code page not found.

82

Client code page (character set) not found.

83

Host code page (for server) not found.

WEB RETRIEVE

Retrieve a document token.

WEB RETRIEVE

► WEB — RETRIEVE — DOCTOKEN(*data-area*) ◄

Conditions: INVREQ

Description

The WEB RETRIEVE command retrieves the DOCTOKEN of the document which was sent using an earlier WEB SEND command.

Options

DOCTOKEN(*data-area*)

specifies a 16-byte buffer to contain the symbolic name of the document to be retrieved.

Conditions

INVREQ

RESP2 values:

1

The command is issued in a non-CICS Web interface application.

2

A WEB SEND command has not been issued.

- **EVENTUAL** sends the response to the Web client at end of task. If CHUNKING is specified, the EVENTUAL option is ignored. This option produces the same behavior as CICS Web support had in releases before CICS Transaction Server for z/VSE, Version 2 Release 2, and it is the default for CICS as an HTTP server.

CHARACTERSET(data-value)

specifies a character set into which CICS translates the entity body of the item sent by the command before sending. The name of the character set can consist of up to 40 alphanumeric characters, including appropriate punctuation. CICS does not support all the character sets named by IANA. [“HTML coded character sets” on page 114](#) lists the IANA character sets that are supported by CICS for code page conversion. When the CHARACTERSET option is specified, SERVERCONV(SRVCONVERT) is assumed, so code page conversion of the entity body takes place. As an alternative to selecting the character set yourself, specifying either SERVERCONV(SRVCONVERT), or HOSTCODEPAGE (if allowed), or both, and omitting CHARACTERSET, lets CICS determine a suitable character set for the message body. The description for the SERVERCONV option tells you what happens in this case. If you omit all of the code page conversion options, no code page conversion takes place.

CHUNKING(cvda)

is used for controlling the message send when the message is being sent in chunks (known as chunked transfer-coding). The default when the option is not specified is that chunked transfer-coding is not in use. Chunked transfer-coding is only acceptable to HTTP/1.1 clients, and it cannot be used with HTTP/1.0 clients or non-HTTP messages.

The content of a chunked message can be divided into chunks in whatever way is most convenient for the application program. The body of a chunked message cannot be formed directly from CICS documents, so the DOCTOKEN option cannot be used.

Use a separate WEB SEND command with CHUNKING(CHUNKYES) for each chunk of the message. Use the FROM option to specify the chunk of data, and the FROMLENGTH option to specify the length of the chunk. Other options for the message, such as the CLOSESTATUS option, can be specified on the first WEB SEND command of the sequence (which sends the first chunk), but do not specify them on subsequent commands (which send the second and subsequent chunks).

When you have sent the last chunk of the data, specify a further WEB SEND command with CHUNKING(CHUNKEND) and no FROM or FROMLENGTH option. CICS then sends an empty chunk to the recipient to complete the chunked message.

If one of the WEB SEND commands fails during the sequence, an error response is returned, and subsequent sends will also fail. The application should handle this situation appropriately. If all of the chunks are sent successfully but the application does not issue the final WEB SEND command with CHUNKING(CHUNKEND), the transaction is abended with abend code AWBP. An incomplete chunked message should be ignored and discarded by the recipient.

CVDA values are:

- **CHUNKNO** Chunked transfer-coding is not used for the message. This is the default if the CHUNKING option is not specified.
- **CHUNKYES** Chunked transfer-coding is in progress. The data specified by the FROM option represents a chunk of the message.
- **CHUNKEND** Chunked transfer-coding is complete. No data is specified for this send. CICS sends an empty chunk to the recipient to complete the chunked message.

CLNTCODEPAGE(name)

specifies the 40-character name of the ASCII code page in which the data is to be delivered to the browser. If this is not specified, data is not converted, and is sent to the browser in the form in which the application supplied it. Valid values for CLNTCODEPAGE are the CICS-supported CLINTCP and SRVERCP values described in [CICS Family: Communicating from CICS on System/390](#). Mapping is performed between the IANA Coded Character Set values used in HTTP requests and the CLINTCP values known to CICS; these values are listed in the [CICS Internet Guide, SC34-5765](#).

CLOSESTATUS(cvda)

specifies whether or not CICS closes the connection after sending the message. The default is that the connection is not closed.

The CVDA values are:

- **CLOSE** CICS writes a Connection header with the "close" connection option (Connection: close) for this response, and closes the connection with the Web client after sending the response. The header notifies the Web client of the closure. (For a Web client at HTTP/1.0 level, CICS achieves the same effect by omitting the Connection: Keep-Alive header.) If chunked transfer-coding is in use, the CLOSESTATUS(CLOSE) option can be specified on the first chunk of the message, to inform the Web client that the connection is closed after the chunked message is complete.
- **NOCLOSE** means that the Connection: close header is not used for this response, and the connection is kept open. If the Web client is identified as HTTP/1.0 and has sent a Connection header with the "Keep-Alive" connection option (Connection: Keep-Alive), CICS sends the same header, to notify that a persistent connection will be maintained.

DOCTOKEN(name)

specifies the 16-byte symbolic name of the document to be sent.

FROM(data-area)

specifies a buffer of data which holds the complete message body, or a chunk of the message body. The message body is built by the application program. When you specify the FROM option, use the FROMLENGTH option to specify the length of the buffer of data. The DOCTOKEN option provides an alternative way to create the message body, but that option cannot be used for the body of a chunked message. There is no set maximum limit for the size of the data-area, but its size is limited in practice by storage considerations.

FROMLENGTH(data-value)

specifies the length, as a fullword binary value, of the buffer of data supplied on the FROM option. It is important to state this value correctly, because an incorrect data length can cause problems for the recipient of the message.

HOSTCODEPAGE(data-value)

specifies the 8-character name of the CICS (host) code page that was used by the application program for the entity body of the response. When the HOSTCODEPAGE option is specified, SERVERCONV(SRVCONVERT) is assumed, so code page conversion of the entity body takes place. Specifying either SERVERCONV(SRVCONVERT), or CHARACTERSET, or both, and omitting HOSTCODEPAGE, lets CICS identify the host code page. If a CICS document is used to form the response body (DOCTOKEN option), do not specify the HOSTCODEPAGE option, because CICS identifies the host code page from the CICS document domain's record of the host code pages for the document. If a buffer of data is used to form the response body (FROM option), you may need to specify HOSTCODEPAGE. The default if this option is not present is the default code page for the local CICS region, as set in the LOCALCCSID system initialization parameter. If you require code page conversion but your application has used a different code page, use HOSTCODEPAGE to specify it. If you omit all of the code page conversion options, no code page conversion takes place.

LENGTH(data-value)

is supported for migration purposes only. STATUSLEN replaces it.

MEDIATYPE(data-value)

specifies the data content of the message body, for example text/xml. The media type is up to 56 alphanumeric characters, including appropriate punctuation. See ["IANA media types and character sets" on page 113](#) for more information about media types. CICS checks that the format of the media type is correct, but does not check the validity of the media type against the data content. CICS does not provide a default. In some circumstances, the media type that you specify affects whether or not code page conversion is carried out; see the description of the SERVERCONV option for more information.

SERVERCONV(cvda)

specifies whether or not CICS translates the entity body of the item sent by the command before sending, from the code page used by the application, to a character set suitable for the recipient. You can use the CHARACTERSET and HOSTCODEPAGE options on this command to specify the character

set and code page that are used. If you specify either of these options, code page conversion (SRVCONVERT) is assumed. Alternatively, you can omit either or both of these options, specify SERVERCONV(SRVCONVERT) and let CICS determine a suitable character set and code page.

- **SRVCONVERT** CICS converts the entity body of the message. When you specify SRVCONVERT without CHARACTERSET, CICS determines a suitable character set as follows:
 1. If the Web client's request has a Content-Type header naming a character set supported by CICS, that character set is used.
 2. If the Web client's request has no Content-Type header or the named character set is unsupported, the ISO-8859-1 character set is used.
 3. For non-HTTP messages (sent using the USER protocol), the ISO-8859-1 character set is used. When you specify SRVCONVERT without HOSTCODEPAGE, CICS identifies the host code page as follows:
 - If the FROM option is used, CICS identifies the host code page as the default code page for the local CICS region, as specified in the LOCALCCSID system initialization parameter.
 - If the DOCTOKEN option is used, CICS identifies the host code page from the CICS document domain's record of the host code pages for the document.

If you specify SRVCONVERT alone, note that for code page conversion to take place, the MEDIATYPE option must specify a type of data content that can be identified as text according to the IANA definitions. For non-text media types, CICS does not convert the message body. However, for compatibility with Web-aware applications coded in earlier releases, if you specify either of the CHARACTERSET or HOSTCODEPAGE options or omit the SERVERCONV option, the MEDIATYPE option does not influence code page conversion.

- **NOSRVCONVERT** CICS does not convert the entity body of the HTTP request, and it is sent to the server in the code page used by the application. If you specify NOSRVCONVERT, you cannot specify the CHARACTERSET or HOSTCODEPAGE options.

Note: If you omit all of the code page conversion options (SERVERCONV, CLNTCODEPAGE, CHARACTERSET, HOSTCODEPAGE), no code page conversion takes place.

STATUSCODE(data-value)

specifies a standard HTTP status code determined by the application program, which is to be inserted on the status line of the HTTP response. The code is a halfword binary value. Examples are 200 (normal response) or 404 (not found). If this option is not specified, CICS supplies a default of 200. ["HTTP status code reference for CICS Web support"](#) on page 108 has information about the use of status codes for CICS Web support. For status codes 204, 205, and 304, a message body is not allowed, and CICS returns an error response to the command if you attempt to include one. Other than that, CICS does not check that your use of the status code is appropriate.

Note: STATUSCODE requires STATUSTEXT and vice versa.

STATUSLEN(data-value)

specifies the length, as a fullword binary value, of the string supplied on the STATUSTEXT option.

STATUSTEXT(data-area)

specifies a data-area containing human-readable text to describe the reason for the status code. The text is known as a reason phrase. Examples are "OK" (accompanying a 200 status code), or "Bad Request" (accompanying a 400 status code). The HTTP/1.1 specification (RFC 2616) defines a recommended reason phrase for each status code, but you do not have to use these.

Note: STATUSCODE requires STATUSTEXT and vice versa.

Conditions

INVREQ

occurs for the following conditions. RESP2 values are:

1

The command is being issued in a non-CICS Web support application.

- 11** Action code invalid.
- 13** Close status invalid
- 14** Invalid code page combination.
- 32** Media type invalid.
- 41** The connection has been closed.
- 46** The SERVERCONV option is invalid.
- 72** Status code does not support a message body.
- 75** Invalid send sequence.
- 77** Chunk incomplete.
- 80** CHARACTERSET cannot be specified with SERVERCONV(NOSRVCONVERT).
- 81** HOSTCODEPAGE cannot be specified with SERVERCONV(NOSRVCONVERT).
- 85** Chunking cannot be used with non-HTTP messages.
- 86** Chunking cannot be used with HTTP/1.0 clients.
- 87** Status code not allowed.
- 88** Host code page not allowed.
- 89** Previous send over this connection failed. No further sends permitted.
- 90** STATUSCODE and STATUSTEXT options not allowed for second or subsequent chunks.
- 91** CHARACTERSET and CLNTCODEPAGE options not allowed for second or subsequent chunks.
- 92** HOSTCODEPAGE option not allowed for second or subsequent chunks.
- 93** MEDIATYPE option not allowed for second or subsequent chunks,
- 94** CLOSESTATUS option not allowed for second or subsequent chunks.
- 95** SERVERCONV option not allowed for second or subsequent chunks.
- 120** The CHUNKING option is invalid.
- 121** FROMLENGTH option required.

WEB STARTBROWSE FORMFIELD

122

FROM option required.

123

No message body specified. Use FROM, DOCTOKEN or CHUNKING(CHUNKEND).

124

CHUNKING option not specified, FROMLENGTH option required.

125

CHUNKNO specified, FROM option required.

126

CHUNKNO specified, FROMLENGTH option required.

127

CHUNKYES specified, FROM option required.

128

CHUNKYES specified, FROMLENGTH option required.

129

FROM option not allowed with CHUNKING(CHUNKEND).

130

FROMLENGTH option not allowed with CHUNKING(CHUNKEND).

131

FROMLENGTH option specified as zero.

IOERR

RESP2 values are:

42

Socket error.

NOTFND

occurs for the following conditions. RESP2 values are:

1

The document has not been created or the name is incorrectly specified.

7

Client code page (character set) not found.

83

Host code page (for server) not found.

WEB STARTBROWSE FORMFIELD

Signal the start of an HTML form field browse.

WEB STARTBROWSE FORMFIELD



Conditions: INVREQ

Description

WEB STARTBROWSE FORMFIELD signals the start of a browse of a set of name-value pairs in an HTML form that is part of the body of an HTTP request being processed by the current CICS task.

Options

CLNTCODEPAGE(name)

specifies the 40-character name of the codepage to be used when data is converted from the client codepage. If this is not specified, CICS obtains it from the charset parameter on the Content-Type header of the HTTP request for the current CICS task. If the Content-Type header is not present, CICS used codepage 819 (ISO-8859-1). If you specify CLNTCODEPAGE you must also specify HOSTCODEPAGE. Valid values for CLNTCODEPAGE are the CICS-supported CLINTCP and SRVERCP values described in *CICS Family: Communicating from CICS on System/390*. Mapping is performed between the IANA Coded Character Set values used in HTTP requests to the CLINTCP values known to CICS; these values are listed in [“IANA media types and character sets”](#) on page 113 .

HOSTCODEPAGE(name)

specifies the 8-character name of the host codepage used when data is converted from the ASCII codepage, it was received in, into the EBCDIC, CICS processes it in. If this is not specified the default (037) is used. If you specify HOSTCODEPAGE you must also specify CLNTCODEPAGE.

Conditions

INVREQ

occurs for the following conditions. RESP2 values are:

- 1** The command is being issued in a non-CICS Web application.
- 3** The command is being issued for a non-HTTP request.
- 5** There is already a WEB STARTBROWSE in progress.
- 11** The client codepage cannot be found.
- 12** The server codepage cannot be found.
- 13** No forms data has been supplied in the body of the HTTP request.
- 14** The codepage combination for client and server is invalid.

WEB STARTBROWSE HTTPHEADER

Signal the start of an HTTP header browse.

WEB STARTBROWSE HTTPHEADER

➤ WEB — STARTBROWSE — HTTPHEADER ➤

Conditions: INVREQ

Description

WEB STARTBROWSE HTTPHEADER signals the start of a browse of the HTTP header information.

Conditions

INVREQ

occurs for the following conditions. RESP2 values are:

WEB WRITE

- 1** The command is being issued in a non-CICS Web interface application.
- 3** The command is being issued for a non-HTTP request.

WEB WRITE HTTPHEADER

Build HTTP header information.

WEB WRITE HTTPHEADER

► WEB — WRITE — HTTPHEADER(*data-value*) — NAMELENGTH(*data-value*) — VALUE(*data-value*) →
 ► VALUELENGTH(*data-value*) ◄

Conditions: INVREQ, LENGERR

Description

WEB WRITE allows the application to add HTTP header information to the response.

Options

HTTPHEADER(*data-value*)

specifies the name of the HTTP header to be added to the request or response. The name, which is a string of text, should conform to the standards in the HTTP specification to which you are working. No validation of the information is done by CICS. If the header already exists, it is replaced with the new information.

NAMELENGTH(*data-value*)

specifies the length, as a fullword binary value, of the HTTP header name.

VALUE(*data-value*)

specifies the value of the named HTTP header. The value, which is a string of text, should conform to the standards in the HTTP specification to which you are working.

VALUELENGTH(*data-area*)

specifies the length, as a fullword binary value, of the HTTP header value.

Conditions

INVREQ

occurs for the following conditions. RESP2 values are:

- 1** The command is being issued in a non-CICS Web support application.
- 6** Client did not send TE: trailers on request, so trailing headers cannot be used.
- 19** Header not allowed. Some request headers may only be generated by CICS.
- 44** Header not allowed as a trailing header (trailer).
- 69** Chunked transfer-coding not supported.
- 70** Trailer header has not been created, so trailing headers cannot be written.
- 71** Chunked transfer-coding error.

74

Previous send failed.

78

Too late to write trailing headers for this message.

LENGERR

RESP2 values are:

35

The length in NAMELENGTH is not greater than zero.

55

The length in VALUELENGTH is not greater than zero.

Chapter 19. Reference Information

HTTP header reference for CICS Web support

In CICS Web support, when messages are sent out from CICS, some HTTP headers are provided automatically by CICS, and some can be added by the user. When messages are sent to CICS, CICS takes action in response to some HTTP headers, and a user application program can take action in response to others. This reference describes how CICS Web support handles HTTP headers.

The standard HTTP headers are described in the HTTP/1.1 specification (RFC 2616) and the HTTP/1.0 specification (RFC 1945). There are many possible HTTP headers, including extension headers that are not part of the HTTP protocol specifications. For fuller listings, you should consult the HTTP specification to which you are working.

This topic explains the general use of HTTP headers in CICS Web support, and the actions that CICS Web support takes for specific headers. Check the HTTP specification to which you are working for detailed guidance and requirements about how you should use HTTP headers, such as the correct format for header values, and the contexts in which each header should be used.

HTTP headers on messages received by CICS

- When an HTTP request or response is received by CICS, some of the HTTP headers are used to determine actions that CICS Web support takes. [Table 3 on page 106](#) shows the actions taken by CICS for headers on an HTTP request. Other headers are not used by CICS, and it is up to the user application to take appropriate action in response to these.
- All headers received for a message, whether or not they have been used by CICS, are made available to a user application for inspection using the WEB READ HTTPHEADER command and the HTTP header browsing commands. CICS does not alert the user application to the presence of any particular header on a message. Ignore any headers that the application does not need or understand.
- CICS already deals with the MUST level requirements in the HTTP/1.1 specification relating to actions that the server or client must perform on receiving a message. Because of this, you may receive and use a request or response without examining the headers. However, you will probably need to examine the headers for information relating to actions that you take in future communications with the Web client or server.

HTTP headers on messages sent out from CICS

- On an HTTP request or response that is sent out from CICS with HTTP/1.1 as its version, CICS automatically supplies key headers that should normally be written for a basic message to be compliant with the HTTP/1.1 specification. On an HTTP response with HTTP/1.0 as its version, CICS automatically supplies a smaller number of headers. Some of these headers are generated by CICS for every message, and some are produced because of options that you specify on the WEB SEND command in a user application program. [Table 4 on page 108](#) list the headers that are written for each HTTP version, and the source of the header. If the user application program writes a header that CICS also generates, CICS handles this depending on the situation:
 - For CICS as an HTTP server, if the header is appropriate for a response, CICS does not overwrite it, but allows the application's version to be used.
 - If the header is not normally appropriate for the type of message (request or response), CICS allows it, as is the case for all user-defined headers. This situation should not occur if your message is compliant with the HTTP specification to which you are working.
- A user application program can add further HTTP headers to a response using the WEB WRITE HTTPHEADER command. CICS tolerates and passes on any additional HTTP headers. If you are

providing a static response with a CICS document template, headers cannot be added to the response beyond those that are automatically supplied by CICS.

- CICS does not check the name or value of user-written headers. You should ensure that your application program is providing correct, and correctly formatted, information in a way that meets the HTTP specification to which you are working. Be particularly careful to check the HTTP specification for applicable requirements if your application is performing complex actions. There are likely to be important (MUST or SHOULD level) requirements to provide certain headers to describe these actions. For example, special HTTP headers are required if you are:
 - Responding to, or making, conditional requests using the modification date of the document or an entity tag.
 - Varying the content of a response according to the client capability or national language requirements of the Web client.
 - Providing a response, or making a request, that involves a range of a document rather than the full document.
 - Providing cache control information for a response.

The use of certain status codes on your response might also require particular HTTP headers. For example, if you use the status code 405 (Method not allowed), you must use the Allow header to state the methods which *are* allowed. [“HTTP status code reference for CICS Web support”](#) on page 108 has more information about the use of status codes.

The Upgrade header

- Note as a special case that in CICS Web support, protocol upgrading is not supported. This means:
 - For CICS as an HTTP server, it is not possible for an application to take any action in response to an Upgrade header sent by a Web client.

CICS does not support a switch in HTTP version during a connection, and upgrades in the security layer are not supported.

CICS as an HTTP server: Headers where CICS takes action on receiving an HTTP request

Table 3 on page 106 shows the action that CICS takes for certain headers on a request received from a Web client.

<i>Table 3. CICS actions for headers on an HTTP request</i>		
Header received from Web client	Action taken by CICS where response is to be handled by user application program	Action taken by CICS where response is to be provided by static document
Authorization	Passes supplied user ID and password to RACF for verification, and rejects request if these are invalid	As for application-generated response.
Connection	Carries out Web client's request for connection close after sending response.	As for application-generated response.

Table 3. CICS actions for headers on an HTTP request (continued)

Header received from Web client	Action taken by CICS where response is to be handled by user application program	Action taken by CICS where response is to be provided by static document
Content-Length	CICS requires the Content-Length header on all inbound HTTP/1.1 messages that have a message body. If a message body is present but the header is not provided, or its value is inaccurate, the socket receive for the faulty message or for a subsequent message can produce unpredictable results. For HTTP/1.0 messages that have a message body, the Content-Length header is optional.	Although a message body is not used in processing for a static response, it must still be received from the socket, so the same requirements apply as for an application-generated response.
Content-Type	Parses header to identify media type and character set for code page conversion	Parses header to identify character set for code page conversion of response.
Expect	Sends 100-Continue response to Web client and waits for remainder of request.	As for application-generated response.
Host	If this header is not present and the client is HTTP/1.1, sends 400 (Bad Request) response to Web client.	As for application-generated response.
If-Unmodified-Since	If header is present, always sends 412 (Precondition Failed) response to Web client, indicating that the response has been modified since the specified time. (This means that user applications do not have to check for the presence of this header.)	Document template: As for application-generated response, assumes that the response has been modified and sends 412 response.
Trailer	Makes individual trailing headers available to application through WEB READ HTTPHEADER command.	Chunked messages are not suitable for a static response.
Transfer-Encoding	For "chunked", receives all chunks and assembles into single message to pass to application. For anything other than "chunked", sends 501 (Not Implemented) response to Web client. The Transfer-Encoding header remains on the message, but it is for information only.	Chunked messages are not suitable for a static response.
Warning	Writes warning text to the TD queue CWBW. If more than 128 characters are used, the warning text is truncated.	As for application-generated response.

CICS as an HTTP server: Headers that CICS writes for an HTTP response

Table 4 on page 108 shows the headers that CICS writes when responding to a request from a Web client, the HTTP for which the headers are used, and the source of the information that CICS provides in the header.

Table 4. CICS-written headers for an HTTP response

Header written by CICS	HTTP Version	Source where response is handled by user application program	Source where response is provided by static document
Connection	1.0 and 1.1	CLOSESTATUS option on WEB SEND command. If no close is specified, and client is HTTP/1.0, Keep-Alive is sent. If close is specified, Connection: close is sent, or for HTTP/1.0 client Keep-Alive is omitted.	Keep-Alive is sent on static responses
Content-Length(unless chunked transfer-coding is used)	1.0 and 1.1	Where response body is a buffer of data, the length is taken from the FROMLENGTH option on the WEBSSEND command. (CICS checks the length that you specify. If it is wrong, CICS sends the response but omits the Connection: Keep-Alive header.) Where response body is a CICS document, the length is calculated by CICS.	Calculated by CICS.
Content-Type	1.0 and 1.1	MEDIATYPE option on WEB SEND command, and character set for response body. (Header is only created when the MEDIATYPE option was specified.)	Character set for response body.
Date	1.0 and 1.1	Current date and time generated by CICS.	Current date and time generated by CICS.
Server	1.0 and 1.1	Preset to "IBM_CICS_Transaction_Server/2.2.0(zVSE)".	Preset to "IBM_CICS_Transaction_Server/2.2.0(zVSE)".
Transfer-Encoding	1.1 only	CHUNKING option on WEB SEND command.	Not used.

HTTP status code reference for CICS Web support

HTTP status codes are provided to clients by a server, to explain the consequence of the client's request. When CICS is an HTTP server, depending on the circumstances, either CICS Web support or the user application program selects an appropriate status code for each response.

For full information about the meaning and correct use of status codes, you should consult the HTTP specification to which you are working.

This topic provides a brief summary of the HTTP/1.1 status codes as they relate to CICS Web support. When you are selecting status codes to be sent through the Web error programs, or directly from a user application, it is important to check the HTTP specification to which you are working. The HTTP specification provides detailed guidance and requirements about how you should use status codes, such as what should be the content of the response body, and what HTTP headers should be included.

Status codes for responses sent by CICS (when CICS is an HTTP server)

- CICS Web support generates a response to a Web client in the following circumstances:
 - When CICS Web support detects a problem in initial processing of a request from a Web client; for example, if required information is missing from the request, or if the request is sent too slowly and the receive timeout is reached.

- When the analyzer specified for the TCPIP SERVICE definition is unable to process the request and passes control to a Web error program.
- When neither the analyzer nor the converter program processing, manages to determine what application program should be executed to service the request.
- When an abend occurs in the analyzer program, converter program, or user-written application program. This ensures that a response can be returned to the Web client even though processing has failed.

In these situations, CICS selects an appropriate status code and creates a default error response. Table xx describes the status codes used by CICS for these purposes. Note that CICS does **not** generate a response in situations where the user-written application program has completed processing successfully and wants to return a response indicating an error; for example, where the client has specified a method not supported for the resource. The user-written application creates the response in this case.

- For most CICS-generated responses with 4xx and 5xx status codes, the response sent to the Web client can be modified by tailoring the user-replaceable Web error program DFHWBEP. CICS-generated responses involving 1xx, 2xx and 3xx status codes cannot be modified. The Web error programs can change the status code, reason phrase, HTTP headers and message body for the response. When you modify the Web error programs, ensure that your selection of status code and response content is made according to the requirements in the HTTP specification to which you are working. The CICS *Internet Guide*, section “Web error program,” explains how to tailor the Web error programs.
- A user application program that responds to a client's request needs to select a suitable status code for the response. The status code can convey the following messages to a Web client:
 - The request has completed as expected.
 - There is an error that prevents fulfillment of the request.
 - The client needs to do something else in order to complete its request successfully. This could involve following a redirection URL, or amending the request so that it is acceptable to the server.

The status code influences the other content of the response, that is, the message body and HTTP headers.

CICS as an HTTP server: Status codes that CICS provides to Web clients

Table 5 on page 109 shows the status codes used in situations in which CICS provides a response to a Web client's request. Some of these responses can be tailored by modifying the Web error programs. A user application program may also use many of the status codes listed here. Some status codes are only appropriate for HTTP/1.1 clients. CICS does not return these status codes to HTTP/1.0 clients.

<i>Table 5. Status Codes for CICS-generated responses sent to Web clients</i>			
Status code and reason phrase provided	Sent to HTTP/1.0 clients?	Situation(s) in which this response is provided	Can be modified in Web error program?
100 Continue	No	Web client sent an Expect header.	No
200 OK	Yes	Delivery of normal response.	No
400 Bad Request	Yes	Syntax error in request (such as request line wrongly specified, request incomplete) OR Host header is not supplied (HTTP/1.1 only).	Yes
404 Not Found (some situations: Program Not Found, File Not Found)	Yes	The program specified to respond to the request is not found.	Yes

Table 5. Status Codes for CICS-generated responses sent to Web clients (continued)

Status code and reason phrase provided	Sent to HTTP/1.0 clients?	Situation(s) in which this response is provided	Can be modified in Web error program?
408 Request Timeout	No	Receive timeout for request has been exceeded. This is determined by the SOCKETCLOSE attribute in the TCPIP SERVICE definition for the port.	Yes
412 Precondition Failed	No	If-Unmodified-Since header was used on request.	Yes
417 Expectation Failed	No	Expect header received which did not have value"100-continue".	No
500 Internal Server Error	Yes	Abend in one of the programs involved with processing the request and providing the response	Yes
501 Method Not Implemented	Yes	Method is not supported by CICS for this HTTP version.(Includes methods that are supported but not in the way the client requests, such as OPTIONS requests that cite a specific resource.) OR Media type for request is "multipart/byteranges",which is not supported. OR Transfer coding for request is other than "chunked". (Note: Connection is closed by CICS.)	Yes
505 Version Not Supported	No	HTTP version is higher than 1.1, and method is not recognized for highest version supported by CICS.	Yes

CICS as an HTTP server: Status codes in user applications

Table 6 on page 110 shows each status code, describes its relevance for a user application, and suggests appropriate actions, in accordance with the recommendations in the HTTP/1.1 specification. Remember that CICS does not take any specific action that might be implied by these status codes, and that CICS does not generally check their validity against the content of the message. You should ensure that the status codes are correct and that you have taken any necessary action. Ensure that you check the HTTP specification to which you are working, for further information and requirements that apply to each status code.

Table 6. Status codes for user-written responses sent to Web clients

Status code and usual reason phrase	Suitable for HTTP/1.0 client?	Situation(s) in which you might provide this response	Effect on message body and HTTP headers (where status code is appropriate for a user application). See HTTP specification for more information.
100 Continue	No	Do not use. CICS handles Expect requests and sends 100-Continue response itself.	

Table 6. Status codes for user-written responses sent to Web clients (continued)

Status code and usual reason phrase	Suitable for HTTP/1.0 client?	Situation(s) in which you might provide this response	Effect on message body and HTTP headers (where status code is appropriate for a user application). See HTTP specification for more information.
101 Switching Protocols	No	Do not use. CICS does not support upgrades in HTTP version or security protocol.	
200 OK	Yes	You have fulfilled the request. A normal response.	Provide normal response body.
201 Created	Yes	You have created a new resource. (Use 202 Accepted if the resource has not yet been created.)	Message body content and one or more headers required.
202 Accepted	Yes	You have accepted the request but have not yet processed it, and do not guarantee to process it.	Message body content required.
203 Non-Authoritative Information	No	Do not use. The headers that you supply will give authoritative information.	
204 No Content	Yes	You are not sending a message body, perhaps because you only need to send updated headers.	No message body permitted.
205 Reset Content	No	You want the client to clear the form that initiated the request.	No message body permitted.
206 Partial Content	No	You support byte range requests, and this response fulfills the request.	Normal response body. One or more headers required.
300 Multiple Choices	Yes	You are able to provide more than one version of the resource (for example, documents in different languages).	Message body content and one or more headers required.
303 See Other	No	You want client to make a GET request for another resource that gives a response (in particular, a response about the out come of a POST request).	Message body content and one or more headers required.
304 Not Modified	Yes	The client made a conditional request, and the resource you are providing has not changed. Note that a response that is built dynamically by an application is likely to be modified on every request.	No message body permitted. One or more headers required.
400 Bad Request	Yes	The client's request contains syntax errors or similar problems, and you cannot process it.	Message body content required.
403 Forbidden	Yes	You are refusing the client's request.	Message body content required.

Table 6. Status codes for user-written responses sent to Web clients (continued)

Status code and usual reason phrase	Suitable for HTTP/1.0 client?	Situation(s) in which you might provide this response	Effect on message body and HTTP headers (where status code is appropriate for a user application). See HTTP specification for more information.
404 Not Found	Yes	You do not have a resource to respond to the request; or you want to refuse the request without explanation; or no other status code is relevant	Message body content required.
405 Method Not Allowed	No	The client used a method which is not supported for this resource.	Message body content and one or more headers required.
406 Not Acceptable	No	The client made a conditional request using Accept headers, but you do not have a version of the resource that meets their criteria. Note that as an alternative to using this status code, you can send a response which does not meet the conditions.	Message body content required.
408 Request Timeout	No	Not recommended for issuing by user application. Timeout should be specified for handling by CICS Web support using the SOCKETCLOSE attribute on the TCPIP SERVICE definition.	
409 Conflict	No	The resource has been changed and the client's request cannot be applied to the resource as it now stands.	Message body content required.
410 Gone	No	The resource is permanently unavailable.	Message body content required.
411 Length Required	No	Do not use. CICS requires HTTP/1.1 requests to specify the Content-Length header for successful socket receive.	
412 Precondition Failed	No	The client made a conditional request and the conditions were not met.	Message body content required.
413 Request Entity Too Large	No	Not recommended for issuing by user application. Request size limit should be specified for handling by CICS Web support using the MAXDATALEN attribute on the TCPIP SERVICE definition.	
414 Request URI Too Long	No	The client's request URL is too large for your application to process.	Message body content required.
415 Unsupported Media Type	No	The message body sent by the client has a media type or content coding that you do not support	Message body content required.

Table 6. Status codes for user-written responses sent to Web clients (continued)

Status code and usual reason phrase	Suitable for HTTP/1.0 client?	Situation(s) in which you might provide this response	Effect on message body and HTTP headers (where status code is appropriate for a user application). See HTTP specification for more information.
416 Requested Range Not Satisfiable	No	The client made a request using the Range header field (but not the If-Range header field), and although you support byte-ranges, that range was not present in the resource.	Message body content and one or more headers required.
417 Expectation Failed	No	Do not use. CICS handles Expect requests.	
500 Internal Server Error	Yes	You cannot handle the request because of an application or system error.	Message body content required.
501 Not Implemented	Yes	The method for the client's request is not supported. This status code should only be issued where the client is HTTP/1.0, or you are using the USER protocol. For the HTTP protocol, during initial processing, CICS rejects any requests with methods that are not recognized. If the method is recognized but does not apply for the resource, 405 Method Not Allowed should be used for HTTP/1.1 clients.	Message body content required.
503 Service Unavailable	Yes	A user application is unlikely to be in a relevant situation to use this status code, unless it needs to access another application or system which is temporarily unavailable.	Message body content and one or more headers required.
505 HTTP Version Not Supported	No	Do not use. CICS matches HTTP version of response to HTTP version of client's request.	

IANA media types and character sets

The Internet Assigned Numbers Authority (IANA) is the international body responsible for assigning names for protocols used on the Internet.

- IANA media types are names for the types of data that are commonly transmitted over the Internet. They are described at <http://www.iana.org/assignments/media-types/>.

Text media types (such as a type that begins with text/, or a type that contains +xml) are identified by RFC 3023, which is available at <http://www.ietf.org/rfc/rfc3023.txt>.

- IANA character sets are the names of character set registries. They are described at <http://www.iana.org/assignments/character-sets>.

CICS does not support all the IANA character sets for code page conversion. The character sets that CICS supports are described in [“HTML coded character sets”](#) on page 114.

HTML coded character sets

This reference lists the supported IANA-registered character set names (specified as charset= values in HTTP headers), and the IBM CCSID equivalents.

All of these values are valid for code page conversion options on the following commands:

- WEB RECEIVE
- WEB SEND
- WEB CONVERSE
- DOCUMENT RETRIEVE
- WEB READ FORMFIELD
- WEB STARTBROWSE FORMFIELD

Table 7. Coded character sets

Language	Coded character set	IANA charset	IBM CCSID
Albanian	ISO/IEC 8859-1	iso-8859-1	819
Arabic	ISO/IEC 8859-6	iso-8859-6	1089
Bulgarian	Windows 1251	windows-1251	1251
Byelorussian	Windows 1251	windows-1251	1251
Catalan	ISO/IEC 8859-1	iso-8859-1	819
Chinese (simplified)	GB	gb2312	1381 or 5477
Chinese (traditional)	Big 5	big5	950
Croatian	ISO/IEC 8859-2	iso-8859-2	912
Czech	ISO/IEC 8859-2	iso-8859-2	912
Danish	ISO/IEC 8859-1	iso-8859-1	819
Dutch	ISO/IEC 8859-1	iso-8859-1	819
English	ISO/IEC 8859-1	iso-8859-1	819
Estonian	ISO/IEC 8859-1	iso-8859-1	819
Finnish	ISO/IEC 8859-1	iso-8859-1	819
French	ISO/IEC 8859-1	iso-8859-1	819
German	ISO/IEC 8859-1	iso-8859-1	819
Greek	ISO/IEC 8859-7	iso-8859-7	813
Hebrew	ISO/IEC 8859-8	iso-8859-8	916
Hungarian	ISO/IEC 8859-2	iso-8859-2	912
Italian	ISO/IEC 8859-1	iso-8859-1	819
Japanese	Shift JIS	x-sjis or shift-jis	943 (932, a subset of 943, is also valid)
EUC Japanese	EUC Japanese	euc-jp	5050 (EUC)
Korean	EUC Korean	euc-kr	970 (for AIX or Unix)
Latvian	Windows 1257	windows-1257	1257

Table 7. Coded character sets (continued)

Language	Coded character set	IANA charset	IBM CCSID
Lithuanian	Windows 1257	windows-1257	1257
Macedonian	Windows 1257	windows-1257	1251
Norwegian	ISO/IEC 8859-1	iso-8859-1	819
Polish	ISO/IEC 8859-2	iso-8859-2	912
Portuguese	ISO/IEC 8859-1	iso-8859-1	819
Romanian	ISO/IEC 8859-2	iso-8859-2	912
Russian	Windows 1251	windows-1251	1251
Serbian (Cyrillic)	Windows 1251	windows-1251	1251
Serbian (Latin 2)	Windows 1250	windows-1250	1250
Slovakian	ISO/IEC 8859-2	iso-8859-2	912
Slovenian	ISO/IEC 8859-2	iso-8859-2	912
Spanish	ISO/IEC 8859-1	iso-8859-1	819
Spanish	ISO/IEC 8859-15	iso-8859-15	923
Swedish	ISO/IEC 8859-1	iso-8859-1	819
Turkish	ISO/IEC 8859-9	iso-8859-9	920
Ukrainian	Windows 1251	windows-1251	1251
Unicode	UTF-16	utf-16	1200
Unicode	UTF-16 big-endian	utf-16be	1201
Unicode	UTF-16 little-endian	utf-16le	1202
Unicode	UTF-8	utf-8	1208

Related concepts

When CICS exchanges messages with a Web client or server, character data in the messages normally needs to undergo code page conversion on entering and leaving the CICS environment.

Reference Information for DFHWBADX

This section provides reference information for analyzer programs, including input and output parameters, and responses and reason codes.

Attention: This section contains Product-sensitive Programming Interface and Associated Guidance Information.

Summary of parameters for analyzer programs

The names of the parameters and constants for analyzer programs, translated into appropriate forms for the different programming languages supported, are defined in files supplied as part of CICS.

Summary of parameters for analyzer programs

Table 8. Names of parameter and constants for analyzer programs as defined in files

Language	Parameters file	Constants file
Assembler	DFHWBTDD	DFHWBUCD
C	DFHWBTDH	DFHWBUCH
COBOL	DFHWBTDO	DFHWBUCO
PL/I	DFHWBTDL	DFHWBUC

These files give language-specific information about the data types of the fields in the COMMAREA. If you use these files you must specify XOPTS(NOLINKAGE) on the Translator step; failure to do this causes the compile to fail.

In the following table, the names of the parameters are given in abbreviated form: each name in the table must be prefixed with **wbra** to give the name of the parameter.

Table 9. Abbreviated parameter names

Input wbra_	Inout wbra_	Output wbra_
client_ip_address	alias_tranid ⁽²⁾	application_style ⁽¹⁾
content_length	converter_program ⁽²⁾	alias_termid ⁽¹⁾
eyecatcher	server_program ⁽²⁾	charset ⁽¹⁾
function	user_data_length	commarea ⁽¹⁾
hostname_length ⁽¹⁾	userid	dfhcnv_key
hostname_ptr ⁽¹⁾		hostcodepage ⁽¹⁾
method_ptr		reason
method length		response
querystring_length ⁽¹⁾		unescape
querystring_ptr ⁽¹⁾		user_token
request_header_length		
request_header_ptr		
request_type		
resource_escaped_ptr ⁽¹⁾		
resource_length		
resource_ptr		
server_ip_address		
urimap ⁽¹⁾		
user_data_ptr		

(1) New with CICS TS for z/VSE 2.2.

(2) With CICS TS for z/VSE 2.2 changed from OUTPUT to INOUT.

Parameters for analyzer programs

wbra_alias_tranid

(Input and output)

A string of length 4. The transaction ID of the alias transaction that is to cover the remainder of processing for this request. If you do not set this field, or if you set it to blanks, CWBA is used.

wbra_alias_termid

(Output only)

A string of length 4. The terminal ID to be used on the START request for the alias transaction that is to cover the remainder of processing for this request.

wbra_characterset

(Output only)

The name of the IANA character set that the client used for the entity body of the request. This information is used for code page conversion of the entity body of the request and the response. If the request is not an HTTP request, this character set is used to translate the entire request and response.

wbra_hostcodepage must also be supplied.

wbra_client_ip_address

(Input only)

The 32-bit IP address of the client.

wbra_commarea

(Output only)

The flag to indicate that pre-CICS TS Version 2 Release compatibility processing is required for a response that uses a non-Web-aware application and a converter program. This flag means that the Web client receives a response identical with the response it would have received before CICS TS Version 2 Release 2.

wbra_content_length

(Input only)

A 32-bit binary representation of the entity body length as specified by the Content-Length HTTP header in the received data.

wbra_converter_program

(Input and output)

A string of length 8. The name of the converter program that is used to process the request. If this field is not set on output, no converter program is called.

wbra_dfhcnv_key

(Output only)

A string of length 8. The name of a conversion template in the DFHCNV table for code page conversion of the entity body for the request and the response. If the request is not an HTTP request, this template is used to translate the entire request and response.

CICS initializes this field to high values. If you use this field to specify a conversion template, the name you choose must be defined in the DFHCNV table. As an alternative, you can set the **wbra_hostcodepage** and **wbra_characterset** fields to specify the pair of code pages to use for code page conversion. If you set **wbra_dfhcnv_key** to nulls or blanks and do not set **wbra_hostcodepage** and **wbra_characterset**, code page conversion is suppressed.

wbra_eyecatcher

(Input only)

A string of length 8. Its value is ">analyze".

wbra_function

(Input only)

A code indicating that an analyzer program is being called. The value is 1.

wbra_hostcodepage

(Output only)

Summary of parameters for analyzer programs

The name of a host code page (IBM EBCDIC code page) suitable for the application program that is handling the request. This information is used for code page conversion of the entity body of the request and the response. If the request is not an HTTP request, this code page is used to translate the entire request and response. **wbra_characterst must also be supplied.**

wbra_hostname_length

(Input only)

The length in bytes of the host name specified on the HTTP request. If no host name was specified, the value is undefined.

wbra_hostname_ptr

(Input only)

A pointer to the host name specified on the HTTP request sent by the client. If an absolute URI was used for the request, the host name is taken from the URI. Otherwise the host name is as specified in the Host header for the request. For HTTP/1.1 requests, a host name is required, so this parameter is always passed to the analyzer. For HTTP/1.0 requests, a host name might not be supplied, in which case the value is undefined.

wbra_http_version_length

(Input only)

For an HTTP request, the length in bytes of the string identifying the HTTP version of the client's request. If the request is not an HTTP request, the value is zero.

wbra_http_version_ptr

(Input only)

For an HTTP request, a pointer to the string identifying the HTTP version of the client's request. If the request is not an HTTP request, the value is undefined.

wbra_method_length

(Input only)

For an HTTP request, the length in bytes of the string identifying the method specified in the HTTP request. If the request is not an HTTP request, the value is zero.

wbra_method_ptr

(Input only)

For an HTTP request, a pointer to the method specified in the HTTP request. If the request is not an HTTP request, the value is undefined.

wbra_querystring_length

(Input only)

The length in bytes of the query string specified on the HTTP request. If no query string was sent, the value is undefined.

wbra_querystring_ptr

(Input only)

A pointer to the query string specified on the HTTP request sent by the client. If no query string was sent, the value is undefined.

wbra_reason

(Output only)

The reason code returned by the analyzer program. See "Responses and reason codes" below.

wbra_request_header_length

(Input only)

For an HTTP request, the length of the first HTTP header in the HTTP request. If the request is not an HTTP request, the value is zero.

wbra_request_header_ptr

(Input only)

For an HTTP request, a pointer to the first HTTP header in the HTTP request. The other HTTP headers follow this one in the request buffer. If the request is not an HTTP request, the value is undefined.

wbra_request_type

(Input only)

If this is an HTTP request, the value is `WBRA_TYPE_HTTP`. If this is not an HTTP request, the value is `WBRA_TYPE_NON_HTTP`.

wbra_resource_escaped_ptr

(Input only)

For an HTTP request, a pointer to a copy of the HTTP headers for the request which have not been unescaped (that is, are still in their escaped form).

wbra_resource_length

(Input only)

For an HTTP request, the length in bytes of the path component of the URL. If the request is not an HTTP request, the value is zero.

wbra_resource_ptr

(Input only)

For an HTTP request, a pointer to the path component of the URL. If the request is not an HTTP request, the value is undefined.

wbra_response

(Output only)

The response value produced by the analyzer program. See “Responses and reason codes” below.

wbra_server_ip_address

(Input only)

The 32-bit IP address that is being used by CICS as an HTTP server.

wbra_server_program

(Input and output)

A string of length 8. The name of a CICS application program that is to process the request. The program name is passed to any converter program specified in **wbra_converter_program**. If you do not set this field, the value passed is nulls. The program name must be set here or by the converter program, or no CICS application program will be called.

wbra_unescape

(Output only)

- To specify that data is to be passed to the CICS application program in its unescaped form, set this parameter to `WBRA_UNESCAPE_REQUIRED`
- To specify that data is to be passed to the application in its escaped form, set this parameter to `WBRA_UNESCAPE_NOT_REQUIRED`. This is the default value.

You should also set the parameter to `WBRA_UNESCAPE_NOT_REQUIRED` if your analyzer has converted the data to its escaped form.

wbra_urimap

(Input only)

Not used with CICS TS for z/VSE.

wbra_user_data_length

(Input and output)

Summary of parameters for analyzer programs

A 15-bit integer, representing the length of the entity body in the HTTP request. If the request is non-HTTP, this value is the length of the request.

The length passed to the analyzer includes any trailing carriage return and line feed (CRLF) characters that may delimit the end of the entity body. If the analyzer reduces the length of the entity body, the newly redundant bytes are replaced by null characters, X'00'. The modified value is passed on to the CICS business logic interface in field **wbbl_user_data_length**, and to the converter program in field **decode_user_data_length**.

wbra_user_data_ptr

(Input only)

For an HTTP request, a pointer to the entity body in the HTTP request. If the request is not an HTTP request, this is a pointer to the request.

wbra_user_token

(Output only)

A 64-bit token that is passed to the converter program as **decode_user_token**. If you do not set this field, the value passed is null. If there is no converter program for this request, the value is ignored.

wbra_userid

(Input and output)

A string of length 8. On input, this is a user ID supplied by the client (using basic authentication or client certificate authentication). On output, it is the user ID that is used for the alias transaction, which can be the supplied user ID or a user ID chosen by the analyzer program. If this field is blank or null on output, the CICS default user ID is used.

Responses and reason codes

An analyzer program must return one of the following values in **wbra_response**:

Symbolic value	Numeric value	Explanation
URP_OK	0	The alias transaction is started.
URP_EXCEPTION	4	The alias transaction is not started. Web attach processing writes an exception trace entry (trace point 4510), and issues a message (DFHWB0723). If the request is an HTTP request, an error response is sent to the Web client. The default status code is 400 (Bad request), and this can be configured using the user-replaceable Web error program DFHWBEP. If the request is not an HTTP request, no response is sent, and the socket is closed.

<i>Table 10. Returned values in wbra_response (continued)</i>		
Symbolic value	Numeric value	Explanation
URP_INVALID	8	The alias transaction is not started. The server controller writes an exception trace entry (trace point 4510), and issues a message (DFHWB0723). If the request is an HTTP request, an error response is sent to the Web client. The default status code is 400 (Bad request), and this can be configured using the user-replaceable Web error program DFHWBEP. If the request is not an HTTP request, no response is sent, and the socket is closed
URP_DISASTER	12	The alias transaction is not started. CICS writes an exception trace entry (trace point 4510), and issues a message (DFHWB0723). If the request is an HTTP request, an error response is sent to the Web client. The default status code is 400 (Bad request), and this can be configured using the user-replaceable Web error program DFHWBEP. If the request is not an HTTP request, no response is sent, and the socket is closed.

If you return any other value in **wbra_response**, the server controller writes an exception trace entry (trace point 4510), and issues a message (DFHWB0723). If the request is an HTTP request, a message with status code 400 (Bad request) is sent to the Web client. If the request is not an HTTP request, no response is sent, and the socket is closed.

You may supply a 32-bit reason code in **wbra_reason** to provide further information in error cases. CICS Web support does not take any action on the reason code returned by an analyzer program, but the user-replaceable Web error program DFHWBEP can use it to decide how to modify the default response. The reason code is output in any trace entry that results from the invocation of an analyzer program, and in message DFHWB0723.

IBM-supplied DFHWBADX responses and reason codes

The meanings of the responses produced by the IBM-supplied default analyzer DFHWBADX are as follows:

URP_OK

The analyzer found that the request conformed to the default HTTP request format, and generated the appropriate outputs for the alias.

URP_EXCEPTION

The analyzer found that the request did not conform to the default format. A reason code is supplied as follows:

Summary of parameters for analyzer programs

1

The length of the resource was less than 6. (The shortest possible resource specification is /A/B/C, asking for program C to be run under transaction B with converter A.) This response and reason are the ones used when the incoming request is not an HTTP request.

2

The resource specification did not begin with a “/”.

4

The length of the converter name in the resource specification was 0 or more than 8.

5

The length of the transaction name in the resource specification was 0 or more than 4.

6

The length of the CICS program name in the resource specification was 0 or more than 8.

8

One of the following:

- There is no second “/”
- There is nothing after the second “/”
- There is no third “/”
- There is nothing after the third “/”

URP_INVALID

The eye-catcher was invalid. This is an internal error.

Part 3. Using Secure Sockets Layer (SSL)

Chapter 20. Introduction to Secure Sockets Layer (SSL)

One of the major concerns when providing commercial services on the Internet is providing for transaction security and communications security.

Information exchanges are secure if all the following are true:

- Messages are confidential.
- The information exchange has integrity.
- Both sender and receiver are accountable.
- Both parties in the exchange can be authenticated.

Secure Sockets Layer (usually referred to as simply *SSL*) is a security protocol that was developed by Netscape Communications Corporation, along with RSA Data Security, Inc. The primary goal of the SSL protocol is to provide a private channel between communicating applications that ensures privacy of data, authentication of the partners, and integrity.

SSL is a protocol layer implemented on top of the standard TCP/IP socket API, which has security implemented within it. There are APIs for implementing SSL support for most languages (C language, Java, and so on).

This chapter consists of these main topics:

- [“Overview of SSL” on page 125](#)
- [“SSL and the Web” on page 126](#)
- [“Encryption and Keys” on page 126](#)
- [“Authentication and Certificates” on page 126](#)
- [“The Role of Certificate Authorities” on page 127](#)

Overview of SSL

SSL is a **handshake protocol** developed to provide security and privacy over the Internet. The SSL protocol uses encryption and authentication to ensure:

Privacy

The data to be exchanged between the client and the server is encrypted, so that only the intended recipient can read it. SSL uses public key encryption as a secure mechanism to distribute a secret key between the server and the client. Public key encryption is a technique that uses a pair of asymmetric keys for encryption and decryption. With SSL, one or more keys (public key, private key, session keys) are passed between the client and the server, using public key cryptography. These keys are then used to encrypt and decrypt all traffic along the SSL connection. This encryption protects the data from other parties trying to eavesdrop, as no other parties will have the secret key needed to decrypt the data. This ensures that private information, such as a credit card number, is transferred securely.

Integrity

The message transport includes a message integrity check based on a secure hashing algorithm. This algorithm is performed when the message is sent, and again when it is received. If the two hash values do not match, the receiver is warned that the message may have been tampered with.

Authentication

When a client establishes a connection with CICS, it may be required to authenticate its details to the server. The authentication mechanism is based on the exchange of digital certificates (X.509v3 certificates). These digital certificates contain information about an entity, such as the system name and public key, and the server's digital signature. Digital certificates are issued by a Certificate Authority, and encrypted using the Certificate Authority's private key. If you can decrypt the certificate

using the Certificate Authority's public key, you know that the information contained within the certificate can be trusted (that is, that the certificate really does belong to whoever claims to own it).

SSL and the Web

The HTTPS protocol is a variant of HTTP for handling secure Web transactions. Most current browsers support the HTTPS URL access method to connect to HTTP servers that use SSL. A secure connection is made with a URL such as

```
https://www.company.com
```

If you use the HTTPS protocol without specifying a port number, a default port number of 443 is assumed.

Encryption and Keys

The SSL protocol operates between the application layer and the TCP/IP layer. This allows it to encrypt the data stream itself, which can then be transmitted securely, using any of the application layer protocols. Two encryption techniques are used:

- Public key cryptography standard (PKCS), which encrypts and decrypts certificates during the SSL handshake. Encryption keys are created in pairs, a public key and its associated private key. Data encrypted with a given public key can be decrypted only with the associated private key; this means that data is readable by only the intended recipient. Data encrypted with a given private key can be decrypted only with the associated public key; this means that authentication data is assured to originate from the owner of the private key.
- A mutually agreed symmetric encryption technique, such as DES (data encryption standard), or triple DES, is used in the data transfer following the handshake.

PKCS, as used by SSL, works briefly as follows:

1. A private and public key pair is requested, usually as part of a certificate application (see [“Authentication and Certificates”](#) on page 126 for details).
2. As part of the certificate creation, a private key and public key are created by means of an algorithm based on two random numbers. The private and public keys which result are related to each other such that:
 - **It is not feasible to deduce the value of the private key from the public key, nor the public key from the private key**

The private key is stored securely, and is not made known to anyone but its owner. The public key can be made freely available to any user, with no risk of compromising the security of the private key.

- **Information encrypted using the public key can be decrypted only with the private key**

Information can be encrypted by any user, and sent securely to the holder of the private key. A third party cannot use the public key to read the information.

- **Information encrypted using the private key can be decrypted only with the public key**

Only the holder of the private key can encrypt information that can be decrypted with the public key. A third party cannot pass as the sender of the information.

Authentication and Certificates

To make an environment secure, you must be sure that any communication is with "trusted" sites whose identity you can be sure of. SSL uses *certificates* for authentication — these are digitally signed documents which bind the public key to the identity of the private key owner. Authentication happens at connection time, and is independent of the application or the application protocol. Authentication involves making sure that sites with which you communicate are who they claim to be. With SSL, authentication is performed by an exchange of certificates, which are blocks of data in a format described in ITU-T

standard X.509. The X.509 certificates are digitally signed by an external authority known as a certificate authority.

Certificates are digitally signed using the public-key encryption technique. The signature is created by partially encrypting the certificate with the certificate authority's private key. A user of the certificate is assured of the origin of the certificate when it is successfully decrypted by the certificate authority's public key.

Client authentication

When SSL is used, the *server certificate* which is used for *server authentication*, is mandatory. However, the *client certificate* which is used for *client authentication*, is optional: some clients may not support client certificates; other may not have certificates installed. Servers can decide whether to require client authentication for a connection.

The Role of Certificate Authorities

In order that one system can be assured that a certificate received from another system is genuine, a trusted third party that can vouch for the certificate is needed. Therefore *Certificate authorities* are independent bodies who act as the trusted third parties, by issuing certificates for use by others. Before issuing a certificate, a certificate authority will examine the credentials of the person or organisation that has requested the certificate. When the certificate has been issued, information about it is held on a publicly accessible repository. Users can consult the repository to check the status and validity of any certificates received.

Certificate authorities issue several levels of security certificates for different purposes. For example:

- Secure e-mail
- Client authentication
- Server authentication

Chapter 21. Configuring CICS to use SSL

This topic explains how to configure CICS to use Secure Sockets Layer (SSL).

It contains these main topics:

- [“Hardware prerequisites” on page 129](#)
- [“Software prerequisites” on page 129](#)
- [“Step 1: Decide Which Authorization Level You Require” on page 129](#)
- [“Step 2: Define the System Initialization Parameters” on page 130](#)
- [“Step 3: Define a TCPIPSERVICE Resource” on page 130](#)
- [“Step 4: Set Up Your z/VSE System for SSL Support” on page 131](#)
- [“Step 5: Configure for Server Authentication” on page 131](#)
- [“Step 6: Configure for Client Authentication” on page 132](#)
- [“Step 7: Configure for Client Certificate Mapping” on page 132](#)
- [“Application programming considerations” on page 132](#)
- [“A sample application program: DFHOWBCA” on page 132](#)
- [“Currently-Supported SSL Cipher Suites” on page 132](#)

Hardware prerequisites

Refer to the [z/VSE Administration](#), SC34-2692 publication regarding hardware crypto support.

Software prerequisites

If you configure CICS Web support to use SSL, you need an SSL/TLS implementation.

These are the software prerequisites for using SSL:

- Up to z/VSE 6.1 and CICS TS for z/VSE 2.1 you can only use TCP/IP for z/VSE and its SSL implementation.
- Starting with z/VSE 6.2 and CICS TS for z/VSE 2.2, you can alternatively use the OpenSSL implementation of z/VSE. This allows you to use any TCP/IP stack (TCP/IP for z/VSE, IBM IPv6/VSE, or Fast Path to Linux on z Systems) for CICS Web support with SSL.

If you use IPv6/VSE or the Fast Path to Linux on z Systems without the OpenSSL implementation of z/VSE, you cannot configure CICS to use SSL.

For information on how to set up SSL with these stacks, refer to the [z/VSE TCP/IP Support](#), SC34-2706 publication. If you use CICS Web support without SSL, you can use any TCP/IP stack.

Step 1: Decide Which Authorization Level You Require

You can decide to use:

- Server authentication only (which is mandatory).
- Server authentication *and* client authentication.

Server authentication is the process by which the CICS client (Web browser) can check that a server (in this case the z/VSE host) is “who he claims to be”, during the SSL “handshake”.

Client authentication is the process by which the server (in this case the z/VSE host) can check that the CICS client (Web browser) is “who he claims to be”, during the SSL “handshake”. See also [“Authentication and Certificates” on page 126](#) for further details.

For applications such as flight booking systems, where the number and identity of potential clients is unknown, client authentication is probably not practical. In this case, *server authentication only* would be performed. However, you must carefully consider which level of authentication you require before making a decision.

If you decide that:

- *Server authentication only* is required, perform Steps 3 to 5 below, ensuring you set SSL (YES) in the TCPIP SERVICE profile.
- *Server and client authentication* is required, perform Steps 3 to 7 below, ensuring you set SSL (CLIENTAUTH) in the TCPIP SERVICE profile.

(TCPIP SERVICE is described in [“Step 3: Define a TCPIP SERVICE Resource”](#) on page 130).

Step 2: Define the System Initialization Parameters

There are *four* system initialization parameters relating to SSL:

- ENCRYPTION={WEAK|Normal|STRONG|SSLV3}, described in [“ENCRYPTION={STRONG|SSLV3}”](#) on page 331 of the topic “CICS system initialization parameters”.
- KEYFILE=*name*, described in [“KEYFILE=*name*”](#) on page 338 of the topic “CICS system initialization parameters”.
- SSLDELAY=600|*number*, described in [“SSLDELAY=600|*number*”](#) on page 355 of the topic “CICS system initialization parameters”.
- TCPIP=YES, described in [“TCPIP={NO|YES}”](#) on page 360 of the topic “CICS system initialization parameters”.

Step 3: Define a TCPIP SERVICE Resource

There are three TCPIP SERVICE resource definition attributes relating to SSL:

1. CERTIFICATE(*certificate-label*)
2. PORTNUMBER
3. SSL(NO|YES|CLIENTAUTH)

See [Chapter 28, “CEDA DEFINE TCPIP SERVICE,”](#) on page 161 for a description of the above three attributes.

Note: If you do not enter a PORTNUMBER in your CICS client (Web browser), your CICS client will expect the default PORTNUMBER **443** (see [“SSL and the Web”](#) on page 126).

You must activate the TCPIP SERVICE definition either by:

- Specifying STATUS (OPEN) and installing the definition.
- Installing the definition and later using a CEMT SET TCPIP SERVICE OPEN command (described in [Chapter 28, “CEDA DEFINE TCPIP SERVICE,”](#) on page 161).

Figure 8 on page 131 shows an example of the entries you make when defining a TCPIP SERVICE definition for use with SSL:

```

OVERTYPE TO MODIFY
CEDA DEFine TCPIPService( CMCIT )
TCPIPService ==> CMCIT
Group ==> CMCI
Description ==> CICS EXPLORER
Urm ==> DFHWBADDY
Portnumber ==> 27283 1-65535
Certificate ==>
SStatus ==> Open Open | Closed
SSL ==> No Yes | No | Clientauth
Attachsec ==> Local | Verify
TRansaction ==> CWXN
Backlog ==> 00001 0-32767
TSqprefix ==>
Ippaddress ==> 9.152,131.141
SOcketclose ==> No No | 0-24000
PRotocol ==> HTTP Http | Eci | User
Maxdatalen ==> 000032 3-524288
CICS RELEASE = 0430

```

Figure 8. Example of a TCPIPService resource definition for SSL

Step 4: Set Up Your z/VSE System for SSL Support

You should ensure that all steps described in the following topic have been completed:

"Preparing Your System to Use TLS" in the [z/VSE Administration, SC34-2692](#).

The above topic contains information about how to:

- Activate TCP/IP.
- Create a client keyring file on your CICS clients.
- Download and customize the IBM *Keyman/VSE* tool, which you use for most of the steps concerning configuring for server and client authentication.
- Get started (for testing and learning purposes) using the IBM-supplied sample keys and certificates contained in the VSE Keyring Library (on the z/VSE host) and client keyring file. These keys and certificates enable your CICS clients to start communicating immediately with the CICS Transaction Server running on the z/VSE host, using *SSL server authentication*.
- With the preceding example:

If you use the TCP/IP for z/VSE implementation you need 3 members in the library specified by KEYFILE:

- KEY1024.CERT
- KEY1024.ROOT
- KEY1024.PRVK.

If you use the OpenSSL implementation of z/VSE you need one member KEY1024.PEM. You can use any suitable tool to generate the PEM file.

Step 4A: Set Up your z/VSE System to use OpenSSL

In order to set up your z/VSE System to use OpenSSL, do this:

- To setup OpenSSL within a CICS partition add // SETPARM BPX\$GSK= 'IJBGSK0S' in the CICS startup job or
- To setup OpenSSL system-wide add // SETPARM SYSTEM BPX\$GSK= 'IJBGSK0S'.

Step 5: Configure for Server Authentication

For most CICS applications, server authentication provides a sufficient level of SSL security, and means that a server certificate is provided by the server (in this case, the z/VSE host) to authenticate the server to CICS clients (Web browsers).

For details of how to configure your z/VSE system for SSL server authentication, refer to the section "Configuring for Server Authentication" in the *z/VSE Administration*, SC34-2692. In this chapter, the steps that are not relevant for CICS Web Support, or that are specific to CICS Web support, are clearly indicated.

Step 6: Configure for Client Authentication

If client authentication is required (in addition to server authentication), a *client certificate* is provided by CICS clients to authenticate the CICS client to the server.

To implement client authentication in CICS Web Support, you must configure *each* CICS client (Web browser) for client authentication. Before beginning, you must ensure that you have set SSL (CLIENTAUTH) in the TCPIPSERVICE profile, as described in "Step 3: Define a TCPIPRESOURCE" on page 130.

Then you must follow the steps described in the topic "Configuring for Client Authentication" in the *z/VSE Administration*, SC34-2692. In this topic, the steps that are not relevant for CICS Web Support, or that are specific to CICS Web support, are clearly indicated.

Step 7: Configure for Client Certificate Mapping

Using the service functions for client authentication, you can introduce access checking on client certificates via z/VSE User IDs that have been assigned to these client certificates.

Since the client certificates belong to CICS clients, using client certificates you can control the access rights from CICS clients to z/VSE host resources.

For details of how to implement access checking on client certificates, refer to the topic "Implementing Client Authentication with VSE User-ID Mapping" in the *z/VSE Administration*, SC34-2692.

Application programming considerations

You can examine existing application programs to see whether they can exploit the EXEC CICS EXTRACT CERTIFICATE command. This command is used with *client authentication*, and allows you to extract information from any client certificate received over an SSL connection. See "EXTRACT CERTIFICATE" on page 68 for details.

A sample application program: DFHOWBCA

DFHOWBCA is a sample program provided by CICS. It demonstrates how you can extract information from an SSL client certificate and construct the response as a CICS document with the EXEC CICS DOCUMENT command. The *CICS Application Programming Guide* contains guidance information about the EXEC CICS DOCUMENT commands.

Currently-Supported SSL Cipher Suites

The SSL Cipher Suites supported by TCP/IP for z/VSE differ from the SSL Cipher Suites supported by the OpenSSL implementation. For a list of supported SSL Cipher Suites see *z/VSE TCP/IP Support*.

Part 4. Using the CICS Transaction Gateway with ECI

Chapter 22. Introduction to the ECI / CICS Transaction Gateway

This section provides an introduction to the CICS Transaction Gateway used with z/VSE, and the External Call Interface (ECI).

It contains these main topics:

- “How the ECI and CICS Transaction Gateway are used” on page 135
- “How the CICS Transaction Gateway accesses CICS” on page 135
- “The External Call Interface (ECI)” on page 136

How the ECI and CICS Transaction Gateway are used

The CICS Transaction Gateway provides a **CICS Java class library** that includes classes that provide an application programming interface (API), and are used to communicate between the Java gateway application and a Java application or applet.

The class *JavaGateway* is used to establish communication with the Gateway process, and uses Java's sockets protocol. The class *ECIRequest* is used to specify the External Call Interface (ECI) calls that are passed to the gateway.

The multithreaded architecture of the CICS Transaction Gateway enables a single Gateway to support multiple concurrently connected users.

The CICS Transaction Server supports the TCP/IP protocol for connections to the CICS Transaction Gateway, which enables the CICS Universal Client to use the External Call Interface (ECI) via TCP/IP. However, the External Presentation Interface (EPI) is *not* supported here.

Figure 9 on page 135 shows how the CICS Transaction Gateway and ECI are used in a 3-tier environment.

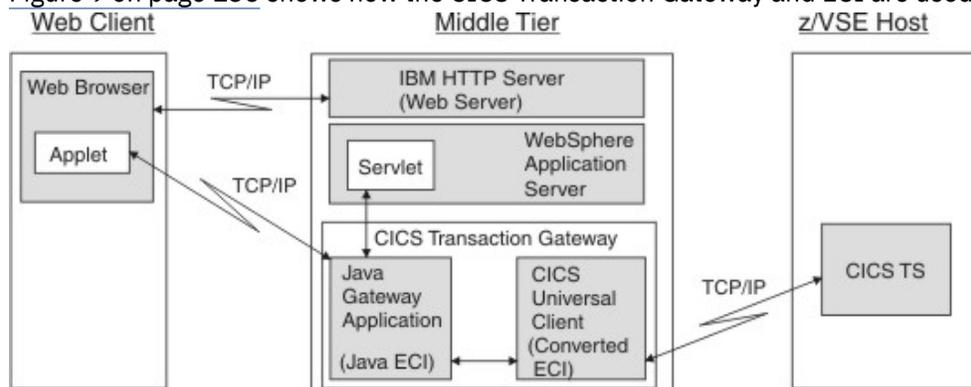


Figure 9. How the CICS Transaction Gateway and ECI are used

How the CICS Transaction Gateway accesses CICS

The flow of control when a Web browser calls CICS transaction processing facilities using the CICS Transaction Gateway (as shown in Figure 9 on page 135) is as follows:

1. The Web browser calls the Web server on the middle-tier, using HTTP (Hypertext Transfer Protocol) to get HTML pages.
2. When the browser, which is interpreting the HTML and presenting it on the screen, finds an applet tag, it calls the Web server on the middle-tier to get the applet and the classes that it needs. It then executes the applet.

3. An applet that is going to communicate with CICS creates a *JavaGateway* object. The creation of this object causes a call to the CICS Transaction Gateway long-running task on the middle-tier .
4. The applet creates an *ECIRequest* object to represent its request for a CICS program, and calls the flow method of the *JavaGateway* object, passing the instance of the *ECIRequest* object.
5. The CICS Transaction Gateway on the middle-tier receives the request, and calls the CICS program on the z/VSE host.
6. When the CICS program ends, the results are returned to the Web browser via the CICS Transaction Gateway on the middle-tier.

The flow of data when a Web browser calls CICS transaction processing facilities using the gateway, is as follows:

1. The Web browser acquires data for the CICS program from the end user.
2. The Web browser constructs a communication area for the CICS program that is to supply transaction processing services.
3. The CICS Transaction Gateway on the middle-tier receives the communication area and passes it to the CICS program on the z/VSE host. The contents of the communication area are translated from ASCII (in the gateway) to EBCDIC (in the CICS Transaction Server).
4. The CICS program on the z/VSE host supplies the transaction processing services, enquiring on and perhaps changing CICS resources. If the program ends normally, changes to recoverable resources are committed. If the program ends abnormally, the changes are backed out.
5. The communication area is translated from EBCDIC to ASCII, and returned to the gateway on the middle-tier, which forwards it to the Web browser.
6. The Web browser presents information to the end user.

The External Call Interface (ECI)

The ECI allows a non-CICS application to call a CICS program in a CICS server. The application can be connected to several servers at the same time, and it can have several program calls outstanding at the same time.

The CICS program cannot perform terminal I/O, but can access and update all other CICS resources. The same CICS program can be called by a non-CICS application using the ECI, or by a CICS program using EXEC CICS LINK. Data is exchanged between the two programs by means of a COMMAREA, in a similar way to CICS. The user can specify the length of the COMMAREA data to optimize performance.

Calls may be made synchronously or asynchronously. Synchronous calls return control when the called program completes, and the information returned is immediately available. Asynchronous calls return control without reference to the completion of the called program, and the application can ask to be notified when the information becomes available.

Calls may also be extended. That is, a single logical unit of work may cover two or more successive calls, though only one call can be active for each logical unit of work at any time. If it uses asynchronous calls, the application can manage many logical units of work concurrently.

The called program can update resources on its own system, it can use distributed program link (DPL) to call CICS programs on other systems, and it can access resources on other CICS systems by function shipping, by distributed transaction processing (DTP), or (in the CICS Transaction Server for z/VSE environment) by the front end programming interface (FEPI).

For more information on the external access interfaces, see [CICS Family: Client/Server Programming](#).

Chapter 23. Configuring for ECI via TCP/IP

This topic explains how to configure the CICS Transaction Server for z/VSE to use the CICS ECI (External Call Interface) via TCP/IP.

It contains these main topics:

- [“Hardware prerequisites” on page 137](#)
- [“Software prerequisites” on page 137](#)
- [“Defining a TCPIP SERVICE Resource for ECI” on page 137](#)
- [“Displaying the ATTACHSEC Attribute” on page 138](#)

Hardware prerequisites

None.

Software prerequisites

These are the software prerequisites for using CICS ECI via TCP/IP:

- VSE/ESA Version 2 Release 6 or later.
- The CICS Universal Client.

Defining a TCPIP SERVICE Resource for ECI

To use the ECI via TCP/IP support, you must install and activate *one or more* TCP/IP services for ECI use.

There are three TCPIP SERVICE resource definition attributes relating to ECI:

1. ATTACHSEC
2. PORTNUMBER
3. TRANSACTION

See [Chapter 28, “CEDA DEFINE TCPIP SERVICE,” on page 161](#) for a description of the above attributes (which are pre-defined in CICS TS systems that contain CICS Web Support).

You activate the TCPIP SERVICE definition either by:

- Specifying STATUS (OPEN) and installing the definition.
- Installing the definition and later using a CEMT SET TCPIP SERVICE OPEN command (described in [Chapter 28, “CEDA DEFINE TCPIP SERVICE,” on page 161](#)).

[Figure 10 on page 138](#) shows an example of the entries you make when defining a TCPIP SERVICE definition for use with ECI:

```

OVERTYPE TO MODIFY
CEDA  DEFine TCpipservice(          )
TCpipservice   : ECI
Group          : TCPIP
Description    : CICS ECI support
Urm           :
Portnumber     : 01435                1-32767
Certificate    :
STatus        : Open                  Open ! Closed
SSL           : No                    Yes ! No ! Clientauth
Attachsec     : Verify                Local ! Verify
TRansaction   : CIEP
Backlog       : 00005                0-32767
TSqprefix     :
Ippaddress    :
S0cketclose   : No                   No ! 0-24000 (HHMSS)

```

Figure 10. Example of a TCPIP SERVICE resource definition for ECI

Related Section:

[“CEMT INQUIRE/SET TCPIP SERVICE” on page 178](#)

Displaying the ATTACHSEC Attribute

If you enter CEMT I TCPIP SERVICE, the ATTACHSEC setting will appear as follows:

```

I TCPIPS
STATUS: RESULTS - OVERTYPE TO MODIFY
  TcpiPs(ECI ) Bac( 00005 ) Con(0000) Por(01435) Clo
  Tra(CIEP) Urm(          ) Ipa(9.20.101.7 ) Attsec(VER) Wai

```

Figure 11. The CEMT INQ TCPIP SERV display.

Part 5. Using CICS documents

This part introduces CICS documents. It tells you what you need to consider when writing applications that use documents as a means of formatting information.

The document handler domain allows you to build up formatted data areas, known as documents. Some examples of how these formatted areas, or documents, can be used, are:

- Sending HTML data to be displayed by a Web browser.
- Creating standard formats for printing (for example, using your own letterhead, address, and so on).

Chapter 24. The DOCUMENT application programming interface

This section explains the function and use of the commands in the DOCUMENT application programming interface:

- EXEC CICS DOCUMENT CREATE
- EXEC CICS DOCUMENT INSERT
- EXEC CICS DOCUMENT RETRIEVE
- EXEC CICS DOCUMENT SET

Creating a document

To create an empty document, use the EXEC CICS DOCUMENT CREATE command. This has a mandatory DOCTOKEN parameter requiring a 16-byte data-area. The document handler domain uses the DOCTOKEN operand to return a token, which is used to identify the document on subsequent calls. The following example creates an empty document, and returns the token in the variable MYDOC:

```
EXEC CICS DOCUMENT CREATE
      DOCTOKEN(MYDOC)
```

To create a document with data, use the EXEC CICS DOCUMENT CREATE command in any of the following ways:

- Specify the BINARY parameter
- Specify the TEXT parameter
- Insert one document into another document
- Use document templates

The BINARY parameter

Use this parameter to add to the document the contents of a data-area that must not undergo conversion to a client code page when the data is sent.

```
EXEC CICS DOCUMENT CREATE
      DOCTOKEN(MYDOC1)
      BINARY(DATA-AREA)
```

The TEXT parameter

Use this parameter to add the specified contents to the document. For example, if you define a character string variable called DOCTEXT and initialise it to *This is an example of text to be added to a document*, you can use the following command to create a document consisting of this text string:

```
EXEC CICS DOCUMENT CREATE
      DOCTOKEN(MYDOC2)
      TEXT(DOCTEXT)
      LENGTH(53)
```

Inserting one document into another

To insert an existing document into a new document, you can use the EXEC CICS DOCUMENT CREATE command with the FROMDOC option. The following example does this:

```
EXEC CICS DOCUMENT CREATE  
DOCTOKEN(MYDOC3)  
FROMDOC(MYDOC2)
```

where MYDOC2 and MYDOC3 are 16-character variables. MYDOC2 must contain the token returned by a previous EXEC CICS DOCUMENT CREATE command.

This results in two identical documents, each containing the text *This is an example of text to be added to a document.*

Using document templates

Portions of the data which make up a document can be created off-line and then inserted directly into the document. These are known as **templates**; they are CICS resources, defined using RDO.

Templates can contain a mixture of static data with symbols embedded in the data, which are substituted at run time when the template is inserted into the document. An example of this is when a programmer creates HTML web pages using an HTML editor. The output from the HTML editor can then be made accessible to CICS Web Interface applications using templates.

Chapter 25. Programming with documents

This section covers the following topics:

- Symbols and symbol lists
- Embedded DOCTEMPLATE commands
- Using DOCTEMPLATES in your application
- The lifespan of a document
- Retrieving the document without control information
- Using multiple calls to construct a document
- Bookmarks and inserting data
- Replacing data in the document
- Codepages and codepage conversion

Setting symbol values

The application program needs to define values for the symbols that will be substituted when the template is used. These values can be defined on the EXEC CICS DOCUMENT CREATE or the EXEC CICS DOCUMENT SET commands. The symbols that are set are associated with a particular document and cannot be used in a different document.

The DOCUMENT CREATE and DOCUMENT SET commands both take a SYMBOLLIST operand which allows several symbols to be defined in a single command. The SYMBOLLIST operand is a character string consisting of one or more definitions with single byte separators. By default, the separator is an ampersand, but you can override this by using the DELIMITER option of the DOCUMENT SET or DOCUMENT CREATE commands. A definition consists of a name, an equals sign, and a value. Here is an example:

```
mytitle=New Authors&auth1=Halliwell Sutcliffe&auth2=Stanley  
Weyman
```

This example defines three symbols. The first symbol called mytitle will have the value 'New Authors'. The second symbol called auth1 will have the value 'Halliwell Sutcliffe' and the last symbol called auth2 will contain the value 'Stanley Weyman'.

The following rules apply when setting symbols using a SYMBOLLIST. The name must contain only uppercase and lowercase letters, numbers and the special characters dollar ('\$'), underscore ('_'), hyphen ('-'), pound ('#'), period ('.') and at sign ('@'). The name is case-sensitive, so uppercase letters are regarded as different from lowercase letters.

The values in the symbol list can contain any characters except the symbol separator (which defaults to an ampersand, but can be overridden by use of the DELIMITER option). The following restrictions on the use of the percent sign ("%") and the plus sign ("+") apply unless the UNESCAPED option of DOCUMENT CREATE or DOCUMENT SET has been specified.. A percent sign must be followed by two characters that are hexadecimal digits (that is, 0–9, a-f, and A-F). When the value is put into the symbol table, a plus sign is interpreted as a space, a percent sign and the two hexadecimal digits following it are interpreted as the EBCDIC equivalent of the single ASCII character denoted by the two digits, and the remaining characters are left as they are. If you want a plus sign in the value in the symbol table, you must put %2B in the value in the symbol list. If you want a percent sign in the value in the symbol table, you must put in the value %25 in the symbol list. If you want an ampersand in the value in the symbol table, you must put %26 in the value in the symbol list. If you want a space in the value in the symbol table, the value in your symbol list may contain a space, a plus sign, or a %20.

The DOCUMENT SET command allows you to set individual symbol values with the SYMBOL and VALUE options. Ampersands have no special significance when used in the VALUE option. The restrictions on

the use of the plus sign and percent sign for SYMBOLLISTS also apply to the VALUE option unless the UNESCAPED option of the DOCUMENT SET has been specified.

The following example shows you how you can pass symbol values to the document handler containing embedded plus signs, percent signs, and ampersands, none of which are to undergo unescape processing:

```
EXEC CICS DOCUMENT CREATE
      DOCTOKEN(ATOKEN)
      DELIMITER('!')
      SYMBOLLIST('COMPANY=BLOGGS & SON!ORDER=NUTS+BOLTS')
      LISTLENGTH(37)
      UNESCAPED
```

Here the symbol COMPANY has a value of 'BLOGGS & SON', and the symbol ORDER has a value of 'NUTS+BOLTS'. The delimiter used in this example is '!', but it is best to use a non-printable character that does not appear in the symbol value. The use of the UNESCAPED option ensures that the plus sign in 'NUTS+BOLTS' does not get converted to a space.

Embedded template commands

The Document Handler recognises four commands which can be embedded in the template. Three of the commands follow the syntax rules for Server Side Include commands. A Server Side Include command starts with the characters left angle bracket, exclamation mark, hyphen, hyphen, pound followed by the command and it is terminated with the characters hyphen, hyphen, right angle bracket. For example:

```
(e.g. <!--#command -->).
```

The three commands that are supported are #set, #echo and #include.

The #set command is used to set the values of symbols and is useful for setting up default values for symbols. The #set command in the template will be ignored if the symbol has already been given a value using the EXEC CICS DOCUMENT SET command. If a previous #set command has been used to assign a value to the symbol, the value will be overridden. A symbol which has been assigned a value using the EXEC CICS DOCUMENT SET command can only be changed by issuing another EXEC CICS DOCUMENT SET command.

The #echo command identifies a symbol that must be substituted when the template is inserted into the document. The string containing the #echo command will be completely replaced by the value associated with the symbol. If no symbol has been defined with that name, the #echo command will remain in the output data. An alternative method to using the #echo command is to specify the symbol name, preceding it with an ampersand and terminating it with a semicolon. If we set a symbol called ASYM and give it a value of 'sample', the following two templates will give the same result after substitution.

```
Template 1:
This is an example template.
<!--#set var=ASYM value='sample'-->
This is a <!--#echo var=ASYM--> symbol.
```

```
Template 2:
This is an example template.
<!--#set var=ASYM value='sample'-->
This is a &ASYM; symbol.
```

```
Result of substitution:
This is an example template.
This is a sample symbol.
```

The #include command allows a template to be embedded within another template. Up to 32 levels of embedding are allowed.

Using templates in your application

If you have created a template and defined it to CICS, the following example shows how you can use the template to create the contents of a document. The following template is created and defined to CICS with the name ASampleTemplate.

```
<!--#set var=ASYM value='DFLTUSER'-->
This is a sample document which has been created by user
<!--#echo var=ASYM-->.
```

In the application program, you can define a 48-byte variable called `TEMPLATENAME` and initialize it to a value of 'ASampleTemplate'. Once again you must define a 16-byte field for the document token (in this example, `ATOKEN`). You can then issue the command to create the document.

```
EXEC CICS DOCUMENT CREATE
          DOCTOKEN(ATOKEN)
          TEMPLATE(TEMPLATENAME)
```

This will result in a document being created with the content “This is a sample document which has been created by user DFLTUSER.”.

To change the symbol to another value, you can issue the `EXEC CICS DOCUMENT CREATE` command with the `SYMBOLLIST` option:

```
EXEC CICS DOCUMENT CREATE
          DOCTOKEN(ATOKEN)
          TEMPLATE(TEMPLATENAME)
          SYMBOLLIST('ASYM=Joe Soap')
          LISTLENGTH(13)
```

This will result in a document being created with the content “This is a sample document which has been created by user Joe Soap.”.

The lifespan of a document

Documents created by an application exist only for the length of the CICS task in which they are created. This means that when the last program in the CICS task returns control to CICS, all documents created during the task’s lifetime are deleted. It is the application’s responsibility to save a document before terminating if the document is going to be used in another task. You can obtain a copy of the document by using the `EXEC CICS DOCUMENT RETRIEVE` command. The application can then save this copy to a location of its choice, such as a temporary storage queue. The copy can then be used to recreate the document.

The following sequence of commands show how a document can be created, retrieved and stored on a temporary storage queue, assuming that the following variables have been defined and initialized in the application program:

- A 16-byte field `ATOKEN` to hold the document token
- A 20-byte buffer `DOCBUF` to hold the retrieved document
- A fullword binary field called `FWORDLEN` to hold the length of the data retrieved
- A halfword binary field called `HWORDLEN` to hold the length for the temporary storage `WRITE` command.

```
EXEC CICS DOCUMENT CREATE
          DOCTOKEN(ATOKEN)
          TEXT('A sample document.')
          LENGTH(18)
```

```
EXEC CICS DOCUMENT RETRIEVE
          DOCTOKEN(ATOKEN)
          INTO(DOCBUF)
          LENGTH(FWORDLEN)
          MAXLENGTH(20)
```

```
EXEC CICS WRITEQ TS
        QUEUE('AQUEUE')
        FROM(DOCBUF)
        LENGTH(HWORDLEN)
```

You can now use the following sequence of commands to recreate the document in the same or another application.

```
EXEC CICS READQ TS
        QUEUE('AQUEUE')
        INTO(DOCBUF)
        LENGTH(HWORDLEN)
```

```
EXEC CICS DOCUMENT CREATE
        DOCTOKEN(ATOKEN)
        FROM(DOCBUF)
        LENGTH(FWORDLEN)
```

When the document is retrieved, the data that is delivered to the application buffer is stored in a form which contains control information necessary to reconstruct an exact replica of the document. The document that is created from the retrieved copy is therefore identical to the original document. To help the application calculate the size of the buffer needed to hold a retrieved document, each document command which alters the size of the document has a DOCSIZE option. This is a fullword value which gives the maximum size that the buffer must be to contain the document when it is retrieved. This size is calculated to include all the control information and data. The size should not be taken as an accurate size of the document as the actual length delivered to the application can often be slightly smaller than this size. The length delivered will however never exceed the length in the DOCSIZE option.

The above example introduced the use of the FROM option on the DOCUMENT CREATE command. The data passed on the FROM option was the buffer returned to the application when the DOCUMENT RETRIEVE command was issued. It is possible for the application to supply data on the FROM option that did not originate from the DOCUMENT RETRIEVE command. When this happens, the document handler treats the data as a template and parses the data for template commands and symbols.

Retrieving the document without control information

The document data containing control information is only useful to an application that wishes to recreate a copy of the original document. It is possible to issue a DOCUMENT RETRIEVE command and ask for the control information to be omitted. The following command sequence uses the DATAONLY option on the DOCUMENT RETRIEVE command to instruct the Document Handler to return only the data. This example assumes that the following variables have been defined and initialised in the application program:

- A 16-byte field ATOKEN to hold the document token
- A 20-byte buffer DOCBUF to hold the retrieved document
- A fullword binary field called FWORDLEN to hold the length of the data retrieved.

```
EXEC CICS DOCUMENT CREATE
        DOCTOKEN(ATOKEN)
        TEXT('A sample document.')
        LENGTH(18)
```

```
EXEC CICS DOCUMENT RETRIEVE
        DOCTOKEN(ATOKEN)
        INTO(DOCBUF)
        LENGTH(FWORDLEN)
        MAXLENGTH(20)
        DATALONLY
```

When the commands have executed, the buffer DOCBUF will contain the string “A sample document.”.

Using multiple calls to construct a document

Once a document has been created, the contents can be extended by issuing one or more EXEC CICS DOCUMENT INSERT commands. The options on the EXEC CICS DOCUMENT INSERT command work in the same way as the equivalent options on the EXEC CICS DOCUMENT CREATE command. The following sequence of commands shows an empty document being created followed by two INSERT commands:

```
EXEC CICS DOCUMENT CREATE
          DOCTOKEN(ATOKEN)
```

```
EXEC CICS DOCUMENT INSERT
          DOCTOKEN(ATOKEN)
          TEXT('Sample line 1. ')
          LENGTH(15)
```

```
EXEC CICS DOCUMENT INSERT
          DOCTOKEN(ATOKEN)
          TEXT('Sample line 2. ')
          LENGTH(15)
```

The document resulting from the above commands will contain:

```
Sample line 1. Sample line 2.
```

You can use the DOCUMENT RETRIEVE and DOCUMENT INSERT commands to insert a whole document into an existing document. The following variables must first be defined and initialized in the application program:

- A 16-byte field RTOKEN which contains the document token of the document to be retrieved
- A buffer DOCBUF of sufficient length to hold the retrieved document
- A fullword binary field called RETRIEVLEN to hold the length of the data retrieved
- A fullword binary field called MAXLEN to hold the maximum amount of data the buffer can receive, i.e. the length of DOCBUF
- A 16-byte field ITOKEN which contains the document token of the document that is being inserted into

The following sequence of commands shows a document indicated by RTOKEN being inserted into another document indicated by ITOKEN:

```
EXEC CICS DOCUMENT RETRIEVE
          DOCTOKEN(RTOKEN)
          INTO(DOCBUF)
          LENGTH(RETRIEVLEN)
          MAXLENGTH(MAXLEN)
```

```
EXEC CICS DOCUMENT INSERT
          DOCTOKEN(ITOKEN)
          FROM(DOCBUF)
          LENGTH(RETRIEVLEN)
```

The retrieved document is inserted at the end of the document specified in the DOCUMENT INSERT command, and all the control information of the retrieved document will be present in the second document. The LENGTH parameter of the DOCUMENT INSERT command must be equal to the value returned from the DOCUMENT RETRIEVE command into the field RETRIEVLEN.

The DOCUMENT INSERT command allows an operand called SYMBOL to be used to add blocks of data to the document. SYMBOL must contain the name of a valid symbol whose value has been set. The Document Handler inserts the value that is associated with the symbol into the document.

Bookmarks and inserting data

The sequence in which an application inserts data into a document might not reflect the desired sequence that the data should appear in the document. Bookmarks allow the application to insert blocks of data in any order yet still control the sequence of the data in the document. A bookmark is a label that the

application inserts between blocks of data. Note: a bookmark cannot be inserted in the middle of a block of data.

The following example creates a document with two blocks of text and a bookmark:

```
EXEC CICS DOCUMENT CREATE
      DOCTOKEN(ATOKEN)
      TEXT('Pre-bookmark text. ')
      LENGTH(19)
```

```
EXEC CICS DOCUMENT INSERT
      DOCTOKEN(ATOKEN)
      BOOKMARK('ABookmark      ')
```

```
EXEC CICS DOCUMENT INSERT
      DOCTOKEN(ATOKEN)
      TEXT('Post-bookmark text. ')
      LENGTH(20)
```

The document will now contain:

```
Pre-bookmark text. <ABookmark>Post-bookmark text.
```

Note that the text <ABookmark> does not appear in the document content but serves merely as a pointer to that position in the document. To add data to this document, you can insert text at the bookmark as follows:

```
EXEC CICS DOCUMENT INSERT
      DOCTOKEN(ATOKEN)
      TEXT('Inserted at a bookmark. ')
      LENGTH(25)
      AT('ABookmark      ')
```

Logically, the data of the document will contain the following (Note that in this instance, only the data is being shown and not the position of the bookmark).

```
Pre-bookmark text. Inserted at a bookmark. Post-bookmark text.
```

If the AT option is omitted, the data is always appended to the end of the document. A special bookmark of 'TOP' can be used to insert data at the top of the document, making it unnecessary to define a bookmark which will mark the top of the document.

Replacing data in the document

The following example shows how data between two bookmarks can be replaced:

```
EXEC CICS DOCUMENT CREATE
      DOCTOKEN(ATOKEN)
```

```
EXEC CICS DOCUMENT INSERT
      DOCTOKEN(ATOKEN)
      TEXT('Initial sample text. ')
      LENGTH(21)
```

```
EXEC CICS DOCUMENT INSERT
      DOCTOKEN(ATOKEN)
      BOOKMARK('BMark1      ')
```

```
EXEC CICS DOCUMENT INSERT
      DOCTOKEN(ATOKEN)
      TEXT('Text to be replaced. ')
      LENGTH(21)
```

```
EXEC CICS DOCUMENT INSERT
      DOCTOKEN(ATOKEN)
      BOOKMARK('BMark2      ')
```

```
EXEC CICS DOCUMENT INSERT
      DOCTOKEN(ATOKEN)
      TEXT('Final sample text. ')
      LENGTH(19)
```

At this point the logical structure of the document will be as follows:

```
Initial sample text. <BMark1>Text to be replaced. <BMark2>Final
sample text.
```

You can now issue the command to replace the text between the two bookmarks, BMark1 and BMark2:

```
EXEC CICS DOCUMENT INSERT
      DOCTOKEN(ATOKEN)
      TEXT('Replacement Text. ')
      LENGTH(18)
      AT('BMark1      ')
      TO('BMark2      ')
```

The document now has the following logical structure:

```
Initial sample text. <BMark1>Replacement Text. <BMark2>Final
sample text.
```

Codepages and codepage conversion

The documents that an application creates may be transmitted to systems running on other platforms, especially when applications running under the CICS Web interface use the Document Handler to generate Web pages. To assist the application with the problem of converting data from the codepages used on the host to the codepages used on the target system, the Document Handler allows the application to specify the codepages being used on each system. When the EXEC CICS DOCUMENT CREATE and EXEC CICS DOCUMENT INSERT commands are used, the TEXT, FROM, TEMPLATE and SYMBOL options can have a HOSTCODEPAGE option coded to indicate the codepage for that block of data, since the data being added with these options is seen as being textual data. Each block can be specified in a different codepage. When the EXEC CICS DOCUMENT RETRIEVE command is issued, the CLNTCODEPAGE option tells the Document Handler to convert all the individual blocks from their respective host codepages into a single client codepage.

Part 6. CICS Resource Definitions Online changes

Chapter 26. CEDA DEFINE DOCTEMPLATE

Use this resource definition to define document templates to CICS. Document templates allow you to perform variable substitution on documents in a manner similar to that done by BMS for 3270 screens.

The template can reside in any one of the following places:

- z/VSE sublibrary
- CICS temporary storage
- CICS transient data
- CICS load module
- exit program

Defining a DOCTEMPLATE

```

DOctemplate ==>
Group      ==>
DEscription ==>
FULL TEMPLATE NAME
TEmplatename ==>
ASSOCIATED CICS RESOURCE
File       ==>
TSqueue    ==>
TDqueue    ==>
Program    ==>
Exitpgm    ==>
TEMPLATE SUBLIBRARY
Library     ==>
Membername ==>
TEMPLATE PROPERTIES
Appendcrlf ==> Yes           Yes | No
Type       ==>             Binary | Ebcdic

```

Figure 12. The DEFINE panel for DOCTEMPLATE

Options

APPENDCRLF

specifies whether CICS is to delete trailing blanks from and append carriage-return line-feed to each logical record of the template.

LIBRARY

specifies the library name of the z/VSE library containing the template. The library applies only to a template of type z/VSE library. If a membername attribute is supplied without a value for library, the SOURCE LIBDEF chain is searched for the template. If a value for library is specified, the sublibrary DFHDOC of that library is used to locate the template.

DESCRIPTION(text)

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

DOCTEMPLATE

specifies the name of this DOCTEMPLATE definition. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 \$ @ # . / - _ % & ¢ ? ! : | " = ~ , ; < and >.

EXITPGM

specifies the exit program to be invoked when a request is made for this template. CICS passes a commarea to the exit program which is mapped by the following copybooks:

DOCTEMPLATE

- DFHDHTXD (Assembler)
- DFHDHTXH (C)
- DFHDHTXL (PL/I)
- DFHDHTXO (COBOL)

FILE(filename)

specifies, for a template of type FILE, the up to 7-character name of the CICS file definition for the data set containing the template.

GROUP(groupname)

specifies the group name, which can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and \$. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see *CICS Resource Definition Guide*.

MEMBERNAME

specifies the name of the z/VSE library member containing the template. It applies only to a template of type z/VSE library. The z/VSE library member type of the template must be HTML.

PROGRAM

specifies the program in which the template data is stored. CICS loads the program and takes all data after the entrypoint to be the template.

TDQUEUE

specifies the name of the TD queue on which the template is stored.

TEMPLATENAME

specifies the extended Template-name by which the doctemplate is to be known outside the resource definition function.

TSQUEUE

specifies the name of the TS queue on which the template is stored.

TYPE(BINARY|EBCDIC)

specifies the format of the contents of the template.

BINARY

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

EBCDIC

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

Note: The fields EXITPGM, FILE, MEMBER, PROGRAM, TDQUEUE, and TSQUEUE define alternative sources of the template data. Only one of them may be specified on each DOCTEMPLATE definition.

Chapter 27. CEDA DEFINE PROFILE

You specify options that control the interactions between transactions and terminals or logical units, as a PROFILE. The PROFILE is a means of standardizing the use of such options as screen size and printer compatibility, and the use of such functions as message journaling and the node error program.

MODENAME

A profile is associated with the communication between a transaction and an LUTYPE6.1 or APPC session to another system. For APPC sessions, you refer on the PROFILE definition to the MODENAME that is also named on the SESSIONS definitions. This MODENAME is the name of the mode set to which the sessions belong. Refer to the [CICS Resource Definition Guide, SC33-1653](#).

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 ISC" in the [CICS Resource Definition Guide, SC33-1653](#). The [CICS Intercommunication Guide, SC33-1665](#) gives further information about the CICS-supplied PROFILES, and tells you about defining your own profiles.

Defining a PROFILE

```

PROFile      ==>
Group        ==>
Description  ==>
Scrnsize    ==> Default      Default | Alternate
Uctran       ==> No          No | Yes
M0dename     ==>
Facilitylike ==>
PRIntercomp ==> No          No | Yes
JOURNALLING
Journal      ==> No          No | 1-99
MSGJrn1     ==> No          No | INPut | Output | INOut
PROTECTION
MSGInteg     ==> No          No | Yes
Onewte       ==> No          No | Yes
PROtect      : No          No | Yes
Chaincontrol ==> No          No | Yes
PROTOCOLS
DVsuprt      ==> All         All | Nonvtam | Vtam
Inbfmh       ==> No          No | All | Dip | Eods
RAq          ==> No          No | Yes
Logrec       ==> No          No | Yes
RECOVERY
Nepclass     ==> 000         0-255
RTimout      ==> No          No | 1-7000

```

Figure 13. The DEFINE panel for PROFILE

Options

CHAINCONTROL({NO|YES})

specifies whether the application program can control the outbound chaining of request units. If you specify CHAINCONTROL(YES), ONEWTE(YES) means one chain and not one terminal control output request.

DESCRIPTION(text)

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

DVSUPRT({ALL|NONVTAM|VTAM})

specifies the devices (terminals or logical units) that are to be supported. The access method used by a particular terminal or logical unit is specified in its associated TCTTE.

ALL

The profile can be used with any terminal or logical unit.

NONVTAM

The profile can be used only with non-VTAM terminals.

VTAM®

The profile can be used only with logical units.

FACILITYLIKE(name)

This is an optional parameter that specifies the name of an existing (four character) terminal resource definition to be used as a template for the bridge facility. It can be overridden by specifying FACILITYLIKE in the bridge exit.

There is no default value for this parameter.

If you are running in a CICS system started with the VTAM=NO system initialization (SIT) parameter, the resource definition specified by FACILITYLIKE must be defined as a remote terminal.

GROUP(groupname)

Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see [CICS Resource Definition Guide, SC33-1653](#).

The GROUP name can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and \$. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

INBFMH({NO|ALL|DIP|EODS}) (SNA LUs only)

specifies, for profiles used with logical units, whether a function management header (FMH) received from a logical unit is to be passed to the application program.

ALL

All FMHs (except APPC FMHs and LU6.1 ATTACH and SYNCPOINT FMHs that are processed by CICS) are passed to the application program. This value is required for function shipping transactions such as CSMI, transactions which use distributed transaction processing, and for distributed program link requests.

DIP

The batch data interchange program (DFHDIP) is to process inbound FMHs. BMS issues a batch data interchange receive request if a BMS receive request has been issued, and a batch data interchange receive request is issued instead of a terminal control receive request.

EODS

An FMH is passed to the application program only if it indicates end of data set (EODS).

NO

The FMHs are discarded.

JOURNAL({NO|number})

specifies that you want automatic journaling of messages to take place, by giving the identifier of the journal.

NO

No automatic journaling of messages is to take place.

number

The journal identification to be used for automatic journaling. This can be any number in the range 01 through 99. This number is appended to the letters DFHJ to give a journal identification of the form DFHJ*nn*.

LOGREC({NO|YES})

specifies whether the design of the application requires that each EXEC CICS RECEIVE request is to be satisfied by a logical record. This option allows existing 2770-and 2780-based application programs to be attached to a batch logical unit (for example, 3790 or 8100) without modification to the program.

MODENAME(*name*)

specifies the name that identifies a group of sessions for use on an APPC connection. The name can be up to eight characters in length, and must be the name of a VTAM LOGMODE entry defined to VTAM. It must not be the reserved name SNASVCMG. If you omit the modename, it defaults to blanks. See the [CICS Intercommunication Guide](#) for more information about VTAM modenames.

If a transaction that specifies this profile has been started using an EXEC CICS START command, the MODENAME is used for allocation of the principal facility. If a transaction performs an EXEC CICS ALLOCATE command specifying this profile, the MODENAME is used for allocation of the alternate facility.

If you do not specify a MODENAME, CICS selects a session from any one of the mode sets that have been defined.

The CICS-supplied profile DFHCICSA is used, if PROFILE is not specified on an EXEC CICS ALLOCATE command. For function shipping, the profile DFHCICSF is always used. MODENAME is not specified on the definition for either of these profiles, but you can add a MODENAME if you make your own copy. You must then ensure that the mode sets using your MODENAME have been defined in the TERMINAL or SESSIONS definition for all the systems with which communication takes place using APPC.

If a MODENAME is specified and you wish to remove it, delete completely the value previously specified by pressing the ERASE EOF key.

MSGINTEG({NO|YES}) (SNA LUs only)

specifies whether a definite response is to be requested with an output request to a logical unit. You cannot specify YES for a pipeline transaction.

MSGJRN({NO|INPUT|OUTPUT|INOUT})

specifies which messages are to be automatically journaled. If you specify a value other than NO, you must also supply a value for the JOURNAL attribute.

NO

No message journaling is required.

INPUT

Journaling is required for input messages.

OUTPUT

Journaling is to be performed for output messages.

INOUT

Journaling is to be performed for input and output messages.

NEPCLASS({@|*value*}) (VTAM only)

specifies the node error program transaction class. This value overrides the value specified on the TYPETERM and SESSION definitions.

0

This results in a link to the default node error program module for VTAM devices, or is the default value for non-VTAM devices.

value

The transaction class for the (nondefault) node error program module. The value can be in the range 1 through 255. For programming information on the node error program, see the [CICS Customization Guide, SC33-1652](#).

ONEWTE({NO|YES})

specifies whether the transaction is permitted only one write operation or EXEC CICS SEND during its execution. YES has the effect of forcing the LAST option on the first write of the transaction. Any additional write requests are treated as errors, and the task is made ready for abnormal termination.

You must specify YES for a PIPELINE transaction.

PRINTERCOMP({NO|YES})

specifies the level of compatibility required for the generation of data streams to support the printer compatibility option for the BMS SEND TEXT command.

NO

Each line of output starts with a blank character, so that the format is equivalent to that on a 3270 display where an attribute byte precedes each line.

YES

No blank character is inserted, so that forms-feed characters included as the first character of your data are honored and the full width of the printer is available for your data.

If you use the BMS forms feed option, specify YES.

PROFILE(name)

specifies the name of this PROFILE definition. The name can be up to eight characters in length. The acceptable characters are: A-Z 0-9 \$ @ # . / - _ % & ¢ ? ! : | " = ~ , ; < and >. Do not use profile names beginning with DFH, because these characters are reserved for use by CICS.

Note: If you use a comma (,) in a name, you will be unable to use those commands such as

```
CEMT INQUIRE PROFILE(value1,value2)
```

where the comma serves as a list delimiter. See the [CICS Supplied Transactions, SC33-1655](#) publication for information about using lists of resource identifiers.

A profile specifies the options that control the interaction between CICS and a terminal or logical unit. A profile name is specified on the transaction definition to indicate the set of options that control the communication between the transaction and its principal terminal. You can also specify a profile name on an EXEC CICS ALLOCATE command to indicate the options that control communication between the transaction and the allocated session.

CICS supplies a number of profile definitions that are suitable for most purposes. For guidance on the names of the definitions, see the [CICS Operations and Utilities Guide, SC33-1654](#). Further guidance is also given in the [CICS Intercommunication Guide, SC33-1665](#).

PROTECT({NO|YES}) (SNA LUs only)

This attribute is obsolete from CICS Transaction Server for z/VSE, but is retained for compatibility with earlier releases. If you already use PROTECT, you can still access it by using compatibility mode (see [CICS Resource Definition Guide, SC33-1653](#) for information). See [CICS Resource Definition Guide, SC33-1653](#) for a description of PROTECT.

RAQ({NO|YES}) (SNA terminals only)

specifies whether the 'read ahead queuing' option is required.

NO

The transaction obeys SNA protocols and only SEND and RECEIVE when in the correct mode. If it does not follow the protocol, it may be abended with code ATCV.

YES

The transaction may not obey SNA protocols, and CICS queues incoming data on temporary storage until the data is specifically requested by the transaction. RAQ(YES) is provided only for compatibility with transactions that support both bi-synchronous devices and logical units, and its use is not recommended.

RTIMOUT({NO|value})

specifies the time-out value for the read time-out feature. The task that is timed out receives an AKCT or AZCT abend. (Note that if a value is specified and you wish to let it default to NO, you must completely delete the value previously specified.)

RTIMOUT has no effect for MRO or basic (unmapped) APPC connections.

NO

The read time-out feature is not required.

value

This is an interval (MMSS for minutes and seconds) after which the task is terminated if no input has been received from the terminal. The maximum value that can be specified is 70 minutes. The value specified in this option is rounded up to units of 16.78 seconds. Thus, the minimum value (after rounding-up) is 16.78 seconds.

SCRNSIZE({DEFAULT|ALTERNATE})

specifies whether the DEFAULT or ALTERNATE buffer size for a 3270 display or printer is to be used. For further information on the choice of screen sizes and buffer sizes, refer to the ALTSCREEN and DEFSCREEN attributes on the TYPETERM definition.

The SCRNSIZE value is ignored if the TYPETERM definition has ALTSCREEN(0,0) and DEFSCREEN(0,0). That is, the screen size is assumed from the related TERMMODEL attribute in the TYPETERM definition; the page size is taken from PAGESIZE, and the ALTPAGE value is ignored. The 3270 erase write (EW) command is inserted for output requests with the ERASE option.

ALTERNATE

If the TYPETERM definition has nonzero ALTSCREEN, the alternate screen size mode is applied, using the erase write alternate (EWA) command. That is, whenever a terminal output request with the ERASE option is issued, the 3270 EWA command is inserted in the data stream. The ALTSCREEN value is assumed as the screen size, and BMS uses the value in ALTPAGE as the page size.

SCRNSIZE(ALTERNATE) may be used for all CICS service transactions (for example, CSMT).

DEFAULT

If the TYPETERM definition has nonzero ALTSCREEN or nonzero DEFSCREEN, the default screen size mode is applied, using the erase write (EW) command. That is, whenever the terminal issues a terminal output request with the ERASE option, the 3270 EW command is inserted in the data stream. The screen size specified in the DEFSCREEN attribute is assumed, and BMS uses the value specified in the PAGESIZE attribute as the page size.

Note: Both DEFAULT and ALTERNATE can be overridden by the DEFAULT and ALTERNATE options on the SEND MAP, SEND TEXT, and SEND CONTROL commands. See the [CICS Application Programming Reference](#), SC33-1658 publication for programming information about these commands.

UCTRAN({NO|YES}) (VTAM only)

specifies whether terminal input is to be translated to uppercase before passing to programs for the transaction using this profile.

You can also request translation to uppercase at the terminal level on the associated TYPETERM definition (see [CICS Resource Definition Guide](#), SC33-1653) 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.

PROFILE

- A PROFILE UCTRAN(YES) definition overrides a TYPETERM UCTRAN(NO) definition.
- Specifying TYPETERM UCTRAN(TRANID) causes the tranid to be translated to uppercase so that CICS can locate the transaction definition. All other input received by the application is translated according to what is specified for PROFILE UCTRAN.
- UCTRAN(YES) on a profile definition does not cause translation of the input data until an EXEC CICS RECEIVE or CONVERSE is executed. This means that if the transaction is routed through a dynamic routing program, for example DFHDYP, the copy of the input data passed to the routing program is unaffected by the UCTRAN option of the PROFILE definition.

Note: In a transaction routing environment where your VTAM terminals have a remote definition on the AOR, and the AOR has a different UCTRAN value from the TOR, the TOR value of UCTRANST (as specified in an EXEC CICS SET TERMINAL command) overrides that on the AOR.

Table 11 on page 160 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.

<i>Table 11. The effect of UCTRAN attributes on tranid and data translation</i>			
Profile (PROFILE)	Terminal (TYPETERM)		
	UCTRAN (YES)	UCTRAN (NO)	UCTRAN (TRANID)
UCTRAN (YES)	Tranid: Yes Data: yes	Tranid: No Data: Yes	Tranid: Yes Data: Yes
UCTRAN (NO)	Tranid: Yes Data: Yes	Tranid: No Data: No	Tranid: Yes Data: No

Chapter 28. CEDA DEFINE TCPIP SERVICE

Use the TCPIP SERVICE resource to define which TCP/IP services are to use CICS® internal sockets support. The internal CICS services that can be defined are ECI over TCP/IP, CICS Web support (HTTP), or a user-defined protocol.

The TCPIP SERVICE 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.

TCPIP SERVICE definitions are for use only with the CICS-provided TCP/IP services, and have nothing to do with the IP CICS Sockets interface of the TCP/IP stack.

Defining a TCPIP SERVICE

```

OVERTYPE TO MODIFY                                CICS RELEASE = 0430
CEDA DEFINE TCPIP SERVICE(                          )
  TCPIP SERVICE ==>
  GROUP ==>
  DESCRIPTION ==>
  URM ==>
  PORTNUMBER ==> 00000                               1-65535
  CERTIFICATE ==>
  STATUS ==> Open                                    Open | Closed
  SSL ==> No                                         Yes | No | Clientauth
  ATTACHSEC ==>                                     Local | Verify
  TRANSACTION ==>
  BACKLOG ==> 00001                                 0-32767
  TSQPREFIX ==>
  IPADDRESS ==>
  SOCKETCLOSE ==> No                               No | 0-24000
  PROTOCOL ==>                                     Http | Eci | User
  MAXDATALEN ==>                                    3-524288

```

Figure 14. The DEFINE panel for the TCPIP SERVICE resource definition

Examples of TCPIP SERVICE resource definitions are given in:

[Figure 8 on page 131](#)

[Figure 10 on page 138](#)

Options

ATTACHSEC

specifies the level of attach-time user security required for this connection. This option is valid only for PROTOCOL(ECI).

Possible values are:

VERIFY

The CICS client must provide a userid and password to the z/VSE host. These are checked before the CICS client is allowed to communicate with the z/VSE host.

LOCAL

The CICS client is not required to provide a userid and password to the z/VSE host.

You can define some CICS clients to have LOCAL security, and others to have VERIFY by using two TCPIP SERVICE definitions that:

- Have different security requirements.

- Listen on different ports.

You can then set up each CICS client (using the client's CICSCLI.INI file) to communicate with the port supporting a specific security type.

BACKLOG(*number*)

specifies the number of TCP/IP connections for this service which are queued in TCP/IP before TCP/IP starts to reject incoming client requests.

CERTIFICATE(*certificate-label*)

The TCP/IP service uses one of the certificates in the VSE Keyring Library as its server certificate. Use the CERTIFICATE option to specify a particular certificate (where *certificate_label* is the name that you assigned to the certificate). If you do not specify CERTIFICATE, CICS uses the default server certificate in the VSE Keyring Library.

DESCRIPTION(*text*)

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

GROUP(*groupname*)

Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. For more information about groups, see [CICS Resource Definition Guide](#).

The GROUP name can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and \$. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

IPADDRESS

specifies the dotted decimal IP address on which this TCPIPSERVICE will listen for incoming connections. It must be of the form nnn.nnn.nnn.nnn where nnn is 0 through 255. Possible values are:

blank

When no IPaddress is specified, the default is *INADDR_ANY*

value

The TCPIPSERVICE accepts connections on this particular address. If the address specified is not known to TCP/IP on the z/VSE host, the TCPIPSERVICE will not open. If you enter a specific address here, this definition may not be valid for CICS servers running on other z/VSE partitions, and you may not be able to share the definition with those servers.

INADDR_ANY

The TCPIPSERVICE listens on any of the addresses known to TCP/IP for the z/VSE host. It is possible to have multiple IP addresses defined for a host. Specifying *INADDR_ANY* also allows for the TCPIPSERVICE definition to be shared among CICS servers.

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.

PORTNUMBER(*number*)

specifies, in the range 1 through 65535, the decimal number of the TCP/IP 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

PROTOCOL({HTTP|ECI|USER})

Specifies the application level protocol used on the TCP/IP port.

SOCKETCLOSE(NO|hhmms)

specifies if, 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, CICS waits for data for 30 seconds before it closes the socket. The interval is measured after issuing a receive for incoming data on the socket.

No

The socket remains open until it is closed by the client, or by a user application program in CICS.

hhmms

The interval (in HHMMSS format) from the time of the initial receive request for incoming data, after which CICS is to timeout (close) the socket. 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 specify PROTOCOL(ECI) 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, then re-install the TCPIPSERVICE resource with the modified definition.

SSL(NO|YES|CLIENTAUTH)

specifies the level of SSL to be used:

NO

No SSL support.

YES

SSL support is to be activated. Clients connecting to the specified port number must use the SSL protocol to connect with CICS (that is, they must specify `https` rather than `http` as the protocol in the URL used to access the service).

CLIENTAUTH

The CICS client, as well as the server, must have a certificate. The client certificate is received by CICS during the SSL handshake, and can be used either to determine the userid under which the CICS transaction can be executed, or to provide information about the client by means of the EXEC CICS EXTRACT CERTIFICATE command (described in [“EXTRACT CERTIFICATE”](#) on page 68).

If you specify SSL(CLIENTAUTH), this will not force the client to supply a certificate. Your client's Web browser program might not support client certificates, or the client might not have a certificate installed.

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.

TCPIP SERVICE(*name*)

specifies the 8-character name of this service.

TRANSACTION(*name*)

specifies the 4-character ID of the CICS transaction attached to process new requests received for this service. For a HTTP TCPIP SERVICE definition, specify CWXN or an alias of CWXN. For an ECI TCPIP SERVICE definition, specify CIEP (which is required together with its associated program DFHIEP). For a USER TCPIP SERVICE definition specify CWXU (or another transaction that executes program DFHWBXN).

TSQPREFIX

specifies the 6-character prefix of the temporary storage queue used to store inbound data and Web documents created by applications.

URM(*name*)

specifies the name of the user-replaceable module to be invoked by this service. The name you specify depends upon the value of the PROTOCOL attribute:

- For the HTTP protocol, specify the name of an analyzer program that is associated with this TCPIP SERVICE definition. The CICS-supplied default analyzer program DFHWBADX is the default.
- For the USER protocol, specify the name of an analyzer program that is associated with this TCPIP SERVICE definition. The analyzer program must be present, and it handles all requests on this protocol.

Part 7. CICS Supplied Transactions changes

Chapter 29. CEBR—temporary storage browse

Use the CEBR transaction to browse temporary storage queues and to delete them.

You can also use CEBR to transfer a transient data queue to temporary storage in order to look at its contents, and to recreate the transient data queue when you have finished. The CEBR commands that perform these transfers allow you to create and delete transient data queues as well.

Remember that:

- browsing of Temporary Storage queues retrieves the next record, following whichever record has most recently been retrieved by *ANY active task*.

This can lead to confusion if for example an EXEC CICS READQ NEXT and a CEBR transaction attempt to work with the same Temporary Storage queue at the same time.

- transient data queue reads are destructive. If you read a transient data queue that is being used on an active system by applications, this is likely to cause problems.

You start the CEBR transaction by entering the transaction identifier CEBR, followed by the name of the queue you want to browse. For example, to display the temporary storage queue named CEBRS209, you enter:

```
CEBR CEBRS209
```

CICS responds with a display of the queue, a sample of which is shown in [Figure 15 on page 167](#):

```
CEBR TSQ AXBYQUEUEENAME1  SYSID CIJP REC    1 OF    3  COL    1 OF    5
ENTER COMMAND ==>
***** TOP OF QUEUE *****
00001 HELLO
00002 HELLO
00003 HELLO
***** BOTTOM OF QUEUE *****

PF1 : HELP          PF2 : SWITCH HEX/CHAR    PF3 : TERMINATE BROWSE
PF4 : VIEW TOP      PF5 : VIEW BOTTOM        PF6 : REPEAT LAST FIND
PF7 : SCROLL BACK HALF PF8 : SCROLL FORWARD HALF PF9 : UNDEFINED
PF10: SCROLL BACK FULL PF11: SCROLL FORWARD FULL PF12: UNDEFINED
```

Figure 15. Typical CEBR screen displaying temporary storage queue contents

Alternatively, you can start the CEBR transaction from CEDF. You do this by pressing PF5 from the initial CEDF screen (see *CICS Supplied Transactions*) which takes you to the working-storage screen, and then you press PF2 from that screen to browse temporary storage (that is, you invoke CEBR). CEBR responds by displaying the temporary storage queue whose name consists of the four letters 'CEBR' followed by the four letters of your terminal identifier. (CICS uses this same default queue name if you invoke CEBR directly and do not supply a queue name.) The result of invoking CEBR without a queue name or from an EDF session at terminal S21A is shown in [Figure 16 on page 168](#).

If you enter CEBR from CEDF, you will return to the EDF panel when you press PF3 from the CEBR screen.

```

CEBR TSQ AXBYQUEUEAME1  SYSID CIJP REC  1 OF  0  COL  1 OF 10
ENTER COMMAND ==>
***** TOP OF QUEUE *****
*
***** BOTTOM OF QUEUE *****

TSQUEUE AXBYQUEUEAME1 DOES NOT EXIST
PF1 : HELP          PF2 : SWITCH HEX/CHAR    PF3 : TERMINATE BROWSE
PF4 : VIEW TOP      PF5 : VIEW BOTTOM      PF6 : REPEAT LAST FIND
PF7 : SCROLL BACK HALF PF8 : SCROLL FORWARD HALF PF9 : UNDEFINED
PF10: SCROLL BACK FULL PF11: SCROLL FORWARD FULL PF12: UNDEFINED

```

Note: 1|Header 2|Command line 3|Body 4|Message line 5|Menu of options

Figure 16. Typical CEBR display of default temporary storage queue

For information about the CEBR temporary storage browse transaction and guidance on using it, see the [CICS Application Programming Guide](#).

The HELP panel

If you press the help key (PF1), the following panel is displayed:

```

These commands are available to you (abbreviations in UPPER CASE):
Find /string/          - Keyword optional. Final delimiter optional if
                        string has no blanks. Any other delimiter is OK.

Line line-number
Column column-number
Top
Bottom
TERMinal terminal-id  - Browse temp. storage queue for another terminal.
Queue temp-stg-queue - Browse a named temp. storage queue
                        (name may be in hex - e.g., X'C134')

Sysid shared/remote sysid - Browse shared or remote temp. storage queue
Put transient-data-queue - Copy current queue into a transient data queue.
Get transient-data-queue - Fetch a transient data queue for browsing.
PURGE                 - Destroy the current queue.

```

Chapter 30. CEDX – the execution diagnostic facility

The CICS execution diagnostic facility (EDF) provides two transactions that you can use for testing application programs. These transactions—CEDF and CEDX—enable you to test application programs interactively without having to supply special program testing procedures.

CEDF

Use CEDF to invoke EDF for testing application programs that are associated with user transactions initiated from a terminal.

CEDX

Use CEDX to invoke EDF for testing application programs that are associated with non-terminal transactions.

CEDF transaction

The CEDF transaction is fully documented in the [CICS Supplied Transactions, SC33-1655](#) publication and is not further described here.

For information on how to use the CEDF transaction, see the [CICS Application Programming Guide, SC33-1657](#).

You must ensure that the EDF resource definitions are installed. These resource definitions are provided in the IBM-supplied group, DFHEDF. For information about installing the resource definitions, see the [CICS Resource Definition Guide, SC33-1653](#).

CEDX transaction

Use CEDX to monitor and debug non-terminal transactions. The transaction you specify for debugging can be:

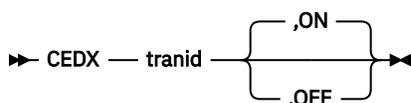
- Initiated without an associated terminal; for example, by an EXEC CICS START command, or by a transient data queue trigger-level.
- Initiated at a terminal, which can be either the EDF terminal or a different terminal.

CICS intercepts the transaction specified on the CEDX *tranid* command, and displays the EDF diagnostic panels at the terminal at which the EDF command is issued.

CEDX provides the same function and diagnostic display panels as CEDF, and the same basic rules for CEDF also apply to CEDX.

Command syntax

CEDX



Command options

OFF

specifies that the EDF screen is to be switched OFF. If you specify OFF you must enter the preceding comma, as shown in the following example:

```
CEDX TRNA,OFF
```

ON

specifies that the EDF screen is to be switched ON. The default is ON. If you specify ON you must enter the preceding comma, as shown in the following example:

```
CEDX TRNB,ON
```

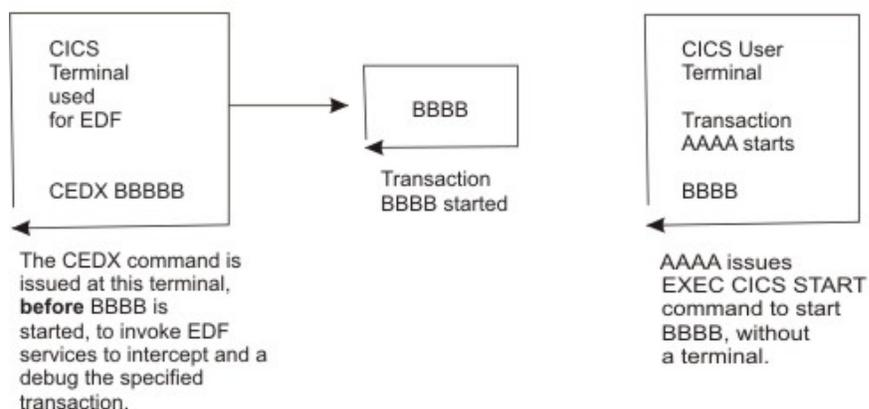
tranid

specifies the identifier of a transaction that you want to test using the execution diagnostic facility. The diagnostic panels are displayed on the terminal at which you issue the CEDX *tranid* command.

The transaction you specify on the CEDX command must run in the local CICS region.

CEDX cannot interrupt transactions that are already running when you issue the CEDX command. It affects only a transaction that starts *after* you issue the command.

The following diagram illustrates the use of CEDX to invoke EDF for a transaction initiated by an EXEC CICS START command:



Chapter 31. CEMT INQUIRE/SET commands

Most CEMT requests either inquire about (INQUIRE), or change (SET) the status of one or more named instances of a resource (such as a terminal), a particular subgroup of resources (such as a class of terminals), or all resources of a particular type (such as all terminals).

The INQUIRE command causes the status of the specified resources to be displayed. The SET command makes the changes that you specify, and displays the new status of the resources. No changes are made if there are syntax errors in the SET command.

If, for example, you want to inquire about or set a resource, enter INQUIRE (or a suitable abbreviation) on the command line. The keywords you can use with CEMT INQUIRE are described in the rest of this section.

If you want to perform those functions that are not concerned with resource status, enter PERFORM (or a suitable abbreviation) on the command line. The keywords you can use with PERFORM are described in the [CICS Supplied Transactions](#) publication.

And finally, if you want to change the attributes of a resource, enter SET (or a suitable abbreviation) on the command line. The keywords you can use with SET are described in the [CICS Supplied Transactions](#) publication.

If you enter INQUIRE, you get the following display:

```

INQ
STATUS:  ENTER ONE OF THE FOLLOWING OR HIT ENTER FOR DEFAULT
                                     ←Command Line
                                     ←Status Line

AUTInstmodel FENode      PROgram      TSQueue
AUTOinstall  INttrace    TDqueue
AUXtrace     IRc        RRms
COnnection   JOurnalname STATistics
DEletshipped MODename    STReamname
Doctemplate  MONitor     TSQueue
DSAs         Netname    Vtam
DSName       PArtner    TAsk
DUmpsds     PROFile    TCLass
EXci        PROGram    TCPIP
FEConnection STATistics  TCPIPService
FENode      SYDumpcode TDqueue
FEPOol      SYStem     TErminAl
ENQ         Task       TRAnSACTION
FEPRopset   TCLass     TRDumpcode
FETartget   TCPIP     TSMoDel
File        TCPIPService TSPool

PF 1 HELP      3 END      5 VAR
                                     SYSID=CICS APPLID=CICSHTC1
                                     9 MSG      ←PF keys

```

Figure 17. Sample of the screen following either the INQUIRE or the SET command

You can inquire about any of the displayed options by typing its keyword after INQUIRE on the command line. For example,

```
INQUIRE PROGRAM
```

gives you the status of all programs, and for each program gives its attributes. Full details are given in [CICS Supplied Transactions](#).

CEMT INQUIRE DOCTEMPLATE

Function

Retrieves information about a DOCTEMPLATE

Description

INQUIRE DOCTEMPLATE returns information about any currently installed document template names.

Input

Press the Clear key to clear the screen, and type CEMT INQUIRE DOCTEMPLATE (the minimum abbreviation is CEMT I DO). You get a display that lists the names and status of any document templates. Note that you cannot change any of the displayed information.

Sample screen

```
I DOC
STATUS: RESULTS - OVERTYPE TO MODIFY
Doc(test1 ) Tsq Nam(test1 ) App Ebc
  Tem(test1 )

RESPONSE: NORMAL          SYSID=HA61 APPLID=CICSHA61
PF 1 HELP      3 END      5 VAR      TIME: 15.54.22 DATE: 09.03.98
7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

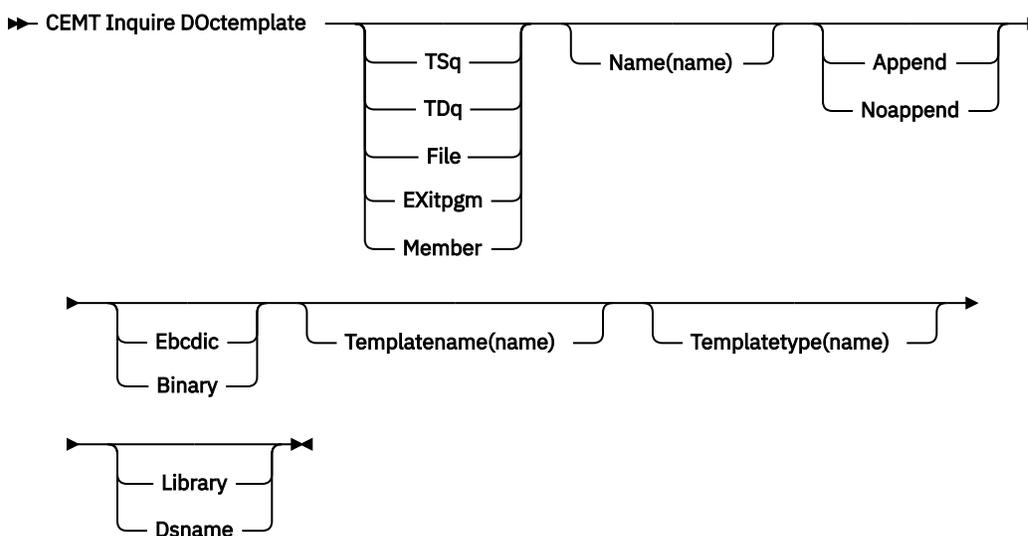
Figure 18. CEMT INQUIRE DOCTEMPLATE screen

If you place the cursor against a specific entry in the list and press ENTER, CICS displays an expanded format.

```
i doc
RESULT - OVERTYPE TO MODIFY
Doctemplate(test1)
Templatetype(Tsqueue)
Name(test1)
Appendcrlf(Append)
Type(Ebcdic)
Templatename(test1)
Library()
Dsname()
```

Figure 19. The expanded display of an individual entry

CEMT INQUIRE DOCTEMPLATE



parameters;**APPENDCRLF**

returns whether CICS is to delete trailing blanks from and append carriage-return line-feed to each logical record of the template.

LIBRARY

returns the file name of the library containing the template. The library applies only to a template of type LIBRARY. If a membername is supplied without a value for LIBRARY, the default value DFHHTML is used.

DOCTEMPLATE

returns the name of this DOCTEMPLATE definition.

DSNAME

returns the file-id of the library containing the template. The Dsname applies only to a template of type LIBRARY.

NAME

returns the name of the location defined in TEMPLATETYPE.

TEMPLATENAME

returns the extended template name by which the doctemplate is known outside the resource definition function.

TEMPLATETYPE

returns the type of resource whose name is returned in NAME.

EXITPGM

an exit program.

FILE

a CICS file name for a data set.

MEMBER

a name of the member in the library described in LIBRARY.

PROGRAM

a name of a program.

TDQUEUE

a name of a TD queue.

TSQUEUE

a name of a TS queue.

TYPE(BINARY|EBCDIC)

returns the format of the template contents.

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.

CEMT INQUIRE/SET TASK

Function

Retrieve information about a user task.

Description

INQUIRE TASK returns information about user tasks. Only information about user tasks can be displayed or changed; information about CICS-generated system tasks or subtasks cannot be displayed or changed.

CEMT INQUIRE/SET TASK

System tasks are those tasks started (and used internally) by CICS, and not as a result of a user transaction.

Input

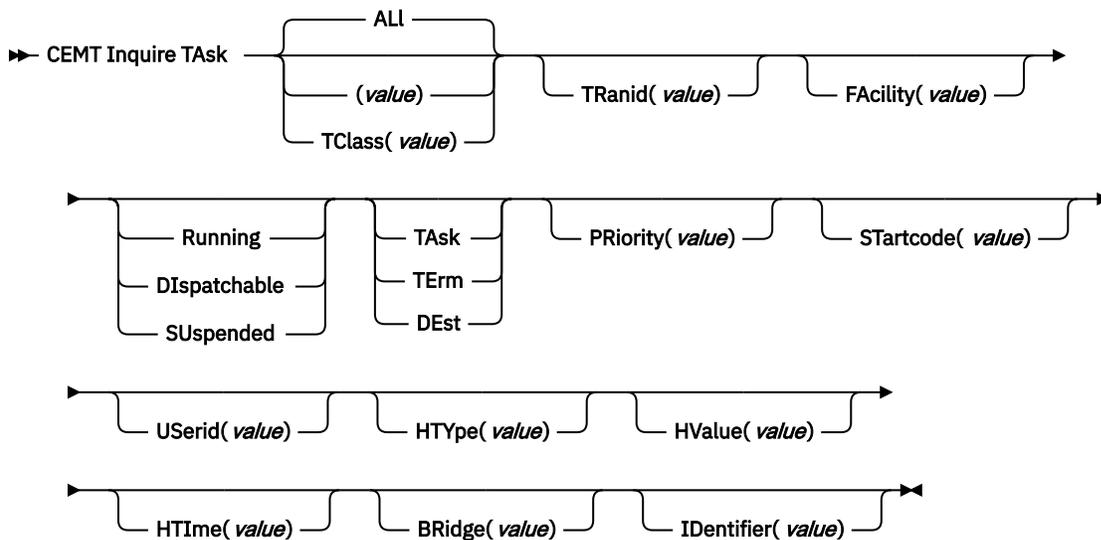
Press the Clear key to clear the screen. There are two ways of commencing this transaction:

- Type CEMT INQUIRE TASK (the minimum abbreviation is CEMT I TA). You get a display that lists the current status.
- Type CEMT INQUIRE TASK (CEMT I TA) followed by as many of the other attributes as are necessary to limit the range of information that you require. So, for example, if you enter `cemt i ta i`, the resulting display will show you the details of only those tasks for which the data is not shared with other tasks (isolated).

To change various attributes, you can:

- Overtyping your changes on the INQUIRE screen after tabbing to the appropriate field (see "Overtyping a display" in the [CICS Supplied Transactions](#) publication).
- Use the CEMT SET TASK command.

CEMT INQUIRE TASK



(value)

is the CICS-generated task number, in the range 1–999999.

ALL

is the default.

TClass(value)

is the 8-character transaction class name to which the transaction belongs.

You cannot specify a list of identifiers, nor can you use the symbols * and + to specify a family of tasks.

Sample screen

```

IN TASK
STATUS: RESULTS - OVERTYPE TO MODIFY
Tas(0000026) Tra(CEMT) Fac(S262) Sus Ter Pri( 255 )
  Sta(T0) Use(CICSUSER) Rec(X'B0C9D8D111440007') Hty(ZCIOWAIT)
Tas(0000030) Tra(CEMT) Fac(S263) Run Ter Pri( 255 )
  Sta(T0) Use(CICSUSER) Rec(X'B0C9D8E9C3B1FB09')

```

Figure 20. CEMT INQUIRE TASK screen

Note: There are blank fields on the screen where a value does not apply or is 'negative'; that is, the value begins with 'No'. To modify these fields, locate them by tabbing (they appear in the same sequence as in the expanded format), and overtype with input valid for that field. You may find it more convenient to use the expanded format when setting one of these values, (negating is easier because you are overtyping a displayed field with its negative causing the result to become non-displayed).

If you place the cursor against a specific entry in the list and press ENTER, CICS displays an expanded format as shown in [Figure 21 on page 175](#).

```

I TASK
RESULT - OVERTYPE TO MODIFY
Task(0000026)
Tranid(CEMT)
Facility(S262)
Runstatus(Suspended)
Ftype(Term)
Priority( 255 )
Purgetype(          )
Startcode(T0)
Userid(CICSUSER)
Recunitid(X'B4105A1E55031245')
Htype(ZCIOWAIT)
Hvalue(DFHZARQ1)
Htime(000159)
Bridge()
Identifier()

                                SYSID=JOHN APPLID=I
                                TIME: 11.26.40 DATE: 08
PF 1 HELP 2 HEX 3 END          5 VAR          7 SBH 8 SFH          10 SB 11 SF

```

Figure 21. The expanded display of an individual entry

parameters;**BRidge(value)**

Returns the 4-character name of the bridge monitor transaction if the current task is running in a 3270 bridge environment, and was started by a bridge monitor transaction with a START BREXIT TRANSID command. Otherwise, blanks are returned.

FACility(value)

displays a 4-character string identifying the name of the terminal or queue that initiated the task. If no FACILITY value is displayed, the task was started without a facility.

Ftype

displays the type of facility that initiated this task. The values are:

TAsk

The task was initiated from another task.

TErm

The task was initiated from a terminal.

DEst

The task was initiated by a destination trigger level as defined in the destination control table (DCT).

HTime(value)

displays the time (in seconds) that the task has been in the current suspended state.

HType(value)

displays the reason why the task is suspended. A null value indicates that there is no hold-up, except for the necessity of reaching the head of the queue.

HValue(value)

displays a 16-character resource name, such as a file name, or a value such as a TCLASS value.

For information on the values that can appear in the HTYPE and HVALUE options, and how they can be used as an aid in problem determination, see the “resource type” and “resource name” details in the [CICS Problem Determination Guide](#).

IDentifier(value)

returns a 48-character field containing user data provided by the bridge exit, if the task was initiated in the 3270 bridge environment, or blanks, otherwise. This field is intended to assist in online problem resolution.

For example, it could contain the MQ™ correlator for the MQ bridge, or a Web token.

PRiority(value)

displays the priority of the task, in the range 0–255 where 255 is the highest priority.

Note: You can reset this value by overtyping it with a different value.

Purgetype (input only field)

specifies whether a task is to be purged or forcepurged. The values are:

Purge

The task is to be terminated. Termination occurs only when system and data integrity can be maintained.

Forcepurge

The task is to be terminated immediately. System integrity is not guaranteed. In some extreme cases, for example if a task is forcepurged during backout processing, CICS terminates abnormally. If you want to terminate a task but do not want to terminate CICS, you should use PURGE instead of FORCEPURGE.

In some BTAM situations, further user action is required to complete the purging of the task. If a task is in a terminal read on a BTAM nonlocal terminal and that task is canceled, it can require input from another terminal on the same line before the cancel operation completes.

Recunitid(value)

displays a unique identifier of the current unit of recovery. The identifier takes the form of an eight byte clock value and is displayed in hexadecimal format.

Runstatus

displays the status of this task. The values are:

Running

The task is running.

DIspatchable

The task is dispatchable.

SUspended

The task is suspended.

STartcode(value)

displays how this task was started. The values are:

D

A distributed program link (DPL) request. The program cannot issue I/O requests against its principal facility or any syncpoint requests.

DS

A distributed program link (DPL) request, as for code D, with the exception that the program can issue syncpoint requests.

QD

A transient data trigger level was reached.

S

Start command (no data)

SD

Start command (with data)

TO

The operator typed a transaction code at the terminal.

TP

The transaction was started by presetting the transaction ID for the terminal.

U

User-attached task.

TAsk(value)

indicates that this panel relates to a TASK inquiry and displays a CICS-generated task number in the range 1–99999.

TRanid(value)

displays a 4-character string identifying the transaction name associated with the task.

USerid(value)

displays the user currently associated with the task.

CEMT INQUIRE/SET TCPIP

Function

Inquire about CICS internal TCP/IP support status.

Description

INQUIRE TCPIP returns information about the current status of CICS internal TCP/IP support.

Input

Press the Clear key to clear the screen. There are two ways of commencing this transaction:

- Type CEMT INQUIRE TCPIP (the minimum abbreviation is CEMT I TCPIP). You get a display that lists the current status.
- Type CEMT INQUIRE TCPIP (CEMT I TCPIP) followed by as many of the other attribute settings that you wish to view.

To change various attributes, you can:

- Overtyping your changes on the INQUIRE screen after tabbing to the appropriate field (see [CICS-Supplied Transactions](#) publication).
- Use the CEMT SET TCPIP command.

CEMT INQUIRE/SET TCPIP SERVICE

CEMT INQUIRE TCPIP



OPENSTATUS(value)

displays the status of CICS internal TCP/IP support. The values are:

OPEN

CICS internal sockets support is open.

CLOSED

CICS internal sockets support has not yet been activated, or has been terminated.

CLOSING

CICS internal sockets support is in the process of closing.

IMMCLOSING

CICS internal sockets support is in the process of immediate termination.

CEMT INQUIRE/SET TCPIP SERVICE

Function

Retrieve information about TCP/IP ports on which CICS internal TCP/IP support is currently listening on behalf of other CICS services.

Description

INQUIRE TCPIP SERVICE returns information about the state of a service using CICS internal TCP/IP support.

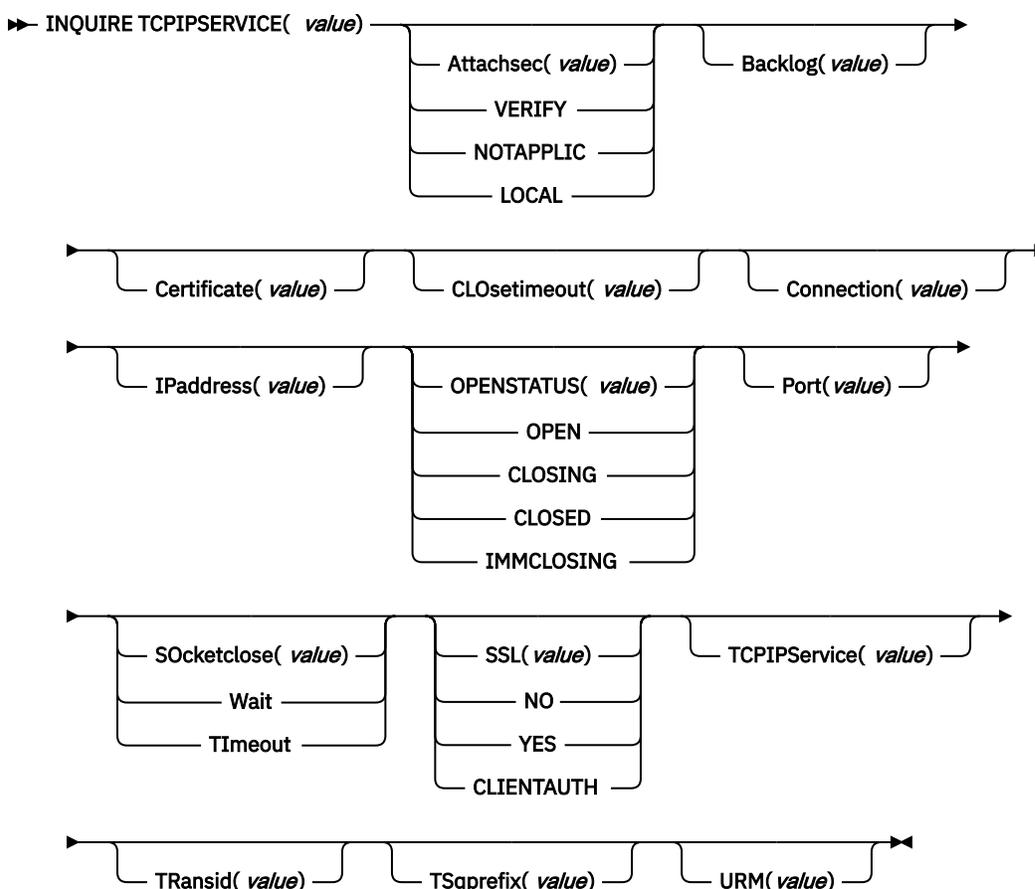
Input

Press the Clear key to clear the screen. There are two ways of commencing this transaction:

- Type CEMT INQUIRE TCPIP SERVICE (the minimum abbreviation is CEMT I TCPIPS). You get a display that lists the current status.
- Type CEMT INQUIRE TCPIP SERVICE (CEMT I TCPIPS) followed by as many of the other attribute settings that you wish to view.

To change various attributes, you can:

- Overtyping your changes on the INQUIRE screen after tabbing to the appropriate field.
- Use the CEMT SET TCPIP SERVICE command.

CEMT INQUIRE TCPIP SERVICE**Attachsec**

returns the ECI security requirements of a CICS client. Attachsec values are:

VERIFY

The CICS client must provide a userid and password to the z/VSE host. These are checked before the CICS client is allowed to communicate with the z/VSE host.

NOTAPPLIC

This option has no meaning for the web interface.

LOCAL

The CICS client is not required to provide a userid and password to the z/VSE host.

Backlog

Change the maximum number of requests which can be queued in TCP/IP waiting to be processed by the service.

Certificate

returns the label of a server certificate stored in the VSE Keyring Library. If your own certificate has not been specified, the label of the default server certificate in the VSE Keyring Library is returned.

Closetimeout

returns a fullword value containing the number of seconds specified for the timeout period. This can be 0 through 86400 (24 hours).

Connection

The number of current sockets connections for this service

Ipaddress

returns an IP address. If you specify an IP address in the TCPIP SERVICE definition, that address is returned, otherwise the default IP address is returned. If there is more than one IP address, only the default is returned.

MAXDATALEN(value)

Returns a fullword value that contains the maximum length of data that can be received by CICS as an HTTP server. This value can be 3 KB through 524288 KB. The default is 32 KB.

Port

returns the number of the port on which CICS is listening on behalf of this service.

PROTOCOL(value)

identifies to CICS the type of service to be provided on the TCP/IP port.

ECI

Connections are handled by CICS ECI over TCP/IP support.

HTTP

Connections are handled by CICS web support.

USER

The user-defined protocol is used. Requests are passed to the analyzer program for the TCPIPService, and handled by using CICS web support facilities, but the HTTP specifications are not used to check the messages.

Socketclose

returns a CVDA value indicating whether a timeout value is in effect for the TCPIPService. CVDA values are:

WAIT

NO was specified on the definition. Socket receives will wait for data indefinitely.

TIMEOUT

A value was specified for the SOCKETCLOSE parameter on the definition.

SSL

returns the level of SSL that has been activated. SSL values are:

NO

SSL is not activated.

YES

SSL support is activated. Clients connecting to the specified port number use the SSL protocol to connect with CICS (that is, they must specify `https` rather than `http` as the protocol in the URL used to access the service).

CLIENTAUTH

The client, as well as the server, must have a certificate. The client certificate is received by CICS during the SSL handshake, and can be used either to determine the userid under which the CICS transaction can be executed, or to provide information about the client by means of the EXEC CICS EXTRACT CERTIFICATE command (described in [“EXTRACT CERTIFICATE” on page 68](#)).

TCPIPService

8-character name identifying this service.

OPENSTATUS(cvda)

returns a CVDA value indicating the state of the service. CVDA values are:

OPEN

CICS internal sockets support is open.

CLOSED

CICS internal sockets support has not yet been activated, or has been terminated.

CLOSING

CICS internal sockets support is in the process of closing.

IMMCLOSING

CICS internal sockets support is in the process of immediate termination.

Transid

4-character transaction id used on the attach for the task started to process a new request. For a Web TCPIPService definition, CWXN or an alias of CWXN will be returned. For ECI, CIEP will be returned.

TSQprefix

returns the 6-character prefix used when the service generates TS queue names.

URM

8 character name of Service URM invoked by attached task.

CEMT INQUIRE/SET TRANSACTION

Function

Retrieve information about transactions.

Description

INQUIRE TRANSACTION returns information about transaction definitions.

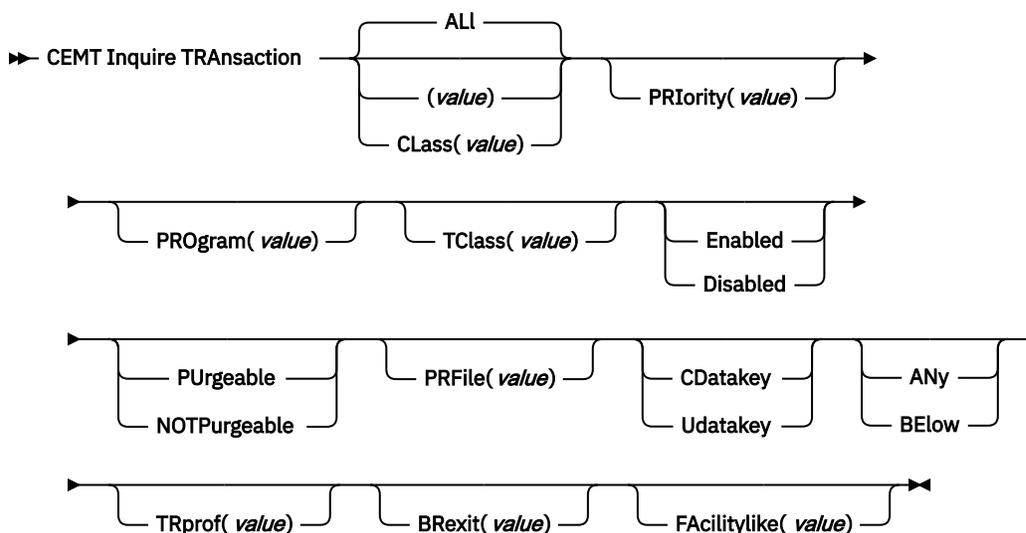
Input

Press the Clear key to clear the screen. There are two ways of commencing this transaction:

- Type CEMT INQUIRE TRANSACTION (the minimum abbreviation is CEMT I TRANS). You get a display that lists the current status.
- Type CEMT INQUIRE TRANSACTION (CEMT I TRANS) followed by as many of the other attributes as are necessary to limit the range of information that you require. So, for example, if you enter `cemt i trans en pu`, the resulting display will show you the details of only those transactions that are enabled and system-purgeable.

To change various attributes, you can:

- Overtyping your changes on the INQUIRE screen after tabbing to the appropriate field (see [CICS Supplied Transactions](#) publication).
- Use the CEMT SET TRANSACTION command.

CEMT INQUIRE TRANSACTION**(value)**

is a 1–4 character transaction identifier. Only transactions that have been defined in the CICS system definition (CSD) file and installed on the running CICS system are accessible through CEMT.

ALL

is the default.

Class(value)

is the 2-character suffix of a transaction list table (XLT).

Sample screen

```

IN TRAN
STATUS: RESULTS - OVERTYPE TO MODIFY
Tra(AADD) Pri( 001 ) Pro(DFH$AALL) Tc1( DFHTCL00 ) Ena Pur
  Prf(DFHCICST) Uda Bel
Tra(ABRW) Pri( 001 ) Pro(DFH$ABRW) Tc1( DFHTCL00 ) Ena Pur
  Prf(DFHCICST) Uda Bel
Tra(ADYN) Pri( 001 ) Pro(DFH99 ) Tc1( DFHTCL00 ) Ena Pur
  Prf(DFHCICST) Uda Bel
Tra(AINQ) Pri( 001 ) Pro(DFH$AALL) Tc1( DFHTCL00 ) Ena Pur
  Prf(DFHCICST) Uda Bel
Tra(AMNU) Pri( 001 ) Pro(DFH$AMNU) Tc1( DFHTCL00 ) Ena Pur
  Prf(DFHCICST) Uda Bel
Tra(AORD) Pri( 001 ) Pro(DFH$AREN) Tc1( DFHTCL00 ) Ena Pur
  Prf(DFHCICST) Uda Bel
Tra(AORQ) Pri( 001 ) Pro(DFH$ACOM) Tc1( DFHTCL00 ) Ena Pur
  Prf(DFHCICST) Uda Bel
Tra(AREP) Pri( 001 ) Pro(DFH$AREP) Tc1( DFHTCL00 ) Ena Pur
  Prf(DFHCICST) Uda Bel
+ Tra(AUPD) Pri( 001 ) Pro(DFH$AALL) Tc1( DFHTCL00 ) Ena Pur
  Prf(DFHCICST) Uda Bel

```

Figure 22. CEMT INQUIRE TRANSACTION screen

If you place the cursor against a specific entry in the list and press ENTER, CICS displays an expanded format as shown in [Figure 23 on page 182](#).

```

IN TRAN
RESULT - OVERTYPE TO MODIFY
Transaction(AADD)
Priority( 001 )
Program(DFH$AALL)
Tclass( DFHTCL00 )
Status( Enabled )
Purgeability( Purgeable )
Prfile(DFHCICST)
Taskdatakey(Udatakey)
Taskdataloc(Below)
Tprof()
Brexid
Facilitylike()

```

Figure 23. The expanded display of an individual entry

parameters;**BRexit(value)**

returns the 8-character name of the bridge exit defined by the BREXID parameter of the named transaction resource definition.

If BREXID is not defined, blanks are returned.

FACilitylike(value)

returns the 4-character name of the terminal defined by the FACILITYLIKE parameter of the PROFILE associated with the named transaction resource definition.

PRFile(value)

displays the name of the profile definition that defines additional options associated with this transaction.

PRIority(value)

displays a value indicating the priority of a transaction relative to other transactions. When a transaction is running as a CICS task, the priority of a task is the sum of the transaction priority, the terminal priority, and the operator priority.

Note: You can reset this value by overtyping it with a different value.

The value is in the range 0–255, where 255 is the highest priority.

PROgram(value)

displays an 8-character string identifying the name of the first program to be executed when this transaction is started.

Purgeability

displays whether the transaction is purgeable in system stall conditions. The values are:

PURgeable

The transaction is system-purgeable. This value relates to the SPURGE parameter on the transaction resource definition and indicates that CICS can purge the transaction in a deadlock time-out situation. See the CICS Resource Definition Guide for information about the SPURGE and DTIMEOUT parameters on a transaction resource definition.

NOTPURgeable

The transaction cannot be purged.

Note: You can reset this value by overtyping it with a different value.

Status

displays whether the transaction is available for use. The values are:

Enabled

The transaction is available for use.

Disabled

The transaction is not available for use.

Note: If a transaction is disabled, this does not prevent a START command that names this transaction from being shipped to a remote region. When a task is attached for the requested transaction, CICS checks that the transaction is enabled in the remote region.

Note: You can reset this value by overtyping it with a different value.

Taskdatakey

displays the storage key in which CICS obtains all storage for use by the transaction. This includes the task life-time storage—the transaction work area (TWA) and the EXEC interface block (EIB)—and the storage that CICS obtains on behalf of programs that run under the transaction.

The values are:

CDatakey

CICS obtains storage for the transaction from CICS-key storage. Application programs that execute in CICS key have read-write access to this storage, but user-key programs have read-only access.

Udatakey

CICS obtains storage for the transaction from user-key storage. Application programs that execute in any key have read-write access to this storage.

See the description of the TASKDATAKEY parameter on the transaction resource definition in the [CICS Resource Definition Guide](#).

Taskdataloc

displays whether certain CICS control blocks (including EIB and TWA) for a transaction are acquired above or below the 16MB line. The values are:

ANy

The transaction accepts task-related data anywhere.

BElow

The transaction requires any task-related data (TWA and EIB plus any internal control blocks) to be located below the 16MB line.

TClass(value)

displays an 8-character string identifying the name of the transaction class to which the transaction belongs. If the transaction does not belong to a class, DFHTCL00 is returned.

Note: You can reset this value by overtyping it with a different value.

To remove a transaction from its TCLASS, set this field to DFHTCL00. An added or changed TCLASS **must** be one that has already been defined.

TRAnSACTION(value)

indicates that this panel relates to a TRANSACTION inquiry and displays a 4-character transaction identifier. Only transactions that have been defined in the CICS system definition (CSD) file and installed on the running CICS system are accessible through CEMT.

TRprof(value)

displays the name of the transaction routing profile that defines additional options associated with this transaction if it is defined as a remote transaction.

CEMT INQUIRE/SET TSQUEUE

Function

Retrieve information about temporary storage queues.

Description

The INQUIRE TSQUEUE command returns information about temporary storage queues (TS queues). The INQUIRE TSQUEUE command operates on all the temporary storage queues that exist in the CICS region, including those created internally by CICS for use by CICS itself (for example, queues used by BMS). You can identify the temporary storage queues created by CICS for its own use by queue names that begin with the following character strings:

BMS paging

\$\$

BMS route

X'fa' to X'ff'

CICS

CEBR

Default CEBR queue name

DF

CICS

DFHM

Message cache for message-protected tasks

DFxxxx

CICS REQIDS (where x is hexadecimal)

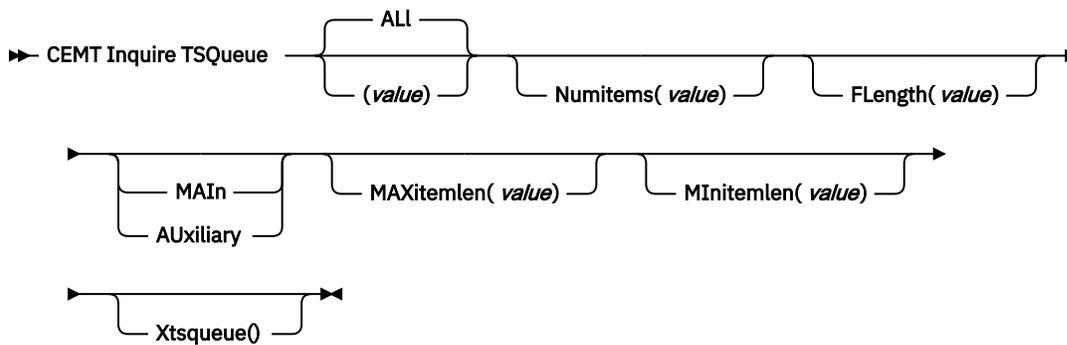
Input

Press the Clear key to clear the screen. There are two ways of commencing this transaction:

- Type CEMT INQUIRE TSQUEUE (the minimum abbreviation is CEMT I TSQ). You get a display that lists the current status.
- Type CEMT INQUIRE TSQUEUE (CEMT I TSQ) followed by as many of the other attributes as are necessary to limit the range of information that you require. So, for example, if you enter `cemt i tsq main`, the resulting display will show you the details of only those temporary storage queues that are resident in main storage.

To change various attributes, overtype your changes on the INQUIRE screen after tabbing to the appropriate field (see [CICS Supplied Transactions](#) publication).

CEMT INQUIRE TSQUEUE



ALL

is the default.

(value)

is the name of the temporary storage queue for which information is requested.

Sample screen

```

INQ TSQ
STATUS: RESULTS - OVERTYPE TO MODIFY
Tsq(ABCD ) Num(00001) Fle(0000000064) Mai
      Max(00064) Max(00064) Xts(C1D5C4E8840404040)
Tsq(. . . . .) Num(00003) Fle(00000000192) Aux
      Max(00064) Min(00064) Xts(01020304AABBCCDD)

```

Figure 24. CEMT INQUIRE TSQUEUE screen

If you place the cursor against a specific entry in the list and press ENTER, CICS displays an expanded format as shown in [Figure 25](#) on page 185.

```

INQ TSQ
RESULT - OVERTYPE TO MODIFY
Tsqqueue (AXBYQUEUEUENAME1)
Numitems (00003)
Flength(00000000192)
Location(Main)
Maxitemlen(00064)
Minitemlen(00064)
XTsqqueue(C1E7C2E8D8E4C5E4)

```

Figure 25. The expanded display of an individual entry

If you place the cursor against a specific entry in the list and type a 'B' (upper or lower case), CICS invokes the CEBR transaction to browse the contents of this TS queue, (see [Chapter 29, "CEBR—temporary storage browse,"](#) on page 167 for details).

parameters;

Flength(value)

displays the total length in bytes of all the items in the temporary storage queue. For information about how CICS calculates the length of items, see the [CICS System Programming Reference](#).

LLocation(value)

displays where the temporary storage queue resides. The values are:

AUxiliary

The temporary storage queue is held on the CICS temporary storage VSAM data set DFHTEMP, or in temporary storage pools in the coupling facility.

MAIn

The temporary storage queue is held in main storage.

MAXitemlen(value)

displays the length in bytes of the largest item in the temporary storage queue. For information about how CICS calculates the length of items, see the [CICS System Programming Reference](#).

MINitemlen(value)

displays the length in bytes of the smallest item in the temporary storage queue. For information about how CICS calculates the length of items, see the [CICS System Programming Reference](#).

Numitems(value)

displays the number of items in the temporary storage queue.

TSqueue(value)

indicates that this panel relates to a TSQUEUE inquiry and displays the 16-character name of a temporary storage queue.

Note: Non-displayable characters appear as periods. You can use PF2 on the expanded panel to see the value in hexadecimal.

Part 8. CICS channels and containers

This part of the publication describes how you can use CICS channels and containers within your CICS programs.

Chapter 32. Overview of CICS channels and containers

This topic provides an overview of the terms and concepts, when using channels and containers in a CICS Transaction Server for z/VSE (short form *CICS TS for z/VSE*) environment.

Instead of using a communication area (COMMAREA), a more modern method of transferring data between CICS programs is to use **a channel (and its containers)**. Channels have several advantages over COMMAREAs. A COMMAREA is limited to 32KB. With channels there is no limitation, except the storage that is available in the CICS partition. The channel and container approach provides an easy and flexible way for exchanging large amount of structured data between CICS programs.

CICS TS for z/VSE supports channel and containers through the EXEC CICS application programming interface (API) for use within CICS programs. The channels and containers API was first introduced with the CICS Transaction Server for z/OS 3.1 and was extended several times since then. CICS TS for z/VSE supports a subset of the functionality which is available on CICS Transaction Server for z/OS.

EXEC CICS provides a set of API commands that can be used to perform actions on channels and containers (see [Chapter 33, “CICS API commands for using channels and containers,”](#) on page 191 for application programming reference information).

- New CICS API commands have been introduced, which are described here.
- Existing CICS API commands have been extended with the CHANNEL option. For these commands, only the CHANNEL option is described here. Therefore, you also require each command’s description in the [CICS Transaction Server for VSE/ESA, Application Programming Reference, SC33-1658](#).

[Chapter 34, “How to use channels and containers,”](#) on page 213 complements the reference information. It gives guidance about the development of new programs that use channels and containers or the migration of existing programs to replace COMMAREAs by channels and containers.

Note: There is no need to migrate existing programs from COMMAREAs to channels if the used interface works well for you.

Languages supporting the channels and containers API:

- HLASM
- COBOL
- C
- PL/I

AMODE/RMODE Considerations:

- CICS programs can execute in either 24-bit or 31-bit addressing mode (AMODE).
- CICS TS for z/VSE does not support CICS programs executing AMODE 64.
- CICS TS for z/VSE does not support the use of extended (8-byte) registers.
- The channels and containers API can be used in either AMODE 24 or AMODE 31.
- CICS TS for z/VSE only supports the 31-bit channels and containers API. Therefore, the channels and containers data must be stored in either 24-bit or 31-bit storage.
- When migrating from COMMAREA to channels and containers, consider to also migrate to AMODE 31 in order to reduce 24-bit storage requirements.

Source code compatibility with CICS Transaction Server for z/OS:

The CICS TS for z/VSE channels and containers API provides source code compatibility with z/OS: the program can be developed on CICS Transaction Server for z/OS (with the functionality supported by CICS TS for z/VSE) but has to be compiled and linked on CICS TS for z/VSE.

Chapter 33. CICS API commands for using channels and containers

This topic provides "reference type" information for each CICS API command that you can use together with channels and containers.

These are the new API commands that were introduced with CICS TS for z/VSE 2.1:

- DELETE CONTAINER (CHANNEL)
- ENBBROWSE CONTAINER
- GET CONTAINER (CHANNEL)
- GETNEXT CONTAINER (CHANNEL)
- MOVE CONTAINER (CHANNEL)
- PUT CONTAINER (CHANNEL)
- START TRANSID CHANNEL
- STARTBROWSE CONTAINER

These are existing API commands for which the channel option was added:

- ASSIGN
- LINK
- RETURN
- XCTL

Note: BTS services in CICS TS for z/OS have been extended to also support containers. Therefore, the CICS TS for z/OS application reference manual distinguishes between container (channel) and container (BTS) commands. There is no BTS support in z/VSE. However, to use the same naming convention as in CICS TS for z/OS, the commands in CICS TS for z/VSE are also named container (channel).

ASSIGN

Request values from outside the local environment of the application program. This topic describes the CHANNEL option only.

ASSIGN

➤ ASSIGN ————— ➤
 └── CHANNEL(*data-value*) ───┘

Conditions: INVREQ

Description

ASSIGN gets values from outside the local environment of the application program.

Options

CHANNEL(*data-value*)

returns the 16-character name of the program's current channel, if one exists; otherwise blanks.

Conditions

16 INVREQ

RESP2 values: Unchanged when using the CHANNEL option.

DELETE CONTAINER (CHANNEL)

Delete a named channel container.

DELETE CONTAINER (CHANNEL)

►► DELETE — CONTAINER(*data-value*) — CHANNEL(*data-value*)

Conditions: CHANNELERR, CONTAINERERR, INVREQ

Description

DELETE CONTAINER (CHANNEL) deletes a container from a channel and discards any data that it contains.

The container is identified by name and by the channel for which it is a container - the channel that “owns” it. The channel that owns the container can be identified:

- Explicitly, by specifying the CHANNEL option.
- Implicitly, by omitting the CHANNEL option. If this is omitted, the current channel is implied.

Options

CHANNEL(*data-value*)

specifies the name (1–16 characters) of the channel that owns the container.

CONTAINER(*data-value*)

specifies the name (1–16 characters) of the container to be deleted.

Conditions

122 CHANNELERR

RESP2 values:

2

The channel specified on the CHANNEL option could not be found.

3

Either the current channel or the channel specified on the CHANNEL option is read-only.

110 CONTAINERERR

RESP2 values:

10

The container named on the CONTAINER option could not be found.

16 INVREQ

RESP2 values:

4

The command was issued outside the scope of a currently-active channel.

30

You cannot delete a CICS-defined read-only container.

ENDBROWSE CONTAINER

End a browse of the containers associated with a channel. You must specify the system programmer (SP) parameter in the EXEC statement of your compile job’s translate step.

ENDBROWSE CONTAINER

➤ ENDBROWSE — CONTAINER — BROWSETOKEN(*data-value*) ➤

Conditions: TOKENERR

Description

ENDBROWSE CONTAINER ends a browse of the containers associated with a channel and invalidates the browse token.

Options

BROWSETOKEN(*data-value*)

specifies, as a fullword binary value, the browse token to be deleted.

Conditions

112 TOKENERR

RESP2 values:

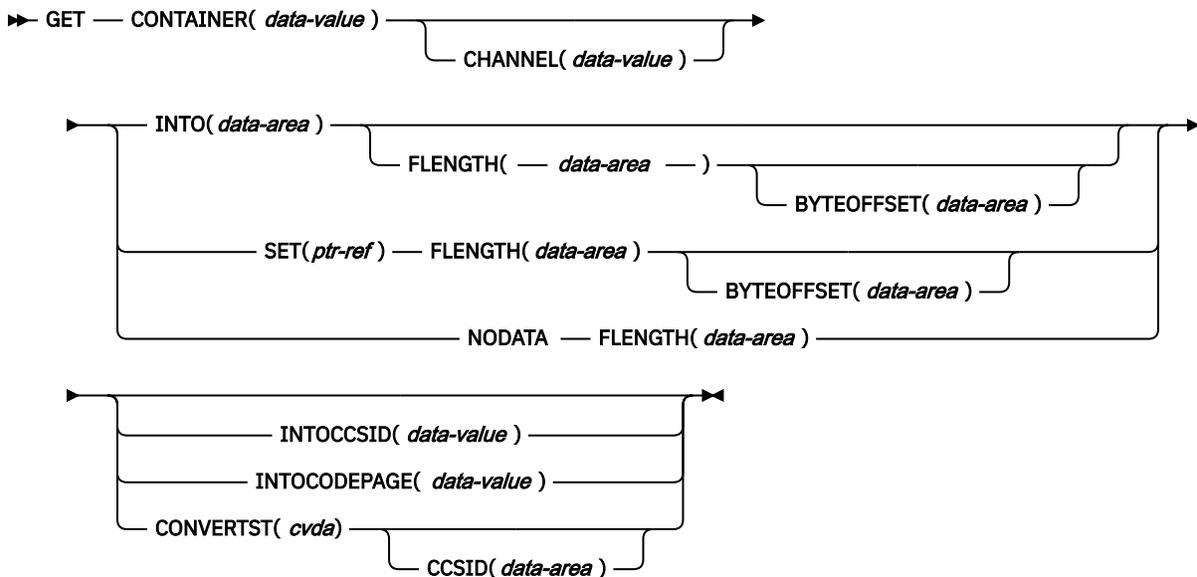
3

The browse token is not valid.

GET CONTAINER (CHANNEL)

Retrieve data from a named channel container.

GET CONTAINER (CHANNEL)



Conditions: CCSIDERR, CHANNELERR, CODEPAGEERR, CONTAINERERR, INVREQ, LENGERR

Description

GET CONTAINER (CHANNEL) reads the data associated with a specified channel container.

The container that holds the data is identified by name and by the channel for which it is a container - the channel that "owns" it. The channel that owns the container can be identified:

- Explicitly, by specifying the CHANNEL option.
- Implicitly, by omitting the CHANNEL option. If this is omitted, the current channel is implied.

Options

BYTEOFFSET(*data-value*)

Specifies the offset in bytes where the data returned starts. For CHAR containers, the BYTEOFFSET value is used as an offset into the data in the requested codepage. If you use a codepage with multibyte characters, depending on the BYTEOFFSET value you specify, the data returned might have partial characters at the beginning, end, or both. In this situation, your application program must be able to handle and interpret the data returned. If the value specified is less than zero, zero is assumed.

CCSID(*data-value*)

Returns a fullword that contains the Coded Character Set Identifier (CCSID) of the data returned by CONVERTST(NOCONVERT) option. This option allows you to retrieve containers with a DATATYPE of CHAR, without converting the data. If a DATATYPE of BIT is specified for the container, this value is zero.

CHANNEL(*data-value*)

Specifies the name (1–16 characters) of the channel that owns the container.

CONTAINER(*data-value*)

Specifies the name (1–16 characters) of the container that holds the data to be retrieved.

CONVERTST(*cvda*)

Specifies the required data conversion status.

NOCONVERT

The container data is retrieved without being converted.

FLENGTH(*data-area*)

As an input field, FLENGTH specifies, as a fullword binary value, the length of the data to be read. As an output field, FLENGTH returns the length of the data in the container. Whether FLENGTH is an input or an output field depends on which of the INTO, SET, or NODATA options you specify.

INTO option specified

FLENGTH is both an input and an output field.

On input, FLENGTH specifies the maximum length of the data that the program accepts. If the value specified is less than zero, zero is assumed. If the length of the data exceeds the value specified, the data is truncated to that value and the LENGERR condition occurs. If the length of the data is less than the specified value, the data is copied but no padding is performed.

FLENGTH need not be specified if the length can be generated by the compiler from the INTO variable. If you specify both INTO and FLENGTH, FLENGTH specifies the maximum length of the data that the program accepts.

On output (that is, on completion of the retrieval operation) CICS sets the data area, if specified, to the actual length of the data in the container. If the container holds character data that has been converted from one CCSID to another, this is the length of the data *after conversion*.

SET or NODATA option specified

FLENGTH is an output-only field. It must be present and must be specified as a data-area.

On completion of the retrieval operation, the data area is set to the actual length of the data in the container. If the container holds character data that has been converted from one CCSID to another, this is the length of the data *after conversion*.

INTO(*data-area*)

Specifies the data area into which the retrieved data is to be placed.

INTOCCSID(*data-value*)

Specifies the Coded Character Set Identifier (CCSID) into which the character data in the container is to be converted, as a fullword binary number. If you prefer to specify an IANA name for the code page, or if you prefer to specify the CCSID as alphanumeric characters, use the INTOCODEPAGE option instead.

For CICS Transaction Server for z/VSE applications, the CCSID is typically an EBCDIC CCSID. (However, it is possible to specify an ASCII CCSID if, for example, you want to retrieve ASCII data without it being automatically converted to EBCDIC.)

If INTOCCSID and INTOCODEPAGE are not specified, the value for conversion defaults to the CCSID of the CICS partition. The default CCSID of the CICS partition is specified on the **LOCALCCSID** system initialization parameter.

Only character data can be converted, and only then if a DATATYPE of CHAR was specified on the **PUT CONTAINER** command used to place the data in the container. (A DATATYPE of CHAR is implied if FROMCCSID or FROMCODEPAGE is specified on the **PUT CONTAINER** command.)

For more information about data conversion with channels, refer to Chapter 34, “How to use channels and containers,” on page 213. For an explanation of CCSIDs, see “Data conversion” on page 231.

INTOCODEPAGE(data-value)

Specifies an IANA-registered alphanumeric charset name or a Coded Character Set Identifier (CCSID) for the code page into which the character data in the container is to be converted, using up to 40 alphanumeric characters, including appropriate punctuation. Use this option instead of the CCSID option if you prefer to use an IANA-registered charset name, as specified in the Content-Type header for an HTTP request. CICS converts the IANA name into a CCSID, and the subsequent data conversion process is identical. Also use this option if you prefer to specify the CCSID in alphanumeric characters, rather than as a fullword binary number.

For more information about data conversion with channels, refer to Chapter 34, “How to use channels and containers,” on page 213. For an explanation of CCSIDs, see “Data conversion” on page 231.

NODATA

Specifies that no data is to be retrieved. Use this option to discover the length of the data in the container (returned in FLENGTH).

The length of character data may change if data conversion takes place. Therefore, if character data is to be converted into any CCSID *other than the default CCSID of this partition*, when you specify NODATA you should also specify INTOCCSID. This ensures that the correct length of the converted data is returned in FLENGTH.

SET(ptr-ref)

Specifies a data area in which the address of the retrieved data is returned. If the application program that issues the GET CONTAINER command is defined with DATALOCATION(ANY), the address of the data can be above or below the 16 MB line. If the application program is defined with DATALOCATION(BELOW), the address of the data is below the 16 MB line. If TASKDATAKEY(USER) is specified for the executing transaction, the data returned is in user key; otherwise, it is in CICS key.

CICS maintains the data area until any of the following occurs:

- A subsequent GET CONTAINER command with the SET option, for the same container in the same channel, is issued by any program that can access this storage.
- The container is deleted by a DELETE CONTAINER command.
- The container is moved by a MOVE CONTAINER command.
- The channel goes out of program scope.

Beware of linking to other programs that might issue one of the above commands.

Do *not* issue a FREEMAIN command to release this storage.

If your application needs to keep the data, it should move it into its own storage.

Note: Be aware of additional storage requirements when using the SET option. For containers with a DATATYPE of CHAR, this can be up to four times the size of the container due to data conversion. Depending on the DATALOCATION, this can be 24-bit storage.

Conditions

123 CCSIDERR

RESP2 values:

- 1**
The CCSID specified on the INTOCCSID option is outside the range of valid CCSID values.
- 2**
The CCSID specified on the INTOCCSID option and the CCSID of the container are an unsupported combination. (The CCSID of the container is the value that was specified using either FROMCCSID or FROMCODEPAGE, or defaulted, when the container was built.)
- 3**
The data was created with a data-type of BIT. Code page conversion is not possible. The data was returned without any code page conversion.
- 4**
One or more characters could not be converted. The character has been replaced by a blank in the converted data.
- 5**
There was an internal error in the code page conversion of a container.

122 CHANNELERR

RESP2 values:

- 2**
The channel specified on the CHANNEL option could not be found.

125 CODEPAGEERR

RESP2 values:

- 1**
The code page specified on the INTOCODEPAGE option is not supported.
- 2**
The code page specified on the INTOCODEPAGE option and the code page of the channel are an unsupported combination.
- 3**
The data was created with a data-type of BIT. Code page conversion is not possible. The data was returned without any code page conversion.
- 4**
One or more characters could not be converted. The character has been replaced by a blank in the converted data.
- 5**
There was an internal error in the code page conversion of a container.

110 CONTAINERERR

RESP2 values:

- 10**
The container named on the CONTAINER option could not be found.

16 INVREQ

RESP2 values:

- 2**
The INTOCCSID or INTOCODEPAGE option was specified without the CHANNEL option, and there is no current channel (because the program that issued the command was not passed one). INTOCCSID or INTOCODEPAGE is valid only on GET CONTAINER commands that specify (explicitly or implicitly) a channel.
- 4**
The CHANNEL option was not specified, there is no current channel (because the program that issued the command was not passed one).

5

The CONVERTST cvda value is invalid.

22 LENGERR

RESP2 values:

11

The length of the program area is shorter than the length of the data in the container. When the area is smaller, the data is truncated to fit into it.

12

The offset is greater than, or equal to, the length of the container.

GETNEXT CONTAINER

Browse the containers associated with a channel. You must specify the system programmer (SP) parameter in the EXEC statement of your compile job's translate step.

GETNEXT CONTAINER

► GETNEXT — CONTAINER(*data-area*) — BROWSETOKEN(*data-value*) ◄

Conditions: END, TOKENERR

Description

GETNEXT CONTAINER returns the name of the next container associated with a channel.

Note:

1. You can use successive GETNEXT CONTAINER commands to retrieve the names of all the channel's containers that existed at the time the STARTBROWSE CONTAINER command was executed. However, the names of any containers that are deleted after the STARTBROWSE and before they have been returned by a GETNEXT are not returned.
2. The names of any containers that are created on (or moved to) this channel after the STARTBROWSE command is executed may or may not be returned.
3. The order in which containers are returned is undefined.

Options

BROWSETOKEN(*data-value*)

Specifies, as a fullword binary value, a browse token returned on a previous STARTBROWSE CONTAINER command.

CONTAINER(*data-area*)

Returns the 16-character name of the next data-container.

Conditions

83 END

RESP2 values:

2

There are no more containers associated with this channel.

112 TOKENERR

RESP2 values:

3

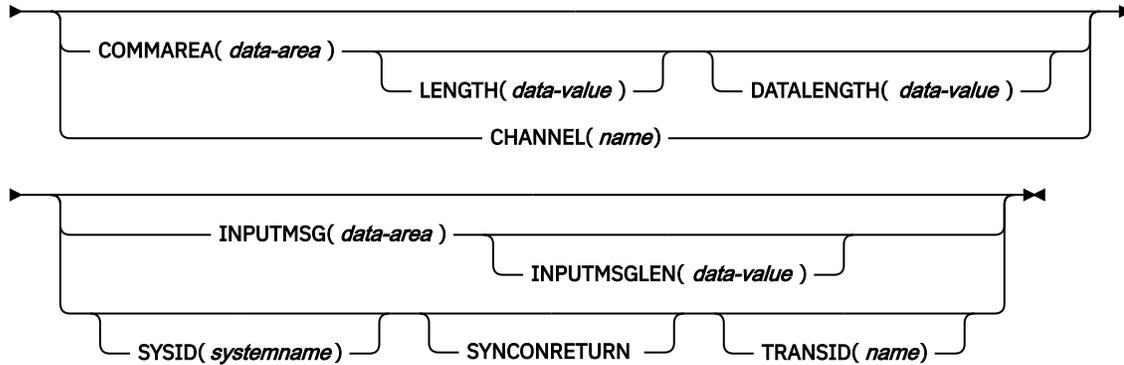
The browse token is not valid.

LINK

Link to another program expecting return. This topic describes the CHANNEL option and CHANNELERR condition only.

LINK

►► LINK — PROGRAM(*name*) ►►



Condition: CHANNELERR, INVREQ, LENGERR, NOTAUTH, PGMIDERR, ROLLEDBACK, SYSDIDERR, TERMERR

Description

LINK passes control from an application program at one logical level to an application program at the next lower logical level.

Option

CHANNEL(*name*)

Specifies the name (1 - 16 characters) of a channel that is to be made available to the called program. The acceptable characters are A - Z a - z 0 - 9 \$ @ # / % & ? ! : | " = ~ , ; < > . - and _ . Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

Channel names are always in EBCDIC. The allowable set of characters for channel names, listed above, includes some characters that do not have the same representation in all EBCDIC code pages. If containers are to be shipped between partitions, restrict the characters used in naming them to A - Z a - z 0 - 9 & : = , ; < > . - and _ .

The program that issues the LINK command might perform one or more of these actions:

- Have created the channel with one or more PUT CONTAINER CHANNEL commands.
- Specify its current channel, by name.
- Name a nonexistent channel, in which case a new, empty, channel is created.

Conditions

122 CHANNELERR

RESP2 values:

1

The name specified on the CHANNEL option contains an illegal character or combination of characters.

53 SYSDIDERR

RESP2 values related to the CHANNEL option:

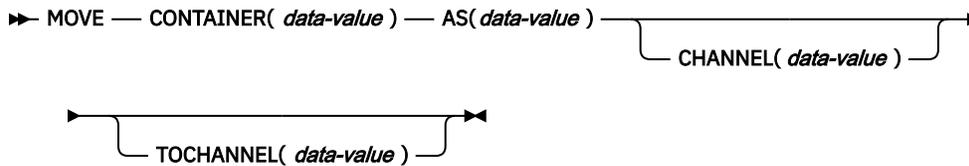
21

The CHANNEL option was used and the LINK request was shipped or routed to a remote system which does not support it (MRO connections only).

MOVE CONTAINER (CHANNEL)

Move a container (and its contents) from one channel to another.

MOVE CONTAINER (CHANNEL)



Conditions: CHANNELERR, CONTAINERERR, INVREQ

Description

MOVE CONTAINER (CHANNEL) moves a container from one channel to another. After the move, the source container no longer exists.

The source and target containers are identified by name and by the channels that own them. The channel that owns the source container can be identified:

- Explicitly, by specifying the CHANNEL option.
- Implicitly, by omitting the CHANNEL option. If this is omitted, the current channel is implied.

Similarly, the channel that owns the target container can be identified:

- Explicitly, by specifying the TOCHANNEL option.
- Implicitly, by omitting the TOCHANNEL option. If this is omitted, the current channel is implied.

You can move a container:

- From one channel to another.
- Within the same channel—for example, from the current channel to the current channel. This has the effect of renaming the container.

You can use MOVE CONTAINER, instead of GET CONTAINER and PUT CONTAINER, as a more efficient way of transferring data between channels.

Note:

1. The source channel must be within the scope of the program that issues the MOVE CONTAINER command.
2. If the target channel does not exist, within the scope of the program that issues the MOVE CONTAINER command, it is created.
3. If the source container does not exist, an error occurs.
4. If the target container does not already exist, it is created. If the target container already exists, its previous contents are overwritten.
5. If you try to overwrite a container with itself, nothing happens. That is, if you specify the same value for the CONTAINER and AS options, and either omit both the CHANNEL and TOCHANNEL options or give them the same value, so that the same channel is specified, the source container is not changed and not deleted. No error condition is raised.

Options

AS(*data-value*)

the name (1–16 characters) of the target container. If the target container already exists, its contents are overwritten.

The acceptable characters are A-Z a-z 0-9 \$ @ # / % & ? ! : | " = ~ , ; < > . - and _ . Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

Container names are always in EBCDIC. The allowable set of characters for container names, listed above, includes some characters that do not have the same representation in all EBCDIC code pages. We therefore recommend that, if containers are to be shipped between partitions, the characters used in naming them should be restricted to A-Z a-z 0-9 & : = , ; < > . - and _ .

CHANNEL(*data-value*)

Specifies the name (1–16 characters) of the channel that owns the source container. If this option is not specified, the current channel is implied.

CONTAINER(*data-value*)

Specifies the name (1–16 characters) of the source container that is to be moved.

TOCHANNEL(*data-value*)

Specifies the name (1–16 characters) of the channel that owns the target container. If you are specifying a new channel, remember that the acceptable characters are A-Z a-z 0-9 \$ @ # / % & ? ! : | " = ~ , ; < > . - and _ . Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

Channel names, listed above, includes some characters that do not have the same representation in all EBCDIC code pages. We therefore recommend that, if channels are to be shipped between partitions, the characters used in naming them should be restricted to A-Z a-z 0-9 & : = , ; < > . - and _ .

If this option is not specified, the current channel is implied.

Conditions

122 CHANNELERR

RESP2 values:

1

The name specified on the TOCHANNEL option contains an illegal character or combination of characters.

2

The channel specified on the CHANNEL option could not be found.

3

Either the current channel or the channel specified on the CHANNEL option is read-only.

110 CONTAINERERR

RESP2 values:

10

The container named on the CONTAINER option could not be found.

18

The name specified on the AS option contains an illegal character or combination of characters.

16 INVREQ

RESP2 values:

4

The CHANNEL or TOCHANNEL option (or both) was not specified, there is no current channel (because the program that issued the command was not passed a channel).

30

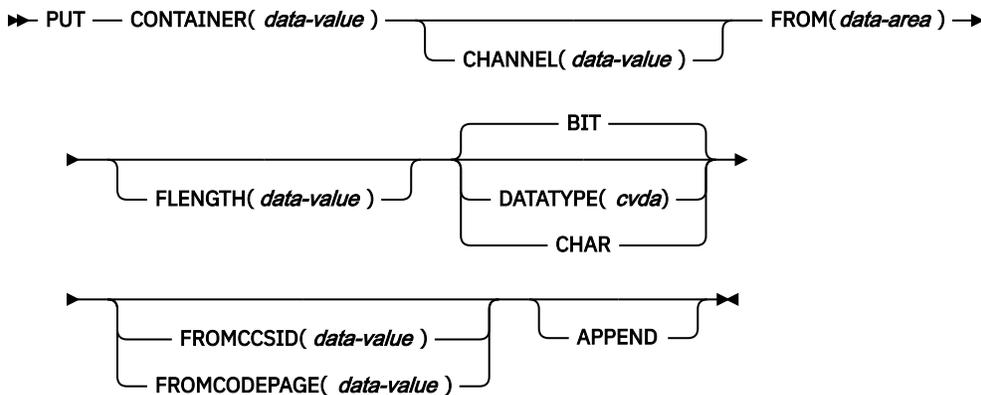
You cannot move a CICS-defined read-only container.

You cannot move a container to (that is, overwrite) an existing, CICS-defined, read-only container.

PUT CONTAINER (CHANNEL)

Place data in a named channel container.

PUT CONTAINER (CHANNEL)



Conditions: CCSIDERR, CHANNELERR, CODEPAGEERR, CONTAINERERR, INVREQ, LENGERR

Description

PUT CONTAINER (CHANNEL) places data in a container associated with a specified channel.

The container is identified by name. The channel that owns the container can be identified:

- Explicitly, by specifying the CHANNEL option.
- Implicitly, by omitting the CHANNEL option. If this is omitted, the current channel is implied.

Note:

1. There is no limit to the number of containers that can be associated with a channel.
2. The size of individual containers is limited only by the amount of storage available.

CAUTION: Take care not to create so many large containers that you limit the amount of storage available to other applications.

3. If the named container does not already exist, it is created. If the named container already exists, its previous contents is overwritten. For a CHAR container, the data is stored in the CCSID specified on the original PUT CONTAINER command that created the container.

Options

APPEND

Specifies that the data passed to the container is appended to the existing data in the container. If this option is not stated, the existing data in the container is overwritten by the data passed to the container.

CHANNEL(data-value)

Specifies the name (1–16 characters) of the channel that owns the container.

The acceptable characters are A-Z a-z 0-9 \$ @ # / % & ? ! : | " = ~ , ; < > . - and _ . Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

Channel names are always in EBCDIC. The allowable set of characters for channel names, listed above, includes some characters that do not have the same representation in all EBCDIC code pages.

We therefore recommend that, if channels are to be shipped between partitions, the characters used in naming them should be restricted to A-Z a-z 0-9 & : = , ; < > . - and _.

CONTAINER(*data-value*)

Specifies the name (1–16 characters) of the container into which data is to be placed.

The acceptable characters are A-Z a-z 0-9 \$ @ # / % & ? ! : | " = ~ , ; < > . - and _ . Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

Do not use container names beginning with "DFH", unless requested to do so by CICS.

Container names are always in EBCDIC. The allowable set of characters for container names, listed above, includes some characters that do not have the same representation in all EBCDIC code pages. We therefore recommend that, if containers are to be shipped between partitions, the characters used in naming them should be restricted to A-Z a-z 0-9 & : = , ; < > . - and _.

DATATYPE(*cvda*)

Specifies the type of data to be put into the container. This option applies only to *new* containers. If the container already exists, its data type was established when it was created and cannot be changed. CVDA values are:

BIT

Bit data. The data in the container cannot be converted. This is the default value, unless FROMCCSID or FROMCODEPAGE is specified.

CHAR

Character data. The data to be stored in the container is converted (if required) according to the setting in the FROMCCSID or FROMCODEPAGE value. If the FROMCCSID and FROMCODEPAGE option are not specified, it is assumed that the data is encoded in the CCSID of the partition, as specified in the LOCALCCSID system initialization parameter.

For CHAR containers, the data is stored in the Coded Character Set Identifier (CCSID) specified on the original PUT CONTAINER command that created the container. If the FROMCCSID and FROMCODEPAGE option were not specified on the original PUT CONTAINER command, the data is stored in the partition's default CCSID. The data on all future PUT CONTAINER CHANNEL commands for this container is converted into this same CCSID. If you want to avoid this, the application program should delete the existing container before issuing the new PUT CONAINTER command, thus recreating the container.

All the data in a container is converted as if it were a single character string. For SBCS code pages, a structure consisting of several character fields is equivalent to a single-byte character string. However, for DBCS code pages this is not the case. If you use DBCS code pages, to ensure that data conversion works correctly you must put each character string into a separate container.

FLENGTH(*data-value*)

Specifies, as a fullword binary value, the length of the data area from which data is to be read.

FLENGTH need not be specified if the length can be generated by the compiler from the FROM variable.

FROM(*data-area*)

Specifies the data area from which the data is written to the container.

FROMCCSID(*data-value*)

Specifies the current Coded Character Set Identifier (CCSID) of the character data to be put into the container, as a fullword binary number. Use this option if the data to be placed in the container is not encoded in the CCSID of the partition, as specified in the LOCALCCSID system initialization parameter.

If you prefer to specify an IANA name for the code page, or if you prefer to specify the CCSID as alphanumeric characters, use the FROMCODEPAGE option instead. All the data in a container is converted as if it were a single character string. For SBCS code pages, a structure consisting of several character fields is equivalent to a single-byte character string. However, for DBCS code

pages this is not the case. If you use DBCS code pages, to ensure that data conversion works correctly you must put each character string into a separate container.

If the FROMCCSID option is specified, DATATYPE(DFHVALUE(CHAR)) is implied.

For more information about data conversion with channels, refer to Chapter 34, “How to use channels and containers,” on page 213. For an explanation of CCSIDs, see “Data conversion” on page 231.

FROMCODEPAGE(*data-value*)

Specifies an IANA-registered alphanumeric charset name or a Coded Character Set Identifier (CCSID) for the current code page of the character data to be put into the container, using up to 40 alphanumeric characters, including appropriate punctuation. Use this option instead of the CCSID option if you prefer to use an IANA-registered charset name, as specified in the Content-Type header for an HTTP request. CICS converts the IANA name into a CCSID, and the subsequent data conversion process is identical. Also use this option if you prefer to specify the CCSID in alphanumeric characters, rather than as a fullword binary number.

If the FROMCODEPAGE option is specified, DATATYPE(DFHVALUE(CHAR)) is implied.

For more information about data conversion with channels, refer to Chapter 34, “How to use channels and containers,” on page 213. For an explanation of CCSIDs, see “Data conversion” on page 231.

Conditions

123 CCSIDERR

RESP2 values:

1

The CCSID specified on the FROMCCSID option is outside the range of valid CCSID values.

2

The CCSID specified on the FROMCCSID option and the CCSID of the container are an unsupported combination. The CCSID of the container is the value that was specified, or defaulted, on the first PUT CONTAINER command for this container. The first time each invalid combination is used, CICS issues error message DFHAP0802, which contains the pair of CCSIDs.

4

One or more characters could not be converted. Each unconverted character has been replaced by a blank in the converted data.

5

There was an internal error in the code page conversion of a container. This error can occur only when the target of the PUT is an existing, CICS-created, container.

122 CHANNELERR

RESP2 values:

1

The name specified on the CHANNEL option contains an illegal character or combination of characters.

125 CODEPAGEERR

RESP2 values:

1

The code page specified on the FROMCODEPAGE option is not supported.

2

The code page specified on the FROMCODEPAGE option and the CCSID of the container are an unsupported combination. The CCSID of the container is the value that was specified using either FROMCODEPAGE or FROMCCSID, or defaulted, on the first PUT CONTAINER command for this container. The first time each invalid combination is used, CICS issues error message DFHAP0802, which contains the pair of CCSIDs.

4

One or more characters could not be converted. Each unconverted character has been replaced by a blank in the converted data. This error can occur only when the target of the PUT is an existing container.

5

There was an internal error in the code page conversion of a container. This error can occur only when the target of the PUT is an existing, CICS-created, container.

110 CONTAINERERR

RESP2 values:

18

The name specified on the CONTAINER option contains an illegal character or combination of characters.

16 INVREQ

RESP2 values:

1

The DATATYPE option was specified without the CHANNEL option, and there is no current channel (because the program that issued the command was not passed one.) DATATYPE is valid only on PUT CONTAINER commands that specify (explicitly or implicitly) a channel.

2

The FROMCCSID or FROMCODEPAGE option was specified without the CHANNEL option, and there is no current channel (because the program that issued the command was not passed one.) FROMCCSID and FROMCODEPAGE is valid only on PUT CONTAINER commands that specify (explicitly or implicitly) a channel.

4

The CHANNEL option was not specified, there is no current channel (because the program that issued the command was not passed one).

30

You tried to write to a CICS-defined read only container.

32

A CVDA value other than CHAR or BIT was specified for DATATYPE.

33

An attempt was made to change the data-type of an existing container.

34

A data-type of BIT is invalid with a CCSID or CODEPAGE.

22 LENGERR

RESP2 values:

1

A negative number was specified on the FLENGTH option.

RETURN

Return program control. This topic describes the CHANNEL option, CHANNELERR condition, and INVREQ condition only.

These exposures result from the delay between running the START command and the time of task creation. Even on a START CHANNEL request, when the START is always immediate, CICS can delay creating the task, either because the required terminal is not free or because of other system constraints.

You can use INQUIRE commands to ensure that the transaction and program are enabled at the time of the START command, but either can become unavailable before task creation.

Options

TRANSID(*name*)

Specifies the symbolic identifier (1–4 characters) of the transaction to be run by a task started as the result of a START command. If SYSID is specified, and names a remote system, the transaction is assumed to be on that system irrespective of whether the transaction is defined as remote. Otherwise, the transaction definition is used to find out whether the transaction is on a local or a remote system.

CHANNEL(*data-value*)

Specifies the name (1–16 characters) of a channel that is to be made available to the started task.

The acceptable characters are A-Z a-z 0-9 \$ @ # / % & ? ! : | " = ~ , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

Channel names are always in EBCDIC. The allowable set of characters for channel names, listed above, includes some characters that do not have the same representation in all EBCDIC code pages. We therefore recommend that, if channels are to be shipped between partitions, the characters used in naming them should be restricted to A-Z a-z 0-9 & : = , ; < > . - and _.

The program that issues the START command can perform these tasks:

- Have created the channel with one or more PUT CONTAINER CHANNEL commands.
- Specify its current channel, by name.
- Name a non-existent channel, in which case a new, empty, channel is created.

The started task is given a copy of the channel's containers (and the data they contain). The copy is made when the START command is issued.

SYSID(*systemname*)

Specifies the name of the system to which the request is directed.

TERMID(*name*)

Specifies the symbolic identifier (1 - 4 alphanumeric characters) of the principal facility associated with a transaction to be started as a result of a START command. This principal facility can be a terminal (the usual case) or an APPC session. Where an APPC session is specified, the connection (or modeset) name is used instead of a terminal identifier. This option is required when the transaction to be started must communicate with a terminal; it should be omitted otherwise.

You must define the terminal identifier as either a local or a remote terminal on the system in which the START command is run.

The TERMID option is used in previous hop data that is collected. For more information about using the TERMID option with previous hop data, see "Passing previous hop data with origin data using IPIC" in the [CICS Intercommunication Guide](#), SC33-1665.

USERID(*data-value*)

Specifies the user ID under whose authority the started transaction is to run, if the started transaction is not associated with a terminal (that is, when TERMID is not specified). This user ID is referred to as *userid1*.

If you omit both TERMID and USERID, CICS uses instead the user ID under which the transaction that issues the START command is running. This user ID is referred to as *userid2*.

By using either *userid1* or *userid2* CICS ensures that a started transaction always runs under a valid user ID, which must be authorized to all the resources referenced by the started transaction.

CICS performs a surrogate security check against *userid2* to verify that this user is authorized to *userid1*. If *userid2* is not authorized, CICS returns a NOTAUTH condition. The surrogate check is not done here if USERID is omitted.

Conditions

122 CHANNELERR

RESP2 values:

1

The channel specified on the CHANNEL option contains an incorrect character or combination of characters.

16 INVREQ

RESP2 values:

9

The options specified on the command are incompatible.

17

The STARTed transaction is not shutdown-enabled, and the CICS partition is in the process of shutting down.

18

A USERID is specified and the CICS external security manager interface is not initialized.

INVREQ also occurs (RESP2 not set) if the START command is not valid for processing by CICS.

Default action: end the task abnormally.

54 ISCINVREQ

Occurs when the remote system indicates a failure that does not correspond to a known condition.

Default action: end the task abnormally.

70 NOTAUTH

RESP2 values:

7

A resource security check fails on TRANSID (name).

9

A surrogate user security check fails on USERID (name).

The security access capabilities of the transaction that issued the command do not allow the command to be performed with the value specified in the USERID option. The security access capabilities of the transaction have been established by the external security manager according to user security, and whether link security or the execution diagnostic facility (EDF) have been in use.

Default action: end the task abnormally.

53 SYSIDERR

Occurs in all of the following cases:

- The SYSID option specifies a name that is neither the local system nor a remote system (made known to CICS by defining a CONNECTION).
- The link to the remote system is known but unavailable.

In all the previous cases, the nature of the error is indicated by the second byte of the EIBRCODE.

The following errors are indicated by RESP2 values:

2

The CHANNEL option was used and the START request was shipped or routed to a remote system which does not support it. (MRO connections only).

Default action: end the task abnormally.

20

The CHANNEL option is specified and the START request is to be shipped over an LUTYPE61 connection. START CHANNEL requests cannot be shipped over LUTYPE61 connections.

11 TERMIDERR

Occurs if the terminal identifier in a START command is not defined to CICS.

Default action: end the task abnormally.

28 TRANSIDERR

Occurs if the transaction identifier specified in a START command is not defined to CICS.

Default action: end the task abnormally.

69 USERIDERR

RESP2 values:

8

The specified USERID is not known to the external security manager.

10

The external security manager is in a state such that CICS cannot determine whether a specified USERID is valid.

Default action: end the task abnormally.

STARTBROWSE CONTAINER

Start a browse of the containers associated with a channel. You must specify the system programmer (SP) parameter in the EXEC statement of your compile job's translate step.

STARTBROWSE CONTAINER

➔ STARTBROWSE — CONTAINER ————— BROWSETOKEN(*data-area*) ➔
 └── CHANNEL(*data-value*) ─┘

Conditions: ACTIVITYERR, CHANNELERR

Description

STARTBROWSE CONTAINER initializes a browse token which can be used to identify the name of each data-container associated with a specified channel.

Note: The browse token should be used only by the program that issues the STARTBROWSE command.

- If you specify the CHANNEL option, its containers are browsed.
- If the CHANNEL option is not specified and a current channel exists, its containers are browsed.
- Otherwise, ACTIVITYERR 2 is raised: see the description of the "ACTIVITYERR" condition, below.

Options

BROWSETOKEN(*data-area*)

Specifies a fullword binary data area, into which CICS will place the browse token.

CHANNEL(*data-value*)

Specifies the name (1–16 characters) of the channel whose containers are to be browsed. This must be the name of either the current channel or of a channel created by the program that issues the STARTBROWSE CONTAINER command.

If this option is not specified and a current channel exists, the current channel's containers are browsed.

The order in which containers are returned is undefined.

Condition

122 CHANNELERR

RESP2 values:

1

The name specified on the CHANNEL option contains an illegal character or combination of characters.

EXEC Interface Block

This section contains a description of the EXEC interface block (EIB) fields that were changed for channels and containers.

EIBFN

Contains a code that identifies the last CICS command issued by the task.

Code	Command
3412	DELETE CONTAINER
3414	GET CONTAINER
3416	PUT CONTAINER
3440	MOVE CONTAINER

EIBRCODE

Contains the CICS response code returned after the function requested by the last CICS command to be issued by the task has been completed.

EIBFN	EIBRCODE	Condition
0E ..	DA	CHANNELERR
10 ..	DA	CHANNELERR

EIBRESP

Contains a number corresponding to the RESP condition that occurred. These numbers are listed below (in decimal) for the conditions that can occur during execution of the commands described in this publication.

No.	Condition
109	ACTIVITYERR
110	CONTAINERERR
112	TOKENERR
.	
.	
122	CHANNELERR
123	CCSIDERR
125	CODEPAGEERR

Chapter 34. How to use channels and containers

Channels are a method of transferring data between programs, in amounts that far exceed the 32KB limit that applies to COMMAREAs.

This section contains:

- [“Channels: quick start” on page 213](#)
- [“Using channels: some typical scenarios” on page 216](#)
- [“Creating a channel” on page 218](#)
- [“The current channel” on page 219](#)
- [“The scope of a channel” on page 224](#)
- [“Discovering which containers were passed to a program” on page 228](#)
- [“Discovering which containers were returned from a link” on page 228](#)
- [“CICS read-only containers” on page 228](#)
- [“Designing a channel: Best practices” on page 229](#)
- [“Constructing and using a channel: an example” on page 230](#)
- [“Dynamic transaction routing program with channels” on page 231](#)
- [“Data conversion” on page 231](#)
- [“Benefits of channels” on page 236](#)
- [“Migrating from COMMAREAs to channels” on page 236](#)

Channels: quick start

A brief introduction to channels and containers.

Channels and containers

Containers are named blocks of data, designed for passing information between programs. You can think of them as of “named communication areas (COMMAREAs)”. Programs can pass any number of containers between each other. Containers are grouped in sets that are called *channels*. A channel is analogous to a parameter list.

Each container is defined in 31-bit CICS storage, and is not directly addressable by any application program. Instead, the channels and containers API must be used to transfer the current container content to and from other areas of storage in order to both access and replace the data.

To create named containers and assign them to a channel, a program uses **EXEC CICS PUT CONTAINER(container-name) CHANNEL(channel-name)** commands. It can then pass the channel and its containers to a second program by using the **CHANNEL(channel-name)** option of the **EXEC CICS LINK, XCTL, START, or RETURN** commands.

The second program can read containers that are passed to it using the **EXEC CICS GET CONTAINER(container-name)** command. This command reads the named container that belongs to the channel that the program was invoked with.

If the second program is called by a **LINK** command, it can also return containers to the calling program. It can do this by creating new containers, or by reusing existing containers with the help of **EXEC CICS PUT CONTAINER(container-name)** command.

On return, the calling program must use **EXEC CICS GET CONTAINER(container-name)** to see the content of all containers that have been updated by the **LINKed** program or any lower-level **LINKed** programs.

The use of **EXEC CICS GET CONTAINER(container-name) SET(ptr-ref)** must be handled very carefully. This is because the address in **ptr-ref** is not guaranteed to still be valid when lower level programs are then

invoked that use the container and then return. CICS maintains only one SET buffer per container, and certain conditions cause CICS to FREEMAIN the buffer and GETMAIN a new one. Using an out-of-date SET buffer will cause unpredictable results.

Channels and containers are visible only to the program that creates them and the programs they are passed to. When these programs end, CICS automatically destroys the containers and their storage.

Channel containers are not recoverable. Pseudoconversational transactions that are started with RETURN TRANSID CHANNEL() cannot be restarted.

Basic examples

A simple example of a program that creates a channel and passes it to second program.

Figure 26 on page 215 shows a COBOL program, CLIENT1, that:

1. Uses PUT CONTAINER(*container-name*) CHANNEL(*channel-name*) commands to create a channel called `inqcustrec` and add two containers, `custno` and `branchno`, to it; these contain a customer number and a branch number, respectively.
2. Uses a LINK PROGRAM(*program-name*) CHANNEL(*channel-name*) command to link to program SERVER1, passing the `inqcustrec` channel.
3. Uses a GET CONTAINER(*container-name*) CHANNEL(*channel-name*) command to retrieve the customer record returned by SERVER1. The customer record is in the `custrec` container of the `inqcustrec` channel.

Note that the same COBOL copybook, INQINTC, is used by both the client and server programs. Line 3 and lines 5 through 7 of the copybook represent the INQUIRY-CHANNEL and its containers. These lines are not strictly necessary to the working of the programs, because channels and containers are created by being named on, for example, **PUT CONTAINER** commands; they do not have to be defined. However, the inclusion of these lines in the copybook used by both programs makes for easier maintenance; they record the names of the containers used.

Recommendation

For ease of maintenance of a client/server application that uses a channel, create a copybook that records the names of the containers used and defines the data fields that map to the containers. Include the copybook in both the client and the server program.

Note: This example shows two COBOL programs. The same techniques can be used in any of the other languages supported by CICS. However, *for COBOL programs only*, if the server program uses the SET option (instead of INTO) on the **EXEC CICS GET CONTAINER** command, the structure of the storage pointed to by SET must be defined in the LINKAGE section of the program. This means that you will require two copybooks rather than one. The first, in the WORKING-STORAGE section of the program, names the channel and containers used. The second, in the LINKAGE section, defines the storage structure.

```

IDENTIFICATION DIVISION.
PROGRAM-ID. CLIENT1.

WORKING-STORAGE SECTION.

    COPY INQINTC
*       copybook INQINTC
* Channel name
* 01 INQUIRY-CHANNEL PIC X(16) VALUE 'inqcustrec'.
* Container names
* 01 CUSTOMER-NO      PIC X(16) VALUE 'custno'.
* 01 BRANCH-NO       PIC X(16) VALUE 'branchno'.
* 01 CUSTOMER-RECORD PIC X(16) VALUE 'custrec'.
* Define the data fields used by the program
* 01 CUSTNO          PIC X(8).
* 01 BRANCHNO       PIC X(5).
* 01 CREC.
*   02 CUSTNAME     PIC X(80).
*   02 CUSTADDR1    PIC X(80).
*   02 CUSTADDR2    PIC X(80).
*   02 CUSTADDR3    PIC X(80).

PROCEDURE DIVISION.
MAIN-PROCESSING SECTION.

*
* INITIALISE CUSTOMER RECORD
*
*   ... CREATE CUSTNO and BRANCHNO
*
* GET CUSTOMER RECORD
*
*   EXEC CICS PUT CONTAINER(CUSTOMER-NO) CHANNEL(INQUIRY-CHANNEL)
*                 FROM(CUSTNO) FLENGTH(LENGTH OF CUSTNO)
*                 END-EXEC
*   EXEC CICS PUT CONTAINER(BRANCH-NO) CHANNEL(INQUIRY-CHANNEL)
*                 FROM(BRANCHNO) FLENGTH(LENGTH OF BRANCHNO)
*                 END-EXEC
*
*   EXEC CICS LINK PROGRAM('SERVER1') CHANNEL(INQUIRY-CHANNEL) END-EXEC
*
*   EXEC CICS GET CONTAINER(CUSTOMER-RECORD) CHANNEL(INQUIRY-CHANNEL)
*                 INTO(CREC) END-EXEC
*
* PROCESS CUSTOMER RECORD
*
*   ... FURTHER PROCESSING USING CUSTNAME and CUSTADDR1 etc...
*
*   EXEC CICS RETURN END-EXEC
*
*   EXIT.

```

Figure 26. A simple example of a program that creates a channel and passes it to a second program

Figure 27 on page 216 shows the SERVER1 program linked to by CLIENT1. SERVER1 retrieves the data from the custno and branchno containers it has been passed, and uses it to locate the full customer record in its database. It then creates a new container, custrec, on the same channel, and returns the customer record in it.

Note that the programmer hasn't specified the CHANNEL keyword on the GET and PUT commands in SERVER1: if the channel isn't specified explicitly, the current channel is used—that is, the channel that the program was invoked with.

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. SERVER1.  
  
WORKING-STORAGE SECTION.  
  
    COPY INQINTC  
    *           copybook INQINTC  
* Channel name  
* 01 INQUIRY-CHANNEL PIC X(16) VALUE 'inqcustrec'.  
* Container names  
* 01 CUSTOMER-NO     PIC X(16) VALUE 'custno'.  
* 01 BRANCH-NO       PIC X(16) VALUE 'branchno'.  
* 01 CUSTOMER-RECORD PIC X(16) VALUE 'custrec'.  
* Define the data fields used by the program  
* 01 CUSTNO          PIC X(8).  
* 01 BRANCHNO       PIC X(5).  
* 01 CREC.  
*   02 CUSTNAME      PIC X(80).  
*   02 CUSTADDR1     PIC X(80).  
*   02 CUSTADDR2     PIC X(80).  
*   02 CUSTADDR3     PIC X(80).  
  
PROCEDURE DIVISION.  
MAIN-PROCESSING SECTION.  
  
    EXEC CICS GET CONTAINER(CUSTOMER-NO)  
                INTO(CUSTNO) END-EXEC  
    EXEC CICS GET CONTAINER(BRANCH-NO)  
                INTO(BRANCHNO) END-EXEC  
  
    ... USE CUSTNO AND BRANCHNO TO FIND CREC IN A DATABASE  
  
    EXEC CICS PUT CONTAINER(CUSTOMER-RECORD)  
                FROM(CREC)  
                FLENGTH(LENGTH OF CREC) END-EXEC  
  
    EXEC CICS RETURN END-EXEC  
  
    EXIT.
```

Figure 27. A simple example of a linked to program that retrieves data from the channel it has been passed

Using channels: some typical scenarios

Channels and containers provide a powerful way to pass data between programs. These scenarios show some examples of how channels can be used.

One channel, one program

This example shows a stand-alone program with a single channel.

[Figure 28 on page 217](#) shows the simplest scenario—a "stand-alone" program with a single channel with which it can be invoked.

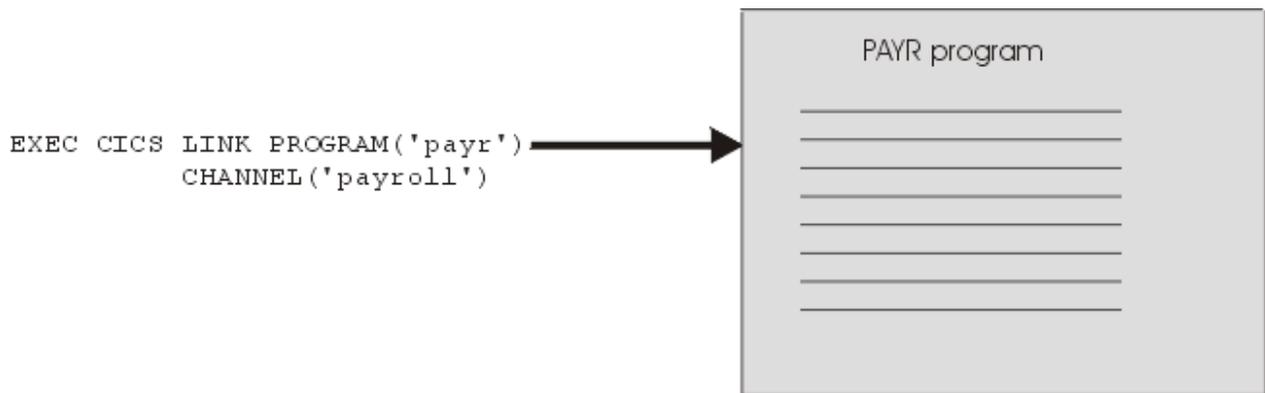


Figure 28. A stand-alone program with a single channel

One channel, several programs (a component)

This example shows a set of related programs (a component) invoked through a single channel.

In Figure 29 on page 217, there is a single channel to the top-level program in a set of inter-related programs. The set of programs within the shaded area can be regarded as a "component". The client program "sees" only the external channel and has no knowledge of the processing that takes place nor of the existence of the back-end programs.

Inside the component, the programs can pass the channel between themselves. Alternatively, a component program could, for example, pass a subset of the original channel, by creating a new channel and adding one or more containers from the original channel.

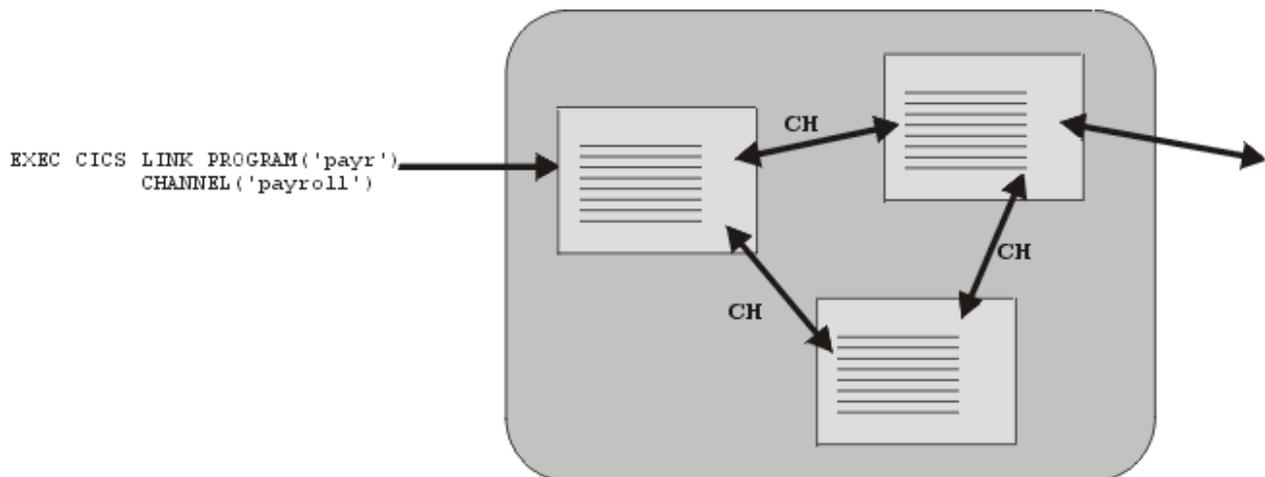


Figure 29. A "component"—a set of related programs invoked through a single external channel

Several channels, one component

This example shows a set of related programs (a component) which can be invoked through two alternative channels.

As in the previous example, we have a set of inter-related programs that can be regarded as a component. However, this time there are two, alternative, external channels with which the component can be invoked. The top-level program in the component issues an EXEC CICS ASSIGN CHANNEL command to determine which channel it has been invoked with, and tailors its processing accordingly.

The "loose coupling" between the client program and the component permits easy evolution. That is, the client and the component can be upgraded at different times. For example, first the component could be upgraded to handle a third channel, consisting of a different set of containers from the first, or second channels. Next, the client program could be upgraded (or a new client written) to pass the third channel.

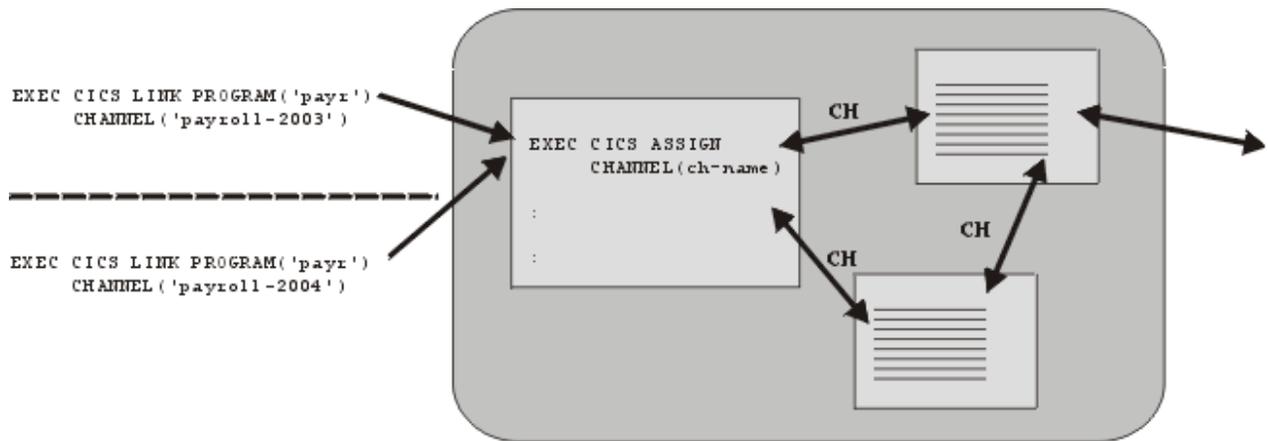


Figure 30. Multiple external channels to the same component

Multiple interactive components

This example shows how multiple components can interact through their channels.

Figure 31 on page 218 shows a "Human resources" component and a "Payroll" component, each with a channel with which it can be invoked. The Payroll component is invoked from both a stand-alone program and the Human resources component.

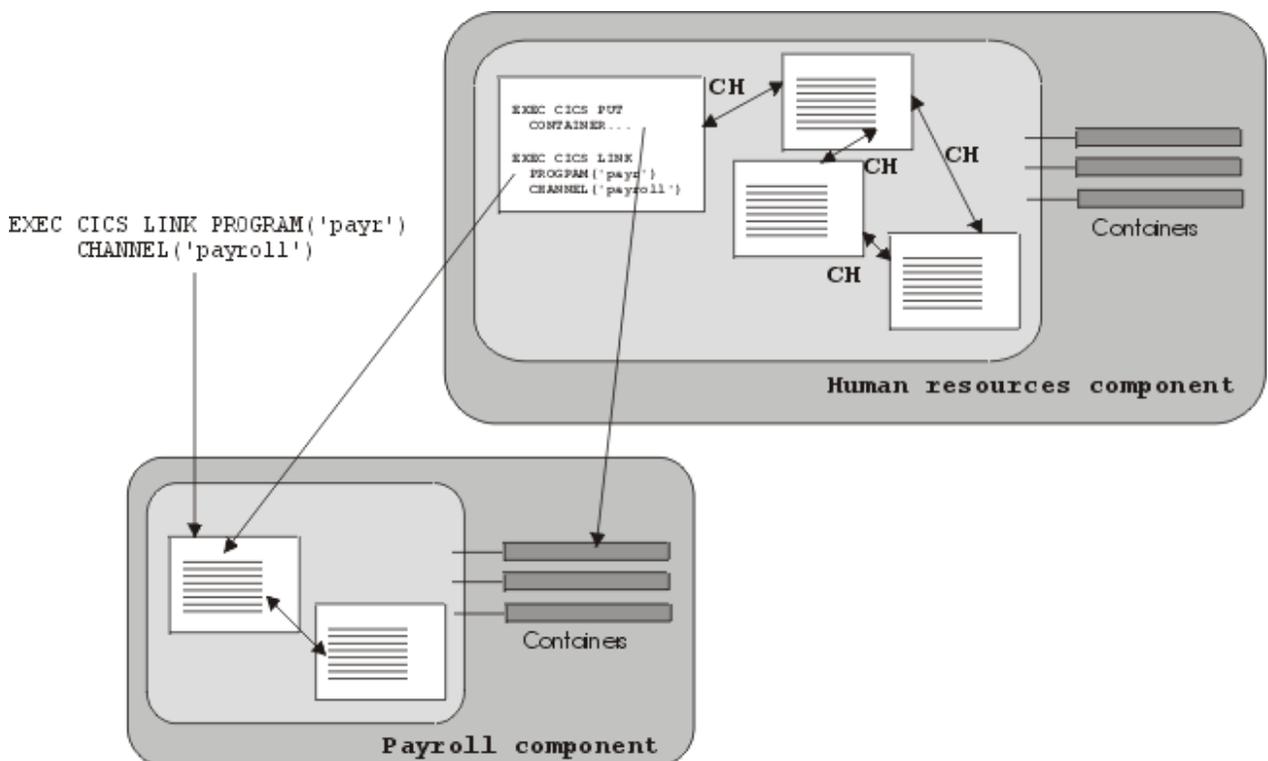


Figure 31. Multiple components which interact through their channels

Creating a channel

You can create a channel by naming it on one of a number of API commands. If the channel does not exist in the current program scope, CICS creates it.

About this task

You create a channel by naming it on one of the following commands:

LINK PROGRAM CHANNEL
MOVE CONTAINER CHANNEL TOCHANNEL
PUT CONTAINER CHANNEL
RETURN TRANSID CHANNEL
START TRANSID CHANNEL
XCTL PROGRAM CHANNEL

The most straightforward way to create a channel and populate it with containers of data is to issue a succession of **EXEC CICS PUT CONTAINER(container-name) CHANNEL(channel-name) FROM(data_area)** commands. The first PUT command creates the channel (if it does not exist), and adds a container to it; the subsequent commands add further containers to the channel. If the containers exist, their contents are overwritten by the new data.

Note: New channels remain in scope until the link level changes. For more information about channel scope, see [“The scope of a channel” on page 224](#).

Do not give a channel a name that starts with the characters *DFH*, apart from those that CICS defines as part of its API. These channels must only be used as described by that API.

An alternative way to add containers to a channel is to move them from another channel. To do this, use the following command:

```
EXEC CICS MOVE CONTAINER(container-name) AS(container-new-name)  
CHANNEL(channel-name1) TOCHANNEL(channel-name2)
```

Note:

1. If the CHANNEL or TOCHANNEL option is not specified, the current channel is implied.
2. The source channel must be in program scope.
3. If the target channel does not exist in the current program scope, it is created.
4. If the source container does not exist, an error occurs.
5. If the target container does not exist, it is created; if the target container exists, its contents are overwritten.
6. When a channel is created, it exists until the task that created it terminates. For example, if a long running task performs many PUT CONTAINER commands to different unique channels, all the channels that are created do not release the main storage acquired by the GETMAIN command until the task ends. You must design your applications accordingly to prevent your regions becoming short on storage.

You can use MOVE CONTAINER, instead of GET CONTAINER and PUT CONTAINER, as a more efficient way of transferring data between channels.

If the channel named on the following commands does not exist in the current program scope, an *empty* channel is created:

- EXEC CICS LINK PROGRAM CHANNEL(channel-name)
- EXEC CICS RETURN TRANSID CHANNEL(channel-name)
- EXEC CICS START TRANSID CHANNEL(channel-name)
- EXEC CICS XCTL PROGRAM CHANNEL(channel-name)

The current channel

A program's *current channel* is the channel - if there is one - with which it was invoked. These examples show how the current channel and its containers are passed between programs.

The program can create other channels. However, the current channel, for a particular invocation of a particular program, does not change. It is analogous to a parameter list.

Current channel example, with LINK commands

This example shows how a program passes the current channel and its containers to another program using the EXEC CICS LINK command.

The following figure illustrates the origin of a program's current channel. It shows five interactive programs. Program A is a top-level program started by, for example, a terminal end user. It is not started by a program and does not have a current channel.

B, C, D, and E are second-, third-, fourth-, and fifth-level programs.

Program B's current channel is X, passed by the CHANNEL option on the EXEC CICS LINK command issued by program A. Program B modifies channel X by adding one container and deleting another.

Program C's current channel is also X, passed by the CHANNEL option on the EXEC CICS LINK command issued by program B.

Program D has no current channel, because C does not pass it one.

Program E's current channel is Y, passed by the CHANNEL option on the EXEC CICS LINK command issued by D.

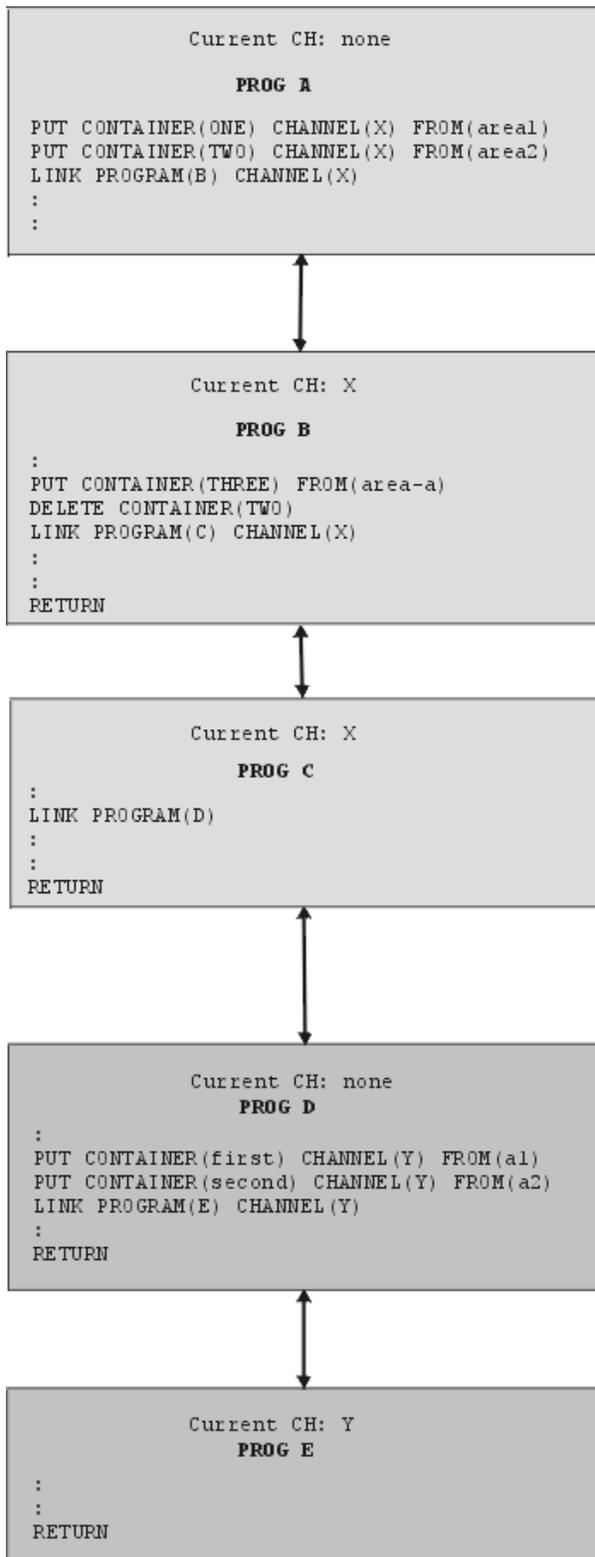


Figure 32. Current channel: example with LINK commands

The following table lists the name of the current channel (if any) of each of the five programs shown in the previous figure.

Table 12. The current channels of interactive programs—example with LINK commands

Prog.	Current CH	Issues commands	Comments
A	None	<pre> EXEC CICS PUT CONTAINER(ONE) CHANNEL(X) FROM(area1) EXEC CICS PUT CONTAINER(TWO) CHANNEL(X) FROM(area2) EXEC CICS LINK PROGRAM(B) CHANNEL(X) . </pre>	<p>Program A creates channel X and passes it to program B.</p> <p>Note that, by the time control is returned to program A by program B, the X channel has been modified—it does not contain the same set of containers as when it was created by program A. (Container TWO has been deleted and container THREE added by program B.)</p>
B	X	<pre> EXEC CICS PUT CONTAINER(THREE) FROM(area-a) EXEC CICS DELETE CONTAINER(TWO) EXEC CICS LINK PROGRAM(C) CHANNEL(X) . EXEC CICS RETURN </pre>	<p>Program B modifies channel X (its current channel) by adding and deleting containers, and passes the modified channel to program C.</p> <p>Program B does not need to specify the CHANNEL option on the PUT CONTAINER and DELETE CONTAINER commands; its current channel is implied.</p>
C	X	<pre> EXEC CICS LINK PROGRAM(D) . EXEC CICS RETURN </pre>	<p>Program C links to program D, but does not pass it a channel.</p>
D	None	<pre> EXEC CICS PUT CONTAINER(first) CHANNEL(Y) FROM(a1) EXEC CICS PUT CONTAINER(second) CHANNEL(Y) FROM(a2) EXEC CICS LINK PROGRAM(E) CHANNEL(Y) . EXEC CICS RETURN </pre>	<p>Program D creates a new channel, Y, which it passes to program E.</p>
E	Y	<pre> RETURN . </pre>	<p>Program E performs some processing on the data it's been passed and returns.</p>

Current channel example, with XCTL commands

This example shows how a program passes the current channel and its containers to another program using the EXEC CICS XCTL command.

Figure 33 on page 223 shows four interactive programs. A1 is a top-level program started by, for example, a terminal end user. It is not started by a program and does not have a current channel. B1, B2, and B3 are all second-level programs.

B1's current channel is X, passed by the CHANNEL option on the EXEC CICS LINK command issued by A1.

B2 has no current channel, because B1 does not pass it one.

B3's current channel is Y, passed by the CHANNEL option on the EXEC CICS XCTL command issued by B2.

When B3 returns, channel Y and its containers are deleted by CICS.

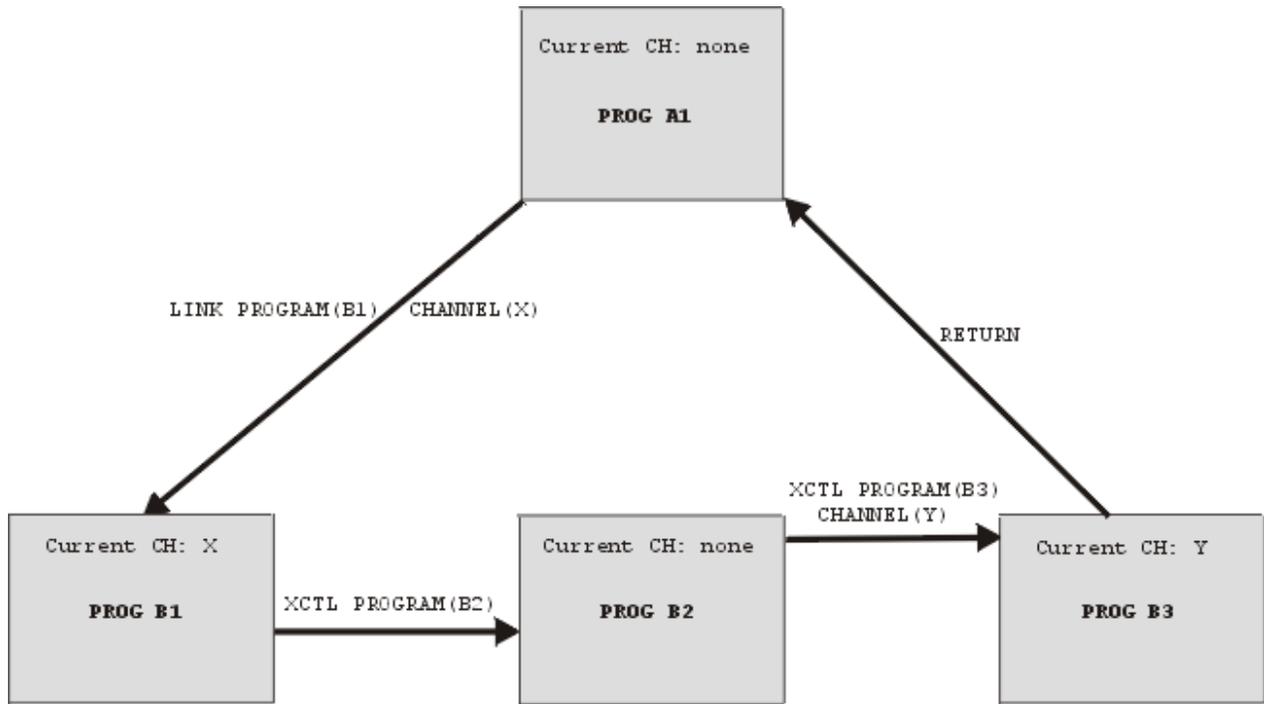


Figure 33. Current channels—example, with XCTL commands

The following table lists the name of the current channel (if any) of each of the four programs shown in Figure 33 on page 223.

Table 13. The current channels of interactive programs—example

Program	Current [®] channel	Issues command
A1	None	. EXEC CICS LINK PROGRAM(B1) CHANNEL(X) .
B1	X	. EXEC CICS XCTL PROGRAM(B2) .
B2	None	. EXEC CICS XCTL PROGRAM(B3) CHANNEL(Y) .
B3	Y	. EXEC CICS RETURN .

Current channel: START and RETURN commands

As well as the **LINK** and **XCTL** commands, you can pass channels on the **START** and **RETURN** commands.

EXEC CICS START TRANSID(*tranid*) CHANNEL(*channel-name*)

The program that implements the started transaction (or the first program, if there are more than one) is passed the channel, which becomes its current channel.

Note: All EXEC CICS START requests that specify a channel cannot be deferred by specifying INTERVAL, AT, FOR or UNTIL as this is not supported. This is monitored by the CICS translator. Using these commands to defer an EXEC CICS START request can result in a translator failure.

EXEC CICS RETURN TRANSID(*tranid*) CHANNEL(*channel-name*)

The CHANNEL option is valid only:

- On pseudoconversational RETURNS—that is, on RETURN commands that specify, on the TRANSID option, the next transaction to be run at the user terminal.
- If issued by a program at the highest logical level—that is, a program that returns control to CICS.

The program that implements the next transaction is passed the channel, which becomes its current channel.

The scope of a channel

The scope of a channel is the code from which it can be accessed. These examples show the scope of each channel in the diagram.

Scope example, with LINK commands

This example adds to the diagram from "Current channel example, with LINK commands" to show the scope of each channel.

The following figure shows the same five interactive programs previously described.

The scope of the X channel—the code from which it can be accessed—is programs A, B, and C.

The scope of the Y channel is programs D and E.

Note that, by the time control is returned to program A by program B, the X channel has been modified—it does not contain the same set of containers as when it was created by program A.

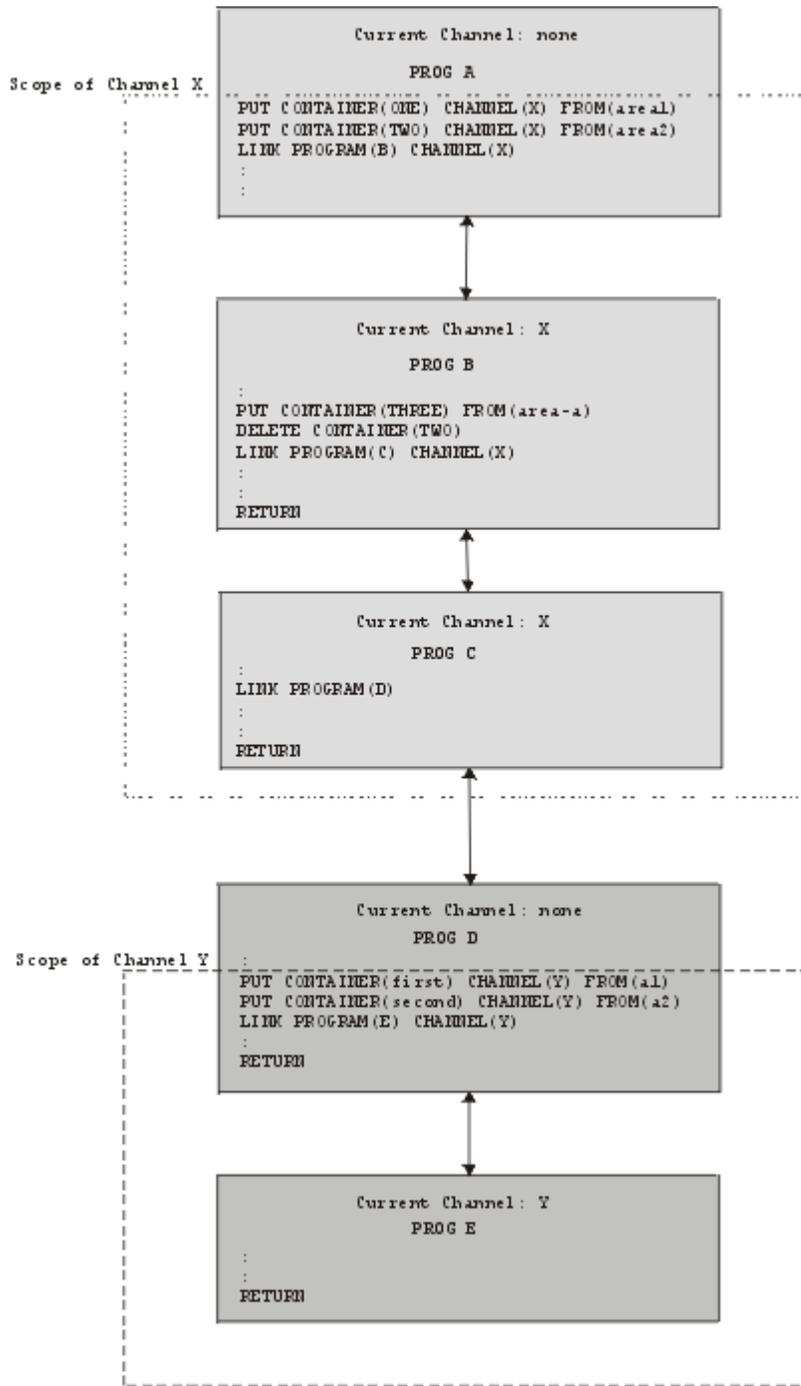


Figure 34. The scope of a channel—example showing LINK commands

The following table lists the name and scope of the current channel (if any) of each of the five programs shown in the previous figure.

Table 14. The scope of a channel—example with LINK commands

Program	Current channel	Scope of channel
A	None	Not applicable
B	X	A, B, C
C	X	A, B, C
D	None	Not applicable

Table 14. The scope of a channel—example with LINK commands (continued)

Program	Current channel	Scope of channel
E	Y	D, E

Scope example, with LINK and XCTL commands

This example adds to the diagram from "Current channel example, with XCTL commands" to show the scope of each channel.

Figure 35 on page 226 shows the same four interactive programs previously described, plus a third-level program, C1, that is invoked by an EXEC CICS LINK command from program B1.

The scope of the X channel is restricted to A1 and B1.

The scope of the Y channel is B2 and B3.

The scope of the Z channel is B1 and C1.

Note that, by the time control is returned to program A1 by program B3, it is possible that the X channel can have been modified by program B1, it might not contain the same set of containers as when it was created by A1.

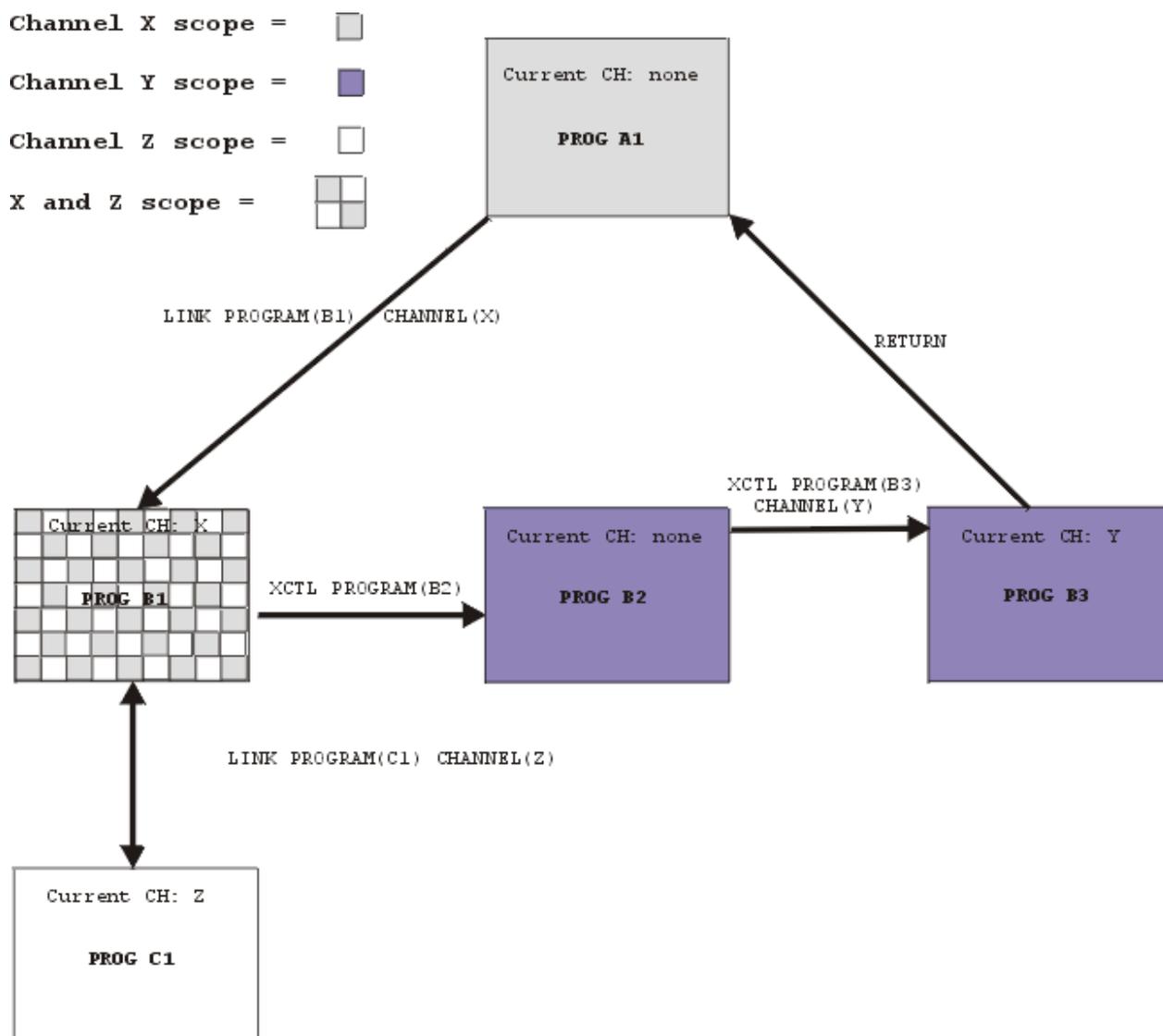


Figure 35. The scope of a channel—example showing LINK and XCTL commands

The following table lists the name and scope of the current channel (if any) of each of the five programs shown in Figure 35 on page 226.

<i>Table 15. The scope of a channel—example with LINK and XCTL commands</i>		
Program	Current channel	Scope of channel
A1	None	Not applicable
B1	X	A1 and B1
B2	None	Not applicable
B3	Y	B2 and B3
C1	Z	B1 and C1

Processing containers in a called sub-routine

When using either a dynamic or static COBOL call, because CICS treats the calling and the called program as being at the same logical link level, channels which are passed to or created by the calling program are available to the called program. However, you need to be aware of the importance of specifying a channel name on any container-related EXEC CICS command issued by the called program.

If the calling program has a current channel then, by default, this is the channel name assumed by the called program. Thus, if the called program issues a container-related EXEC CICS command with no channel name specified, the calling program's current channel is assumed. The calling program's current channel is also the called program's current channel.

If the calling program does not have a current channel, the called program too does not have a current channel. If this is the case, although any channels created by the calling program are available to the called program, any container-related EXEC CICS command issued by the called program to access one of these channels **must** specify the channel name. Failing to provide a channel name, when there is no default channel, causes a container-related EXEC CICS command to return an invalid request (INVREQ) exception with a RESP2 value of 4 ("no current channel").

If the calling program (and therefore the called program) has a current channel, if the called program needs to access any channel other than the default it must specify the other channel's name on the container-related EXEC CICS command.

When a called program wants to access any channel other than the default it must always specify the name of the target channel.

Passing a channel to another program or task

When passing channels between programs and tasks, it is important to understand how changes made to container data within those channels by the passed-to programs are reflected back (or not, in some cases) to the originating program.

LINK CHANNEL

You can pass a channel on a LINK command in the same way as you would pass a COMMAREA. If the linked program alters any data in the containers of the channel, the changes are reflected in the channel when it is returned to the calling program.

RETURN CHANNEL

In pseudo-conversational programming, you can pass a channel to the next transaction to be executed, in a similar way that a COMMAREA can be passed.

START CHANNEL

When starting a new task, a channel can be passed to the new program. The channel that is passed is copied, along with its containers, and placed into another channel with the same name. Any changes made to container data within this new copy of the channel are not reflected in the original channel in the original program.

XCTL CHANNEL

When you transfer control to another program, you can pass a channel to the new program. Because control is never given back to the calling program, any channels that it had access to go out of scope, and the storage is freed.

Discovering which containers were passed to a program

When a program is invoked, it can determine the names of the channel, and any containers that were passed to it.

Typically, programs that exchange a channel are written to handle that channel. That is, both client and server programs know the name of the channel, and the names and number of the containers in the channel. However, if, for example, a server program or component is written to handle more than one channel, on invocation it must discover which of the possible channels were passed to it.

A program can discover its current channel—that is, the channel with which it was invoked—by issuing an EXEC CICS ASSIGN CHANNEL command. (If there is no current channel, the command returns blanks.)

The program can also (should it need to) get the names of the containers in its current channel by browsing. Typically, this is not necessary. A program written to handle several channels is often coded to be aware of the names and number of the containers in each possible channel.

To get the names of the containers in the current channel, use the browse commands:

- EXEC CICS STARTBROWSE CONTAINER BROWSETOKEN(*data-area*) .
- EXEC CICS GETNEXT CONTAINER(*data-area*) BROWSETOKEN(*token*).
- EXEC CICS ENDBROWSE CONTAINER BROWSETOKEN(*token*).

Having retrieved the name of its current channel and, if necessary, the names of the containers in the channel, a server program can adjust its processing to suit the kind of data that was passed to it.

Discovering which containers were returned from a link

Following a LINK command, a program can discover the names of the containers returned by the program that was linked to.

A program creates a channel, which it passes to a second program by means of an EXEC CICS LINK PROGRAM(*program-name*) CHANNEL(*channel-name*) command. The second program performs some processing on the data that was passed to it, and returns the results in the same channel (its current channel).

On return, the first program knows the name of the channel which has been returned, but not necessarily the names of the containers in the channel. (Does the returned channel contain the same containers as the passed channel, or has the second program deleted some or created others?) The first program can discover the container-names by browsing. To do this, it uses the commands:

- EXEC CICS STARTBROWSE CONTAINER BROWSETOKEN(*data-area*) CHANNEL(*channel-name*).
- EXEC CICS GETNEXT CONTAINER(*data-area*) BROWSETOKEN(*token*).
- EXEC CICS ENDBROWSE CONTAINER BROWSETOKEN(*token*).

CICS read-only containers

CICS can create channels and containers for its own use, and pass them to user programs. In some cases CICS marks these containers as read-only, so that the user program cannot modify data that CICS needs on return from the user program.

User programs cannot create read-only containers.

You cannot overwrite, move, or delete a read-only container. Thus, if you specify a read-only container on a **PUT CONTAINER**, **MOVE CONTAINER**, or **DELETE CONTAINER** command, an INVREQ condition occurs.

Currently, CICS TS for z/VSE does not create read-only containers.

Channels and Security

CICS does not define any security mechanism to enforce who can use a channel name.

Designing a channel: Best practices

Containers are used to pass information between programs. These containers are grouped together in sets called channels. It is advisable to design your channels to follow a set of best practices.

Procedure

- At the end of a DPL call, input containers that the server program has not changed are not returned to the client. Input containers whose contents have been changed by the server program, and containers that the server program has created, are returned. Therefore, for optimal DPL performance, use the following best practices:
 - Use separate containers for input and output data.
 - Ensure that the server program, not the client, creates the output containers.
 - Use separate containers for read-only and read-write data.
 - If a structure is optional, make it a separate container.
 - Use dedicated containers for error information.
- The following general tips on designing a channel include, and expand on, the recommendations to achieve optimal DPL performance:
 - Use separate containers for input and output data. This provides the following benefits:
 - Better encapsulation of the data, making your programs easier to maintain.
 - Greater efficiency when a channel is passed on a DPL call, because smaller containers flow in each direction.
 - Ensure that the server program, not the client, creates the output containers. If the client creates them, empty containers are sent to the server region.
 - Use separate containers for read-only and read-write data. This provides the following benefits:
 - A simplification of your copybook structure, making your programs easier to understand.
 - Avoidance of the problems with REORDER overlays.
 - Greater transmission efficiency between CICS regions, because read-only containers sent to a server region will not be returned.
 - Use separate containers for each structure. This provides the following benefits:
 - Better encapsulation of the data, making your programs easier to understand and maintain.
 - Greater ease in changing one of the structures, because you do not need to recompile the entire component.
 - The ability to pass a subset of the channel to subcomponents, by using the **MOVE CONTAINER** command to move containers between channels.
 - If a structure is optional, make it a separate container. This leads to greater efficiency, because the structure is passed only if the container is present.
 - Use dedicated containers for error information. This provides the following benefits:
 - Easier identification of error information.
 - Greater efficiency, for the following reasons:
 - The structure containing the error information is passed back only if an error occurs.
 - It is more efficient to check for the presence of an error container by issuing a **GET CONTAINER** (*known-error-container-name*) command (and possibly receiving a NOTFOUND condition) than it is to initiate a browse of the containers in the channel.

- When you need to pass data of different types, for example, character data in codepage1 and character data in codepage2, use separate containers for each type, rather than one container with a complicated structure. This improves your ability to move between different code pages.
- When you need to pass a large amount of data, split it between multiple containers, rather than put it all into one container.

When a channel is passed to a remote program or transaction, passing a large amount of data might affect performance. This is particularly true if the local and remote regions are connected by an ISC, rather than MRO, connection.



Attention: Take care not to create so many large containers that you limit the amount of storage available to other applications.

Related concepts: Refer to [“Migrating from COMMAREAs to channels”](#) on page 236.

Related reference: Refer to [“Benefits of channels”](#) on page 236.

Constructing and using a channel: an example

In this example, a client program constructs a channel, passes it to a server program, and retrieves the server's output. The server program retrieves data from the channel's containers, and returns output to the client.

[Figure 36 on page 230](#) shows a CICS client program that:

1. Uses EXEC CICS PUT CONTAINER commands to construct (and put data in) a set of containers. The containers are all part of the same named channel—"payroll-2004".
2. Issues an EXEC CICS LINK command to invoke the PAYR server program, passing it the payroll-2004 channel.
3. Issues an EXEC CICS GET CONTAINER command to retrieve the server's program output, which it knows will be in the status container of the payroll-2004 channel.

```
* create the employee container on the payroll-2004 channel
EXEC CICS PUT CONTAINER('employee') CHANNEL('payroll-2004') FROM('John Doe')

* create the wage container on the payroll-2004 channel
EXEC CICS PUT CONTAINER('wage') CHANNEL('payroll-2004') FROM('100')

* invoke the payroll service, passing the payroll-2004 channel
EXEC CICS LINK PROGRAM('PAYR') CHANNEL('payroll-2004')

* examine the status returned on the payroll-2004 channel
EXEC CICS GET CONTAINER('status') CHANNEL('payroll-2004') INTO(stat)
```

Figure 36. How a client program can construct a channel, pass it to a server program, and retrieve the server's output

[Figure 37 on page 231](#) shows part of the PAYR server program invoked by the client. The server program:

1. Queries the channel with which it's been invoked.
2. Issues EXEC CICS GET CONTAINER commands to retrieve the input from the employee and wage containers of the payroll-2004 channel.
3. Processes the input data.
4. Issues an EXEC CICS PUT CONTAINER command to return its output in the status container of the payroll-2004 channel.

```

        "PAYR", CICS COBOL server program

* discover which channel I've been invoked with
EXEC CICS ASSIGN CHANNEL(ch_name)
:
WHEN ch_name 'payroll-2004'
  * my current channel is "payroll-2004"
  * get the employee passed into this program
  EXEC CICS GET CONTAINER('employee') INTO(emp)
  * get the wage for this employee
  EXEC CICS GET CONTAINER('wage') INTO(wge)
  :
  * process the input data
  :
  :
  * return the status to the caller by creating the status container
  * on the payroll channel and putting a value in it
  EXEC CICS PUT CONTAINER('status') FROM('OK')
  :
  :
WHEN ch_name 'payroll-2005'
  * my current channel is "payroll-2005"
  :
  :
  :

```

Figure 37. How a server program can query the channel it's been passed, retrieve data from the channel's containers, and return output to the caller

Dynamic transaction routing program with channels

CICS invokes the dynamic transaction routing program when a transaction defined as DYNAMIC(YES) is initiated. The routing program's communication area is mapped by the DFHDYPDS DSECT. The routing program is passed, in the DYRCHANL field of DFHDYPDS, the name of the channel, if any, associated with the request. If the application has, within the channel, created a special container named DFHROUTE, the routing program is passed in the DYRACMAA and DYRACMAL fields the address and the length of the DFHROUTE container data. Any other containers that are created within the channel are not available to the dynamic transaction routing program.

If you are migrating a program to pass a channel rather than a COMMAREA, you could use its existing COMMAREA structure to map DFHROUTE.

- For terminal-initiated transactions the dynamic transaction routing program is invoked for routing (DYRFUNC=0). It then can change the contents of the DFHROUTE container.

Example: An pseudo-conversational application that uses a channel creates within the channel a special container named DFHROUTE. This application issues a RETURN TRANSID CHANNEL request for a transaction that is to be dynamically routed. The dynamic routing program is given can modify the DFHROUTE container data.

- For transactions started by terminal-related START commands (ATI), the dynamic transaction routing program is invoked for notification (DYRFUNC=3). It can inspect but not change the DFHROUTE container data.

Data conversion

Why is data conversion needed?

Here are some cases in which data conversion is necessary:

- When character data is passed between platforms that use different encoding standards; for example, EBCDIC and ASCII.
- When you want to change the encoding of some character data from one Coded Character Set Identifier (CCSID) to another. For an explanation of CCSIDs, see [CICS Family: Communicating from CICS on System/390](#) publication.

Preparing for code page conversion with channels

The conversion of data to or from either UTF-8 or UTF-16 and EBCDIC and ASCII code pages and conversion between the UTF-8 and UTF-16 forms of Unicode is supported in CICS TS 2.2.

Note: You need to be aware that not all points in each code page have direct counterparts in other code pages. The EBCDIC character NL is one such example. For example, CICS TS for z/VSE, CICS TS for z/OS, and Java may differ in the conversions that they perform. "Technotes", and other internet discussions may offer guidance on particular points. It is also worth observing that programming communities are themselves divided on the question of what is the more appropriate conversion in particular circumstances.

CICS now supports any of these character conversions by making use of z/VSE conversion services. However, those conversions that earlier releases of CICS carried out using a set of tables, continue to be supported in that manner. It is only if CICS TS 2.2 is asked to carry out a conversion between a pair of CCSIDs that are unsupported via these tables, that it attempts the conversion using z/VSE services.

Handling CCSID 1200

CICS supports conversions involving UTF-16 data using any of the following CCSID's: 1200, 1201, and 1202. z/VSE conversion services permit CCSID 1200, in its big-endian form, to be used, but does not contain support for the little-endian form or for CCSIDs 1201 or 1202. CICS transforms any source data that is identified in any of these unsupported forms to the big-endian form of 1200 before passing the data to z/VSE for conversion. If the target data is one of the unsupported forms, then CICS receives the data as the big-endian form of 1200 and transforms it to the required CCSID. If the target CCSID is 1200, then CICS assumes the encoding to be in big-endian form. If the conversion is between any of these CCSIDs, then CICS will carry out the transformation without calling z/VSE conversion services.

CICS respects the byte order marker for inbound conversions, but is not able to retain that information when handling a related outbound conversion. All outbound data for CCSID 1200 is UTF16-BE. Application programmers need to know about this and perform their own BE to LE conversions if they so require.

List of supported Unicode code pages and their Coded Character Set Identifiers (CCSIDs)

Coded character set	IANA charset	IBM CCSID	Byte Order Mask (BOM)
UTF-8	utf-8	1208	optional
UTF-16 (*)	utf-16 (*)	1200 (*)	optional
UTF-16 big-endian	utf-16be	1201	
UTF-16 little-endian	utf-16le	1202	

*) See "Handling CCSID 1200" on page 232 above for details on how code page UTF-16 (CCSID 1200) is handled, especially the endianness (byte-order).

Data conversion with channels

Applications that use channels to exchange data use a simple data conversion model. Frequently, no conversion is required and, when it is, a single programming instruction can be used to tell CICS to handle it automatically.

Note the following:

- Usually, when a CICS TS program calls another CICS TS program, no data conversion is required, because both programs use EBCDIC encoding. For example, if a CICS TS C-language program calls a CICS TS COBOL program, passing it some containers holding character data, the only reason for using data conversion would be the unusual one of wanting to change the CCSID of the data.

- The data conversion model used by channel applications is much simpler than that used by COMMAREA applications. Applications that use COMMAREAs to exchange data use the traditional data conversion model described in the [CICS Family: Communicating from CICS on System/390](#) publication. Conversion is done under the control of the system programmer, using the DFHCNV conversion table, the DFHCCNV conversion program and, optionally, the DFHUCNV user-replaceable conversion program. In contrast, the data in channel containers is converted under the control of the application programmer, using API commands.
- The application programmer is responsible only for the conversion of user data; that is, the data in containers created by the application programs. System data is converted automatically by CICS, where necessary.
- The application programmer is concerned only with the conversion of character data. The conversion of binary data (between big-endian and little-endian) is not supported.
- Your applications can use the container API as a simple means of converting character data from one code page to another.

For data conversion purposes, CICS recognizes two types of data:

CHAR

Character data; that is, a text string. The data in the container is converted (if necessary) to the code page of the application that retrieves it. If the application that retrieves the data is a client on an ASCII-based system, this will be an ASCII code page. If it is a CICS Transaction Server for z/VSE application, it will be an EBCDIC code page.

All the data in a container is converted as if it was a single character string. For single-byte character set (SBCS) code pages, a structure consisting of several character fields is equivalent to a single-byte character string. However, for double-byte character set (DBCS) code pages, this is not the case. If you use DBCS code pages, to ensure that data conversion works correctly you must put each character string into a separate container. For more information about DBCS, see the [IBM Glossary](#).

BIT

All non-character data; that is, everything that is not designated as being of type CHAR. The data in the container cannot be converted. This is the default value.

There are two ways to specify the code page for data conversion of the data in a container:

- As a Coded Character Set Identifier, or CCSID. A CCSID is a decimal number that identifies a specific code page. For example, the CCSID for the ASCII character set ISO 8859-1 is 819.
- As an IANA-registered charset name for the code page. This is an alphanumeric name that can be specified in `charset=` values in HTTP headers. For example, the IANA charset names supported by CICS for ISO 8859-1 is `iso-8859-1`.

If the application programmer does not specify a code page for data conversion, CICS uses the default code page for the whole of the local CICS region, which is specified on the **LOCALCCSID** system initialization parameter.

You can use the following API commands for data conversion:

- EXEC CICS PUT CONTAINER [CHANNEL] [DATATYPE] [FROMCCSID | FROMCODEPAGE]
- EXEC CICS GET CONTAINER [CHANNEL] [INTOCCSID | INTOCODEPAGE]

How to cause CICS to convert character data automatically

You can use the DATATYPE(DFHVALUE(CHAR)) option of the PUT CONTAINER command to specify that a container holds character data eligible for conversion. If the client and server platforms are different, the GET CONTAINER command converts the data automatically.

About this task

The following procedure demonstrates how to convert character data automatically by using PUT CONTAINER and GET CONTAINER commands.

Procedure

1. In the **client program**, use the DATATYPE(DFHVALUE(CHAR)) option of the PUT CONTAINER command to specify that a container holds character data and that the data is eligible for conversion.

For example:

```
EXEC CICS PUT CONTAINER(cont_name) CHANNEL('payroll')
           FROM(data1) DATATYPE(DFHVALUE(CHAR))
```

There is no need to specify the FROMCCSID or FROMCODEPAGE option unless the data is not in the default CCSID of the client platform. (For CICS TS regions, the default CCSID is specified on the LOCALCCSID system initialization parameter.) The default CCSID is implied.

2. In the **server program**, issue a GET CONTAINER command to retrieve the data from the program's current channel:

```
EXEC CICS GET CONTAINER(cont_name) INTO(data_area1)
```

The data is returned in the default CCSID of the server platform. There is no need to specify the INTOCCSID or INTOCODEPAGE option unless you want the data to be converted to a CCSID other than the default. If the client and server platforms are different, data conversion takes place automatically.

3. In the **server program**, issue a PUT CONTAINER command to return a value to the client:

```
EXEC CICS PUT CONTAINER(status) FROM(data_area2)
           DATATYPE(DFHVALUE(CHAR))
```

The DATATYPE(DFHVALUE(CHAR)) option specifies that the container holds character data and that the data is eligible for conversion. There is no need to specify the FROMCCSID or FROMCODEPAGE option unless the data is not in the default CCSID of the server platform.

4. In the **client program**, issue a GET CONTAINER command to retrieve the status returned by the server program:

```
EXEC CICS GET CONTAINER(status) CHANNEL('payroll')
           INTO(status_area)
```

The status is returned in the default CCSID of the client platform. There is no need to specify the INTOCCSID or INTOCODEPAGE option unless you want the data to be converted to a CCSID other than the default. If the client and server platforms are different, data conversion takes place automatically.

Using containers to do code page conversion

Your applications can use the container API as a simple means of converting character data from one code page to another.

About this task

The following example converts data from codepage1 to codepage2:

```
EXEC CICS PUT CONTAINER(temp) DATATYPE(DFHVALUE(CHAR))
           FROMCCSID(codepage1) FROM(input-data)
```

```
EXEC CICS GET CONTAINER(temp) INTOCCSID(codepage2)
SET(data-ptr) FLENGTH(data-len)
```

The following example converts data from the region's default EBCDIC code page to the specified codepage1:

```
EXEC CICS PUT CONTAINER(temp) DATATYPE(DFHVALUE(CHAR))
FROM(ebcdic-data)
EXEC CICS GET CONTAINER(temp) INTOCCSID(codepage1)
SET(data-ptr) FLENGTH(data-len)
```

When using the container API in this way, bear the following in mind:

- On GET CONTAINER commands, use the SET option, rather than INTO, unless the converted length is known. (You can retrieve the length of the converted data by issuing a GET CONTAINER(*cont_name*) NODATA FLENGTH(*len*) command.)
- If you prefer to specify a supported IANA charset name for the code pages, rather than the decimal CCSIDs, or if you want to specify a CCSID alphanumerically, use the FROMCODEPAGE and INTOCODEPAGE options instead of the FROMCCSID and INTOCCSID options.
- To avoid a storage overhead, after conversion copy the converted data and delete the container.
- All-to-all conversion is not possible. That is, a code page conversion error occurs if a specified code page and the channel's code page are an unsupported combination.

File example

A CICS Transaction Server file application:

1. Read a VSAM KSDS file with client records.
2. Put the EBCDIC data into a container.
3. Get the data from the container in ASCII format.
4. Write a TD queue to externalize the file in ASCII format.

The file interface has no code page conversion capabilities. So, if you want to externalize a file from an EBCDIC code page to an ASCII file, for example, it's easy to use the channels and containers. To read the file records in the region's EBCDIC code page, the program can issue the following command:

```
EXEC CICS READ FILE(filename) INTO(data-area1) RIDFLD(keydata-area)
```

Copy the data and move the record into a container and make it eligible for conversion:

```
EXEC CICS PUT CONTAINER(containername) CHANNEL(channelname)
FROM(data-area1) CHAR
```

The CHAR option specifies that the container holds character data and that the data is eligible for conversion. The FROMCCSID has been omitted, the partition's default CCSID is implied.

The following command converts the container from the current local EBCDIC code page to ASCII:

```
EXEC CICS GET CONTAINER(containername) CHANNEL(channelname)
INTO(data-area2) INTOCCSID(858)
```

Note: You can specify the CCSID directly or you can specify a variable containing the CCSID. For the variable, you must use a binary fullword. For example, in COBOL use the following:

```
01 ASCII CODEPAGE PIC 9(8) BINARY VALUE 858
EXEC CICS GET CONTAINER(containername) CHANNEL(channelname)
INTO(data-area2) INTOCCSID(ASCII CODEPAGE)
```

The following command is to externalize the ASCII record to a transient data (TD) queue file:

```
EXEC CICS WRITEQ TD QUEUE(queuname) FROM(data-area2)
```

Benefits of channels

Using the channel and container model in CICS programs to exchange data has several advantages over using communication areas (COMMAREAs) .

- There is no limit to the number of containers that can be added to a channel, and the size of individual containers is limited only by the amount of storage that you have available.
Take care not to create so many large containers that you limit the amount of storage available to other applications.
- A channel can comprise multiple containers, passing data in a structured way.
- Channels do not require the programs that use them to know the exact size of the data returned.
- A channel is a standard mechanism for exchanging data between CICS programs. A channel can be passed on **LINK**, **START**, **XCTL**, and **RETURN** commands. Distributed program link (DPL) is supported, and the transactions started by **START CHANNEL** and **RETURN TRANSID** commands might be remote.
- Channels can be used by CICS application programs written in any of the language supported by CICS. For example, a C client program on one CICS region can use a channel to exchange data with a COBOL server program on a back-end AOR.
- A server program can be written to handle multiple channels. It can, for example:
 - Discover, dynamically, the channel that it was invoked with
 - Browse the containers in the channel
 - Vary its processing according to the channel it has been passed
- You can build "components" from sets of related programs invoked through one or more channels.
- The loose coupling between clients and components permits easy evolution. Clients and components can be upgraded at different times. For example, first a component could be upgraded to handle a new channel, then the client program upgraded (or a new client written) to use the new channel.
- The programmer is relieved of storage management concerns. CICS automatically destroys containers (and their storage) when they go out of scope.
- The data conversion model used by channel applications is simple and is controlled by the application developer, using simple API commands.

Channels might not be the best solution in all cases. When designing an application, there are one or two implications of using channels that you must consider:

- When a channel is to be passed to a remote program or transaction, passing a large amount of data might affect performance, particularly if the local and remote regions are connected by an ISC, rather than MRO, connection.
- A channel might use more storage than a COMMAREA designed to pass the same data because container data can be held in more than one place. COMMAREAs are accessed by pointers, whereas the data in containers is copied between programs.

Migrating from COMMAREAs to channels

CICS application programs that use traditional communications areas (COMMAREAs) to exchange data can continue to work as before. If you want to migrate to channels, here are examples of how to migrate several types of existing application to use channels and containers rather than COMMAREAs.

It is possible to replace a COMMAREA by a channel with a single container. While this may seem the simplest way to move from COMMAREAs to channels and containers, it is not good practice to do this. Because you're taking the time to change your application programs to exploit this new function, you should implement the "best practices" for channels and containers; see ["Designing a channel: Best](#)

practices” on page 229. Channels have several advantages over COMMAREAs (see “Benefits of channels” on page 236) and it pays to design your channels to make the most of these improvements.

Also, be aware that a channel may use more storage than a COMMAREA designed to pass the same data. (See “Benefits of channels” on page 236.)

User-written dynamic transaction routing programs require work whether or not you plan to implement channels and containers in your own applications. If you employ a user-written dynamic routing program, you must modify your program to handle the new value that it may be passed in the DYRVER field of the DFHDYPDS communications area.

Migrating LINK commands that pass COMMAREAs

To migrate two programs which use a COMMAREA on a LINK command to exchange a structure, change the instructions shown in this table.

About this task

In these instructions, `structure` is the name of your defined data structure. The EXEC CICS GET CONTAINER and PUT CONTAINER commands use a 16-character field to identify the container. A helpful convention is to give the container the same name as the data structure that you are using, shown here as `structure-name`.

Program	Before	After
PROG1	EXEC CICS LINK PROGRAM(PROG2) COMMAREA(structure)	EXEC CICS PUT CONTAINER(structure-name) CHANNEL(channel-name) FROM(structure) EXEC CICS LINK PROGRAM(PROG2) CHANNEL(channel-name) : EXEC CICS GET CONTAINER(structure-name) CHANNEL(channel-name) INTO(structure)
PROG2	EXEC CICS ADDRESS COMMAREA(structure- ptr) ... RETURN	EXEC CICS GET CONTAINER(structure-name) INTO(structure) ... EXEC CICS PUT CONTAINER(structure-name) FROM(structure) RETURN

Note: In the COMMAREA example, PROG2, having put data in the COMMAREA, has only to issue a RETURN command to return the data to PROG1. In the channel example, to return data *PROG2 must issue a PUT CONTAINER command before the RETURN.*

Migrating XCTL commands that pass COMMAREAs

To migrate two programs which use a COMMAREA on an XCTL command to pass a structure, change the instructions shown in this table.

About this task

In these instructions, `structure` is the name of your defined data structure. The EXEC CICS GET CONTAINER and PUT CONTAINER commands use a 16-character field to identify the container. A helpful convention is to give the container the same name as the data structure that you are using, shown here as `structure-name`.

Table 17. Migrating XCTL commands that pass COMMAREAs

Program	Before	After
PROG1	EXEC CICS XCTL PROGRAM(PROG2) COMMAREA(structure)	EXEC CICS PUT CONTAINER(structure-name) CHANNEL(channel-name) FROM(structure) EXEC CICS XCTL PROGRAM(PROG2) CHANNEL(channel-name) :
PROG2	EXEC CICS ADDRESS COMMAREA(structure- ptr) ...	EXEC CICS GET CONTAINER(structure-name) INTO(structure) ...

Migrating pseudoconversational COMMAREAs on RETURN commands

To migrate two programs which use COMMAREAs to exchange a structure as part of a pseudoconversation, change the instructions shown in this table.

About this task

In these instructions, `structure` is the name of your defined data structure. The EXEC CICS GET CONTAINER and PUT CONTAINER commands use a 16-character field to identify the container. A helpful convention is to give the container the same name as the data structure that you are using, shown here as `structure-name`.

Table 18. Migrating pseudoconversational COMMAREAs on RETURN commands

Program	Before	After
PROG1	EXEC CICS RETURN TRANSID(PROG2) COMMAREA(structure)	EXEC CICS PUT CONTAINER(structure-name) CHANNEL(channel-name) FROM(structure) EXEC CICS RETURN TRANSID(TRAN2) CHANNEL(channel-name)
PROG2	EXEC CICS ADDRESS COMMAREA(structure- ptr)	EXEC CICS GET CONTAINER(structure-name) INTO(structure)

Migrating START data

To migrate two programs which use START data to exchange a structure, change the instructions shown in this table.

About this task

In these instructions, `structure` is the name of your defined data structure. The EXEC CICS GET CONTAINER and PUT CONTAINER commands use a 16-character field to identify the container. A helpful convention is to give the container the same name as the data structure that you are using, shown here as `structure-name`.

Program	Before	After
PROG1	EXEC CICS START TRANSID(TRAN2) FROM(structure)	EXEC CICS PUT CONTAINER(structure-name) CHANNEL(channel-name) FROM(structure) EXEC CICS START TRANSID(TRAN2) CHANNEL(channel-name)
PROG2	EXEC CICS RETRIEVE INTO(structure)	EXEC CICS GET CONTAINER(structure-name) INTO(structure)

Note that the new version of PROG2 is the same as that in the pseudoconversational example.

Channels and containers storage requirements

Using channels and containers to replace COMMAREA usage or other methods of passing data is likely to increase the requirements for CICS storage even without increasing the size of the data being passed. Therefore CICS storage usage should be monitored carefully to avoid SOS, especially if 24-bit programs use the API.

Please keep the following in mind:

1. CICS keeps the source copy of every in-scope container created or updated by PUT CONTAINER in 31-bit ECDSA storage. GET CONTAINER is used to retrieve the current data, and PUT CONTAINER must be used to change the data.
2. One or more copies of the container's data will typically exist in contiguous areas of task-related storage due to PUT CONTAINER FROM and GET CONTAINER INTO being used.
3. One copy of the container's data will exist in contiguous task-related storage after the first GET CONTAINER SET has been issued by the task, and the SET storage will normally remain allocated until the container is deleted or the channel is no longer in-scope. However, the SET storage address and even size may change as a result of other activity on that container in the same task, resulting in CICS issuing a FREEMAIN and GETMAIN to make the change. For example, a subsequent PUT CONTAINER to change the size followed by a GET CONTAINER SET will normally result in a FREEMAIN and GETMAIN for SET storage. A GET CONTAINER SET for a program that requires 24-bit SET storage will always cause the storage to be re-allocated by a FREEMAIN and GETMAIN.
4. Channels and containers are task-related, therefore, if you use START TRANSID CHANNEL, CICS will create a copy of the specified channel and containers for the new task. Similarly, RETURN CHANNEL will make a copy of the specified channel and containers.
5. Performing container data conversion with SET normally results in four times the amount of storage being allocated in order to perform the conversion. For example, using GET CONTAINER SET to convert 64KB of CCSID 037 (US English) data to 858 (ASCII), will result in 256KB of SET storage being GETMAINED for the conversion even though both CCSIDs use single-byte values. Any previously allocated SET storage will be FREEMAINED before the GETMAIN.
6. START TRANSID CHANNEL or LINK PROGRAM CHANNEL to a remote CICS system will result in a copy of the containers being shipped and hence duplicated in the connected system(s).
7. Container data will be deleted when a channel goes out-of-scope, by DELETE CONTAINER, and any remaining containers will be deleted as part of task termination.

From this you can see that the amount of storage that is required for using a container will be a minimum of twice the container size.

The correct choice of DATALOCATION in the PROGRAM definition may be important to ensure that GET CONTAINER SET storage is allocated in 31-bit and not in 24-bit storage even when you are executing 31-bit programs.

To make it easier to handle large containers and to change container sizes, the ECDSA copy of a container is managed as a series of discontinuous segments. Each segment except the last is a unit of 4KB, and comprises a CSCB control block and 4,056 bytes of container data. The last segment contains a CSCB and the residual amount of data. The ECDSA subpools are PGCSCB4K and PGCSCBV respectively. Changing the size of the container in ECDSA will either add or delete segments. The output from DFHPD420 DATA SM=1 or DFHSTUP will identify PGCSCB4K and PGCSCBV subpool usage.

While CICS needs control blocks other than the CSCB to define a channel and each container, the number and sizes are very small and are not likely to significantly affect ECDSA storage usage. They can be seen in a dump by running DFHPD420 with DATA PG=3.

It is impossible to predict the effect of fragmentation (also known as "Storage Creep") , if any, within CICS storage resulting from the container GETMAIN and FREEMAIN activity, particularly when large containers are used. Fragmentation can effectively reduce the available DSALIM and EDSALIM available storage by increasing the number of available contiguous storage areas while decreasing their sizes. Therefore, you may need to allocate a larger DSALIM or EDSALIM than calculated in order to preserve a suitably large amount of contiguous free storage. Wherever possible, perform load testing of the applications, and use DFHOSTAT or DFHSTUP output as the basis for estimating DSALIM and EDSALIM. Also ensure that the CICS partition GETVIS usage can be expanded by CEMT I DSA if it is necessary to increase DSALIM or EDSALIM while CICS is running.

Channels and containers performance

While it is not possible to provide generic figures for the difference in performance between using channels and containers and other methods of passing data, there are a few considerations.

1. Additional EXEC CICS calls are typically required, and will result in additional cpu time for a task.
2. Using remote DPL and START requests could have a significant impact on response times when additional data is transmitted compared to previously used techniques.
3. Using PUT CONTAINER to create a container in ECDSA will require one GETMAIN for approximately each 4KB of data.
4. Subsequent changes to the size of the container in ECDSA will use GETMAIN and FREEMAIN only to add or remove segments.
5. GET CONTAINER SET will attempt to reuse the storage GETMAINED for an earlier GET CONTAINER SET, however, a FREEMAIN and GETMAIN will always be performed for a program requiring 24-bit storage, and if CICS detects that the size of the area needs to change.
6. Explicit or implicit DELETE CONTAINER will require one FREEMAIN for approximately every 4KB of the ECDSA container data.
7. CCSID conversion is performed one container segment at a time.

Part 9. CICS Assembler Programming

Chapter 35. Relative Addressing Instructions in Assembler Programs

DFHEIENT Macro

For AMODE(24) and AMODE(31) programs, the values provided by the DFHEIENT macro that the translator inserts automatically might be inadequate for application programs that produce a translated output greater than 4095 bytes. In this situation, you can provide your own version of the DFHEIENT macro. To prevent the translator from automatically inserting its version of the DFHEIENT macro, specify the NOPROLOG translator option.

For example, by default, the translator sets up only one base register (register 3), which might cause addressability problems for your program. You can provide your own DFHEIENT macro with the CODEREG operand so that you can specify more than one base register. If you code your own version of the DFHEIENT macro with the same label as the CSECT statement, it can replace the CSECT statement in your source program. If you code the DFHEIENT macro without a label, it must immediately follow the CSECT statement.

You can use the following operands to specify base registers:

- CODEREG – base registers
- DATAREG – dynamic-storage registers
- EIBREG – register to address the EIB

For example, the following simple assembler language application program uses the BMS command SEND MAP to send a map to a terminal.

```
INSTRUCT CSECT
        EXEC CICS SEND MAP('DFH$AGA') MAPONLY ERASE
        END
```

The following example code increases the number of base and data registers for this program:

```
INSTRUCT DFHEIENT CODEREG=(2,3,4),
        DATAREG=(13,5),
        EIBREG=6
        EXEC CICS SEND
        MAP('DFH$AGA')
        MAPONLY ERASE
        END
```

The symbolic register DFHEIPLR is equated to the first DATAREG either explicitly specified or obtained by default. The DFHECALL macro ensures that register 13 addresses the save area that DFHEISTG defined in dynamic storage, so it is advisable to use register 13 as your first data dynamic-storage register. If you do not, the code generated by DFHECALL adds extra instructions to manipulate register 13.

DFHEIPLR is assumed by the expansion of a CICS command to contain the value set up by DFHEIENT. You should either dedicate this register or ensure that it is restored before each CICS command.

You can also use the DFHEIENT macro to specify that you want to use relative addressing instructions in your program. When you use relative addressing, you do not need to use any base registers to address your program instructions, but you must use at least one register to address static data in your program. Specify the following operands on the DFHEIENT macro:

- CODEREG=0 to specify that no registers are to be used to address program instructions.
- STATREG to specify one or more registers to address the static data area in your program.
- STATIC to specify the address of the start of the static data in your program.

If you use relative addressing, use the DFHKEBRC copybook (similar to the EABRCX DEFINE macro provided by z/OS) to redefine the assembler mnemonics for branch instructions to use relative branch instructions. Also, ensure that any LTORG statements, and instructions that are the target of EXECUTE instructions, appear after the label specified in the STATIC operand. For example:

```

COPY DFHKEBRC          Define relative branch mnemonics
RELATIVE DFHEIENT CODEREG=0,STATREG=(8,9),STATIC=MYSTATIC
.....
EX    R2,VARMOVE      Execute instruction in static area
.....

MYSTATIC DS    0D          Static data area
MYCONST DC    C'constant' Static data value
VARMOVE MVC   WORKA(0),WORKB Executed instruction
LTORG ,                Literal pool

```

Assembler language programs that are translated with the DLI option have a DLI initialization call inserted after each CSECT statement. Assembler language programs that are larger than 4095 bytes and that do not use the CODEREG operand of the DFHEIENT macro to establish multiple base registers, must include an LTORG statement to ensure that the literals, generated by either DFHEIENT or a DLI initialization call, fall in the range of the base register.

In general, an LTORG statement is needed for every CSECT that exceeds 4095 bytes in length.

DFHEIRET Macro

For AMODE(24) and AMODE(31) programs, the DFHEIRET macro calls the EPILOG program to release the working storage of the application program.

You can specify the following parameter for the DFHEIRET macro:

- RCREG – Specify the register into which the return code is placed. If you do not specify a value, a return code is not passed back to the caller of the program. There is no default value.

The translator inserts the DFHEIRET macro, with no parameters specified, immediately before the END statement (unless you specify the NOEPILOG translator option to prevent this). END must be in uppercase for the translator to recognize it. If the DFHEIRET macro is invoked before this translator-inserted DFHEIRET macro, the translator-inserted macro does not generate any code.

Chapter 36. LE MAIN for Assembler

CICS now supports the coding of Language Environment conforming assembler MAIN programs. A new translator option LEASM causes Language Environment function to be used to set up the program's environment.

Translator Options

A new translator option LEASM (valid only for assembler programs) allows you to specify that this program is to be translated using the macros that will make it a Language Environment-conforming program, ready for assembly as a MAIN program.

Specification of LEASM results in the setting of a new assembler global &DFHEILE. A fourth positional parameter LE has been added to the DFHEIGBL macro that the translator inserts at the top of every output file. DFHEIGBL changes to set &DFHEILE on if LEASM is specified.

The translator sets a value (X'12') for the language in ARG0 when LEASM has been specified. The output from the translator will be identical to that produced without specifying LEASM in every other way.

If &DFHEILE is set, the DFHEISTG, DFHEIENT, DFHEIRET and DFHEIEND macros expand differently to create an LE environment rather than a normal CICS environment. This allows your programs that have used NOPROLOG and NOEPILOG and coded their own DFHEIENT and other macros to take advantage of Language Environment support without changing their program source. For example, all programs that require more than one code base register fall into this category because the translator does not support multiple code base registers.

- **DFHEISTG**

If &DFHEILE is set, the statement CEEDSA SECTYPE=OS will be added at the top of DFHEISTG, immediately after the DSECT statement. This causes the standard Language Environment DSA to be included, without a DSECT statement.

- **DFHEIENT**

DFHEIENT generates a Language Environment CEEENTRY macro that sets up code addressability and working storage, rather than using the standard CICS methods:

- Code addressability

DFHEIENT has a CODEREG parameter that defaults to 3. When the translator option LEASM is specified, the CEEENTRY is generated by DFHEIENT with the BASE parameter a straight copy of the CODEREG value. CEEENTRY does not allow the use of registers 2 or 12 for code addressability. Register 12 must always contain the address of the Language Environment CAA during execution of a Language Environment program. Register 2 is not generally used as a code base because it is modified by instructions such as TRT. DFHEIENT will generate an error message if Register 2 or Register 12 is specified on CODEREG in a program translated using the LEASM option.

- Working storage addressability

DFHEIENT has a DATAREG parameter that defaults to 13. CEEENTRY does not have an equivalent option. LE z/VSE does not support anything other than R13 for working storage (DSA), and does not support multiple working storage registers. DFHEIENT disallows the use of anything other than 13 for DATAREG in a program translated using the LEASM option.

- **DFHEIRET**

Generates CEETERM RC=0 for a program translated using the LEASM option.

Defining Translator Options

- LEASM (Assembler only)

LEASM instructs the translator to generate code for a Language Environment-conforming assembler MAIN program.

If the LEASM option is specified, the DFHEISTG, DFHEIENT, DFHEIRET and DFHEIEND macros expand differently to create a Language Environment-conforming assembler MAIN program, instead of the form of macro expansion used for assembler sub-routines in a CICS environment. This allows customer programs that have used NOPROLOG and NOEPILOG and coded their own DFHEIENT and other macros to take advantage of Language Environment support without changing their program source. For example, all programs that require more than one code base register fall into this category because the translator does not support multiple code base registers. If the LEASM option is specified, it is required to specify the label denoting the entry point on the END statement. Otherwise an ASRD abend will occur.

For an example of an assembler program translated using the LEASM option see [“EXAMPLE Assembler language PROGRAM using LEASM”](#) on page 247.

Using the EXEC Interface Modules

A language translator reads your source program and creates a new one; normal language statements remain unchanged, but CICS commands are translated into CALL statements of the form required by the language in which you are coding. The calls invoke CICS-provided “EXEC” interface modules or stubs, which is a function-dependent piece of code used by the CICS high-level programming interface. The stub, provided in the PRD1.BASE library, must be link-edited with your application program to provide communication between your code and the CICS EXEC interface program, DFHEIP. These stubs are invoked during execution of EXEC CICS and EXEC DLI commands.

There are stubs for each programming language.

Language	Interface Module Name
Assembler and Assembler MAIN programs using the LEASM option	DFHEAI and DFHEAIO
All HLL languages (C/VSE, COBOL/VSE, and PL/I VSE)	DFHELII

The CICS-supplied stub routines work with an internal programming interface, the CICS command-level interface, which is never changed in an incompatible way. Consequently, these stub modules are upward and downward compatible, and CICS application modules never need to be re-linked to include a later level of any of these stubs.

With the exception of DFHEAIO, these stubs all provide the same function, which is to provide a linkage from EXEC CICS commands to the required CICS service. The stubs make this possible by providing various entry points that are called from the translated EXEC CICS commands, and then executing a sequence of instructions that pass control to the EXEC interface function of CICS.

DFHELII contains multiple entry points, most of which provide compatibility for very old versions of the CICS PL/I translator. It contains the entries DFHEXEC (for C application programs), DFHEI1 (for COBOL and assembler), and DFHEIO1 (for PL/1).

Each of these stubs begins with an 8 byte eyecatcher in the form DFHYxxxx, where x indicates the language supported by the stub (for example, A represents assembler, and I indicates that the stub is language independent), and nnn indicates the CICS release from which the stub was included. The letter Y in the eyecatcher indicates that the stub is read-only. Stubs supplied with very early releases of CICS

contained eyecatchers in the form DFHExxxx in which the letter E denotes that the stub is not read-only. The eyecatcher for DFHELII in CICS Transaction Server for z/VSE, Version 2 Release 2 is DFHYI430.

The eyecatcher can be helpful if you are trying to determine the CICS release at which a CICS application load module was most recently linked.

The DFHEAI stub must be included at the beginning of the program in the output from the link edit. To achieve this, the INCLUDE statement for DFHEAI must be the first in the link-edit step of your JCL.

Assembler language

Each EXEC command is translated into an invocation of the DFHECALL macro.

The following example shows the output generated by the translator when processing a simple EXEC CICS RETURN command:

```
*      EXEC CICS RETURN
      DFHECALL =X'0E080000080001000'
```

The assembly of the DFHECALL macro invocation shown above generates code that builds a parameter list addressed by register 1, loads the address of entry DFHEI1 in register 15, and issues a BALR instruction to call the stub routine.

```
DS      0H
LA      1,DFHEITPL
LA      14,=X'0E08000008001000'
ST      14,0(,1)
OI      0(1),X'80'
L       15,=V(DFHEI1)
BALR   14,15
```

The reference to DFHEI1 is resolved by the inclusion of the DFHEAI stub routine in the linkage editor step.

The DFHEAI0 stub for assembler application programs is included by the automatic call facility of the linkage editor. It is called by code generated by the DFHEIENT and DFHEIRET macros to obtain and free, respectively, an assembler application program's dynamic storage area. This stub is required only in assembler application programs; there are no stubs required or supplied to provide an equivalent function for programs written in the high level languages.

EXAMPLE Assembler language PROGRAM using LEASM

```
*ASM XOPTS(CICS,LEASM)
DFHEISTG DSECT
OUTAREA DS      CL200          DATA OUTPUT AREA
*
EILEASM CSECT ,
      MVC  OUTAREA(42),MSG1
      MVC  OUTAREA(4),EIBTRMID
      EXEC CICS SEND TEXT FROM(OUTAREA) LENGTH(42) FREEKB ERASE
      EXEC CICS RECEIVE
      MVC  OUTAREA(13),MSG2
      EXEC CICS SEND TEXT FROM(OUTAREA) LENGTH(13) FREEKB ERASE
      EXEC CICS RETURN
*
MSG1   DC  C'xxxx: LEASM program invoked. ENTER TO END.'
MSG2   DC  C'PROGRAM ENDED'
      END  EILEASM
```

Writing Global User Exit Programs

Assembler programs translated with the LEASM option cannot be used as global user exit programs (GLUEs).

Writing a Task-Related User Exit Program

Assembler programs translated with the LEASM option cannot be used as task-related user exit programs (TRUEs).

Language Environment Considerations for Assembler Applications

Like HLL programs, assembler programs are classified as either conforming or non-conforming with respect to Language Environment. For assembler programs, conformance depends on the linkage and register conventions observed, rather than the assembler used. By definition, a Language Environment-conforming assembler routine is defined as one coded using the CEEENTRY and associated Language Environment macros.

However, there are differences in register conventions and other requirements between the MAIN and sub-routine types. These are described below. Rules for mixing languages, including assembler, are discussed in Mixing languages in Language Environment for z/VSE Writing Interlanguage Communication Applications.

Conforming MAIN programs

The following applies if you explicitly specify LE macros within your program: If you are coding a new assembler MAIN program that you want to conform to the Language Environment interface or if your assembler routine calls Language Environment services, observe the following:

- Use the macros provided by Language Environment. For a list of these macros, see the Language Environment for z/VSE Programming Guide.
- Ensure that the CEEENTRY macro contains the option MAIN=YES. (MAIN=YES is the default).
- Translate your assembler routine with *ASM XOPTS(LEASM) or, if it contains CICS commands, with *ASM XOPTS(LEASM NOPROLOG NOEPILOG).

Conforming sub-routines programs

The following applies if you explicitly specify LE macros within your program: If you are coding a new assembler sub-routine that you want to conform to the Language Environment interface or if your assembler routine calls Language Environment services, observe the following:

- Use the macros provided by Language Environment. For a list of these macros, see the Language Environment for z/VSE Programming Guide.
- Ensure that the CEEENTRY macro contains the option MAIN=NO. (MAIN=YES is the default).
- Translate your assembler routine with *ASM XOPTS(NOPROLOG NOEPILOG) if it contains CICS commands.

All conforming routines

To communicate properly with assembler routines, observe certain register conventions on entry to the assembler routine, while it is running, and on exit from the assembler routine.

• Entry

On entry into a Language Environment-conforming assembler subroutine, these registers must contain the following values when NAB=YES is specified on the CEEENTRY macro:

- R0: Reserved
- R1: Address of the parameter list, or zero
- R12: Common anchor area (CAA) address
- R13: Caller's dynamic storage area (DSA)
- R14: Return address

- R15: Entry point address

Language Environment-conforming HLLs generate code that follows these register conventions, and the supplied macros do the same when you use them to write your Language Environment-conforming assembler routine. On entry to an assembler routine, CEEENTRY saves the caller's registers (R14 through R12) in the DSA provided by the caller. It allocates a new DSA and sets the NAB field correctly in this new DSA. The first half word of the new DSA is set to binary zero and the back chain in the second word is set to point to the caller's DSA.

- **While the subroutine is running**

R13 must point to the routine's DSA at all times while the Language Environment-conforming assembler routine is running.

At any point in your code where you CALL another program, R12 must contain the common anchor area (CAA) address, except in the following cases:

- When calling a COBOL/VSE program
- When calling an assembler routine that is not Language Environment-conforming
- When calling a Language Environment-conforming assembler routine that specifies NAB=NO on the CEEENTRY macro

- **Exit**

On exit from a Language Environment-conforming assembler routine, R0, R1, R14, and R15 are undefined. All the other registers must have the contents they had upon entry.

The CEEENTRY macro automatically sets a module to AMODE (ANY) and RMODE (ANY). If you are converting an existing assembler routine to be Language Environment-conforming and the routine contains data management macros coded using 24-bit addressing mode, then you should change the macros to use 31-bit mode. If it is not possible to change all the modules making up a program to use 31-bit addressing mode, and if none of the modules explicitly sets RMODE (24), then you should set the program to be RMODE (24) during the link-edit process.

Non-conforming routines

Observe the following conventions when running non-Language Environment-conforming subroutines under Language Environment:

- R13 must contain the address of the executing routine's register save area.
- The first four bytes of the register save area must be binary zeros.
- The register save area back chain must be set to a valid 31-bit address (the high-order byte must be zero if it is a 24-bit address of the callers save area).
- Any conditions, interrupts or failures experienced by the non-conforming routine or subsequently called routines may result in an internal LE abend being reported for the CICS transaction.

If your assembler routine relies on C, COBOL, or PL/I control blocks (for example, a routine that tests flags or switches in these control blocks), check that these control blocks have not changed under Language Environment. For more information, see the appropriate Language Environment migration guide.

Non-conforming routines cannot use Language Environment callable services.

Note: CICS does not allow the use of HANDLE ABEND LABEL in Assembler programs that do not use DFHEIENT and DFHEIRET. These Assembler programs should either use HANDLE ABEND PROGRAM or a Language Environment service such as CEEHDLR. See [“Using Language Environment Abend-handling” on page 249](#).

Using Language Environment Abend-handling

When a CICS application is running under Language Environment, the action taken when a task is scheduled for abnormal termination depends on whether a CICS HANDLE ABEND is active or not active.

When a HANDLE ABEND is active, Language Environment condition handling does not gain control for any abends or program interrupts, and any user-written condition handlers that have been established by CEEHDLR are ignored. Instead, the action defined in the CICS HANDLE ABEND takes place.

When a CICS HANDLE ABEND is not active, Language Environment condition handling does gain control for abends and program interrupts if the run-time option TRAP(ON) is specified. Normal Language Environment condition handling is then performed. If TRAP(OFF) is specified, no error handling takes place; the abend proceeds. For details of the actions taken during normal Language Environment condition handling, see the [LE/VSE Programming Guide](#).

User-written Language Environment condition handlers

The run-time option USRHDLR allows you to register a user-written condition handler at the highest level. At a lower level, for example after a subroutine CALL, you can use the CEEHDLR service to register a condition handler for that level. This lower level handler is automatically unregistered on return from the lower level. If desired you can explicitly unregister it by using the CEEHDLU service. For an explanation of stack levels and for details of the USRHDLR run-time option and the CEEHDLR and CEEHDLU services, see the [LE/VSE Programming Guide](#).

If you write your own user-written Language Environment condition handler (other than in COBOL), you can use most CICS commands, provided they are coded with a NOHANDLE, RESP or RESP2 option, to prevent further conditions being raised during execution of the condition handler. The only commands you cannot use are the following, which must not appear in either the condition handler or any program it calls:

- ABEND
- HANDLE ABEND
- HANDLE AID
- HANDLE CONDITION
- IGNORE CONDITION
- POP HANDLE
- PUSH HANDLE

Unless you use the NOLINKAGE translator option (see NOLINKAGE), do not use the CICS translator to translate a COBOL user-written condition handler that you have registered for a routine using the CEEHDLR service. This is because the CICS translator adds two extra arguments to the PROCEDURE DIVISION header of the COBOL program, the EXEC Interface Block (EIB) and the COMMAREA. These arguments do not match the arguments passed by Language Environment. A COBOL condition handler cannot, therefore, contain any CICS commands.

However, a user-written condition handler can call a subroutine to perform CICS commands (and this could be a COBOL routine). If you need to pass arguments to this subroutine, place two dummy arguments before them in the caller. The called subroutine must issue EXEC CICS ADDRESS EIB(DFHEIPTR) before executing any other CICS commands.

Part 10. Migration to CICS Transaction Server for z/VSE 2.2

Chapter 37. Migration Tasks - Overview

This part of the publication deals with changes that affect CICS externals, such as system definitions and programming interfaces.

This topic gives an overview of tasks to be done or at least considered when migrating to CICS Transaction Server for z/VSE 2.2 (CICS TS for z/VSE 2.2). The migration tasks depend on whether your current CICS is CICS TS for z/VSE 2.1 or CICS TS for VSE/ESA 1.1.1. Details can be found in the topics referenced.

CICS TS for z/VSE 2.2 and z/VSE 6.2

- z/VSE 6.2 is delivered with CICS TS for z/VSE 2.2.
- This new CICS release replaces CICS TS for VSE/ESA 1.1.1 and CICS TS for z/VSE 2.1.
- It is the only CICS release that can be used with z/VSE 6.2. It cannot be used on older versions of z/VSE.

You can upgrade your system to z/VSE 6.2 using initial installation or Fast Service Upgrade (FSU) provided your current system is z/VSE 6.1.

On a z/VSE 6.2 system, all IBM-supplied CICS programs and tables are built with CICS TS for z/VSE 2.2.

Task

Overview

Migrate your DFHSIT

If you use own DFHSIT tables you need to reassemble and relink these tables using the macros of CICS TS for z/VSE 2.2. Refer to [Chapter 38, “System Initialization Table - DFHSIT ,”](#) on page 257.

Other CICS tables

If you use own CICS tables (MCT, DCT, and others) you need to reassemble and relink these tables using the macros of CICS TS for z/VSE 2.2. Refer to [Chapter 39, “Resource Definition Macro Changes,”](#) on page 259.

Migrate your CSD data set

Refer to [Chapter 40, “Migration Planning for CICS System Definition Data Set \(CSD\),”](#) on page 261.

Migrate your TCPIPService definitions

Refer to [Chapter 41, “Migrate TCPIPService Definitions,”](#) on page 263.

Migrate CICS application programs

- If you use EXEC CICS API interfaces within your application programs, you can run your programs unchanged with CICS TS for z/VSE 2.1 or later.
- If you use internal CICS control blocks within your application programs:
 - No internal control blocks were changed with CICS TS for z/VSE 2.2. If you upgraded from CICS TS for z/VSE 2.1 (z/VSE 6.1), you can use your programs as they are.
 - If you use internal CICS control blocks within programs and upgraded from CICS TS for VSE/ESA, it is recommended to reassemble these programs. Refer to [Chapter 42, “Changed internal CICS control blocks ,”](#) on page 265 for additional information.

Migrate Global User Exit Programs

This only affects users that upgrade from a system older than z/VSE 6.1.

- The changes because of channels that were introduced with CICS TS for z/VSE 2.1 are transparent for existing user exit programs.
- With CICS TS for z/VSE 2.1 various internal CICS control blocks were changed. The changes are not transparent (refer to [Chapter 42, “Changed internal CICS control blocks ,”](#) on page 265). If you use these control blocks within your exit programs, you have to reassemble these programs.

Refer to [Chapter 43, “The Global User Exit Programming Interface,”](#) on page 267.

Migrate DFHCNV and DFHUCNV user-replacable programs

- Users upgrading from z/VSE 6.1 are not affected.
- Users upgrading from z/VSE 5.2 or older systems:
If you are using the DFHCNV and DFHUCNV user-replaceable programs you need to reassemble the DFHCNV, and modify the DFHUCNV program. Refer to [Chapter 44, “Changes to User-Replaceable Programs,”](#) on page 269.

Migrate Dynamic Transaction Routing programs

If you use user-written dynamic transaction routing programs, refer to [Chapter 45, “Upgrade Dynamic Transaction Routing Programs,”](#) on page 271 to decide whether you need to update these programs.

If you have enabled monitoring

- If you use your own monitoring control table (MCT) it has to be reassembled.
- You have enabled performance class monitoring, you have to create a performance dictionary record with the DFHMNDUP utility program shipped with CICS TS for z/VSE 2.2.
- With CICS TS for z/VSE 2.1, channel and container related values were added to the performance-class monitoring records. To INCLUDE or EXCLUDE these values you can modify the DFHMCT TYPE=RECORD macro.

Refer to [Chapter 46, “Monitoring and Statistics,”](#) on page 273.

If you use statistical reports

- The version of the DFHSTUP program needs to be identical with the version of the CICS system that generates the data.
- With CICS TS for z/VSE 2.1 new fields were added to the statistics A14 record and the statistics report.
- With CICS TS for z/VSE 2.2 new fields were added to the TCP/IP summary global statistics record.

Refer to [Chapter 46, “Monitoring and Statistics,”](#) on page 273.

Use correct version of CICS-supplied utility programs

Always use the version of the CICS-supplied utility programs shipped with your CICS TS version. Refer to [Chapter 47, “CICS-supplied Utility Programs ,”](#) on page 279.

Additional storage requirements for CICS web support

The upgrade of CWS to HTTP/1.1 requires additional storage:

- 4K for each persistent non-SSL connection
- 40K for each persistent SSL connection

You might need to adapt the EDSALIM system initialization parameter. Note, that this does not affect the ECI interface. The storage requirements for ECI have not changed.

Migration from CICS/VSE 2.3 to CICS TS

Status of CICS/VSE 2.3:

- End of service of CICS/VSE 2.3 is effective since October 31, 2012.
- Starting with z/VSE 4.3, the CICS coexistence environment was dropped.
- Starting with z/VSE 5, CICS/VSE 2.3 can no longer be used with z/VSE.

Migration:

- It is recommended to first migrate from CICS/VSE 2.3 to CICS TS for VSE/ESA 1.1.1 on your current system (z/VSE 4.2 or older). This allows you run both CICS versions in a coexistence environment.
- The CICS Application Migration Aid is no longer shipped with CICS TS for z/VSE 2.1. or later.
- Regarding migration from CICS/VSE 2.3 to CICS TS for VSE/ESA additionally refer to:
 - [CICS Transaction Server for VSE/ESA V1.R1.0 Migration Guide, GC33-1646.](#)
 - *Redbook Migration to VSE/ESA 2.4 and CICS Transaction Server for VSE/ESA 1.1.*

Migration of RPG II online applications:

If your current system is z/VSE 4.1 or 4.2, it is recommended to migrate your RPG II online applications on your current system. Refer to [Chapter 48, “How to migrate RPG II online applications to CICS TS for z/VSE,”](#) on page 281.

Chapter 38. System Initialization Table - DFHSIT

Migration Considerations

These are the z/VSE and CICS TS for z/VSE supplied pre-generated system initialization tables:

- DFHSITSP (source DFHSITSP.A) and DFHSITC2 (source DFHSITC2.A) provided by z/VSE in sublibrary IJSYSRS.SYSLIB.
- DFHSIT (default values, source DFHSIT\$\$A) and DFHSIT6\$ (sample, source DFHSIT6\$.A) provided by CICS TS for z/VSE in sublibrary PRD1.BASE.

For clients that use one of the above system initialization tables without changing them, no further action is required.

After initial installation of z/VSE 6.2, CICS TS for z/VSE (job CICSICCF) is started with the z/VSE provided initialization table DFHSITSP. For use with a second CICS, z/VSE provides a second system initialization table DFHSITC2

If you want to migrate your own DFHSITxx tables, including modified DFHSITSP or DFHSITC2 tables, from an earlier release of CICS, you have to recompile and re-link these tables on z/VSE 6.2 using the macros in sublibrary PRD1.BASE.

It is recommended to catalog these tables in sublibrary PRD2.CONFIG:

- PRD2.CONFIG is included in the standard LIBDEF search chain,
- In case of a CICS startup failure, you can IPL your system in BASIC mode (enter MSG BG and select BASIC). This will start CICS without sublibrary PRD2.CONFIG.

Note: If you use a DFHSITxx table which was built on a CICS TS for VSE/ESA system, the CICS TS for z/VSE startup fails with message:

```
DFHPA1107 applid A 411 VERSION OF MODULE DFHSITxx WAS LOADED. CICS  
CAN ONLY INITIALIZE WITH THE CURRENT LEVEL SIT.
```

If you use a DFHSITxx table which was built on a CICS TS for z/VSE 2.1 system, the CICS TS for z/VSE startup fails with message:

```
DFHPA1107 applid A 420 VERSION OF MODULE DFHSITxx WAS LOADED. CICS  
CAN ONLY INITIALIZE WITH THE CURRENT LEVEL SIT.
```

New System Initialization Parameters

Refer to [Chapter 50, “CICS system initialization parameters,”](#) on page 293.

Chapter 39. Resource Definition Macro Changes

This topic summarizes the changes to the CICS resource definition macros for CICS control tables.

It discusses the following topics:

- [“Reassembling Control Tables” on page 259](#)

Reassembling Control Tables

Reassemble *all* CICS control tables using the CICS TS for z/VSE 2.2 macro sublibrary PRD1.BASE, even if there are no changes to the macro externals. This applies also to tables that you are reassembling only to migrate them to the CSD.

If a back-level control table is loaded, the CICS startup will either fail or a message will be issued. For example, in case of a back-level DCT, the system will issue message:

```
DFHSI1594 applid A 4.1.1 level of module DFHDCTxx is being loaded.
```

or

```
DFHSI1594 applid A 4.2.0 level of module DFHDCTxx is being loaded.
```

In case of a back-level SIT, the CICS startup will fail. Refer to [Chapter 38, “System Initialization Table - DFHSIT,” on page 257](#).

Chapter 40. Migration Planning for CICS System Definition Data Set (CSD)

Upgrading the CSD

This section summarizes migration aspects for the CICS system definition data set (DFHCSD).

After initial installation of z/VSE 6.2 or after FSU to z/VSE 6.2, the IBM-supplied DFHCSD data set (CICS.CSD in VSESPUC) contains the IBM-supplied CICS TS for z/VSE 2.2 and z/VSE 6.2 resource definitions.

Note that the DFHCSDUP utility requires a SIZE parameter of at least 400K:

```
EXEC DFHCSDUP,SIZE=400K
```

In the case of FSU:

- The IBM-supplied DFHCSD data set is upgraded using the UPGRADE function of the DFHCSDUP utility program.
- Your own groups (if any) in the IBM-supplied DFHCSD data set are kept, since the UPGRADE command does not operate on your own groups.
- Any other DFHCSD data set is kept. Use the DFHCSDUP UPGRADE function to upgrade any CICS-supplied resource definitions within these data sets.
- The only resource that has changed with CICS TS for z/VSE 2.2 is TCPIP SERVICE. New attributes were added. All TCPIP SERVICE resource definitions must be recreated using either the DEFINE function of DFHCSDUP or the online function CEDA DEFINE. Otherwise INSTALL of the TCPIP SERVICE resource will fail. Refer to [Chapter 41, “Migrate TCPIP Service Definitions,”](#) on page 263.

If, for example after initial installation, you need to migrate your own CSD definitions from a previous z/VSE release:

- Ensure that you have a backup copy of every DFHCSD that is updated during the migration process in case of problems.
- Use the DFHCSDUP utility program if you need to update the IBM-supplied DFHCSD data set on z/VSE 6.2 with your own resource definitions
- For example, if you used VSE/VSAM to migrate any other DFHCSD data set, use the DFHCSDUP UPGRADE function to upgrade any CICS-supplied resource definitions within these data sets.

How to get your own resource definitions:

- Run on your previous z/VSE release the DFHCSDUP utility program with the LIST ALL function to list all resource definitions in the DFHCSD.
- Alternatively use the DFHCSDUP EXTRACT command to extract resource definition data from the CSD, either from a list or from a group, and invoke a user program, for example the CICS supplied sample program DFH0CBDC. The CICS-supplied sample CSD backup utility program, DFH0CBDC, produces a file of DFHCSDUP DEFINE control statements. The file can be used to recreate or add resource definitions to any CSD using DFHCSDUP. The file can be used to recreate or add resource definitions to any CSD using DFHCSDUP.
- Modify the DEFINE statement for TCPIP SERVICE resource definitions. Refer to [Chapter 41, “Migrate TCPIP Service Definitions,”](#) on page 263.

Note:

- Ensure that you do not back-level IBM-supplied resource definitions.
- With z/VSE 6.1, there are no IBM-supplied resource definition changes (neither CICS nor z/VSE).

- With z/VSE 6.2, there are no z/VSE supplied resource definition changes (VSE-supplied groups are, for example, VSExxx, CEE, TCPIP,..).
- With z/VSE 6.2, there are CICS-supplied resource definition changes. Affected groups are:
 - DFHWEB (transaction CWXU added)
 - DFH\$SOT (TCPIPSERVICE HTTPNSSL changed)
 - DFH\$CTXT, DFH\$ACCT, DFH\$FILA, DFHMROFA, DFHMROFD (file name qualifier CICS420 was changed to CICS430).

Chapter 41. Migrate TCIPService Definitions

When upgrading to CICS TS for z/VSE 2.2 both using initial installation as well as FSU.

- All TCIPSERVICE definitions must be recreated.
- Otherwise the install of an TCIPSERVICE will fail with message DFHAM4912E.

With CICS TS for z/VSE 2.2 new attributes were added to the TCIPSERVICE. These are:

- PROTOCOL
- MAXDATALEN

The ATTACHSEC parameter is only valid with PROTOCOL=ECI and must be deleted for PROTOCOL=HTTP.

Chapter 42. Changed internal CICS control blocks

Note that this description only applies to users migrating from a system older than CICS TS for z/VSE 2.1 (z/VSE 6.1). Users upgrading from CICS TS for z/VSE 2.1 are not affected.

With CICS TS for z/VSE 2.1, internal CICS control blocks were changed. The changes are not transparent. Programs using these control blocks have to be recompiled and re-linked prior to execute them on CICS TS for z/VSE 2.1 or CICS TS for z/VSE 2.2.

The [Table 21 on page 265](#) lists the major control blocks that were changed. Some changes are transparent depending on the usage. This is listed in additional information. Programs, that use internal CICS control blocks which are not listed, should also be recompiled.

Control block	Additional information
DFHA14DS	Refer to “Statistical changes with CICS TS for z/VSE 2.1” on page 275
DFHCNVS	
DFHDYPDS	Refer to Upgrade Dynamic Transaction Routing Programs
DFHGDEFS	
DFHIC	
DFHKC	
DFHMCTDR	
DFHMNDEF	
DFHMNPDA	
DFHMNTDS	Monitoring statistic record not transparent
DFHPCEDS	Refer to Chapter 43, “The Global User Exit Programming Interface,” on page 267
DFHSTD2	
DFHTCTTE	
DFHTCA	
DFHUEXIT	Refer to Chapter 43, “The Global User Exit Programming Interface,” on page 267

Chapter 43. The Global User Exit Programming Interface

Changes because of channels

These changes were introduced with CICS TS for z/VSE 2.1. There are no changes for CICS TS for z/VSE 2.2.

Global user exit programs are passed the channel name of the application program, if any. They can create their own channels and pass them to programs which they call.

The table below lists the global user exits that are passed the name of the current channel.

This is transparent for the user exit program:

New fields containing channel information are appended at the end of the DSECT. The storage area that is passed to the exit and mapped by the DSECT is allocated by CICS TS. Therefore, passing the new fields is transparent to existing user exits. The user exit programs have to be recompiled only if the channel related fields are to be used within the exit program. To recompile, use the macros shipped in PRD1.BASE.

<i>Table 22. New fields in the DSECT</i>			
GLUE Exit	Field Name	CHANNEL Information	Macro
ID=XPCABND ID=XPCTA ID=XPCHAIR ID=XPCFTCH	PCUE_CHANNEL_NAME	Address of a 16-byte field containing the name of the channel with which the application program is to be invoked (that is, the program's current channel). If there is no channel, this field is set to blanks. Blank if called after an abend within an exit.	DFHPCUE: DSECT for program control user exit
ID=XPCREQ ID=XPCREQC	PC_ADDR9 (not used) PC_ADDRA	PC_ADDRA = address of the 16-byte channel name, as specified on the CHANNEL parameter.	DFHPCEDS DSECT for EXEC argument list for Program Control exits. On entry to the XPCREQ and XPCREQC user exits, the EXEC parameter list is pointed to by UEPCPLPS.
ID=XALCAID	UEPALCHN	Address of a 16-byte field containing the name of the channel associated with the AID. If there is no channel associated with the AID, this field is set to blanks.	DFHUEXIT DSECT for terminal allocation program exit

Table 22. New fields in the DSECT (continued)

GLUE Exit	Field Name	CHANNEL Information	Macro
ID=XICEREQ ID=XICEREQC	IC_ADDR1F	IC_ADDR1F = is the address of a 16-byte area containing the value of CHANNEL, if any.	DFHICUED DSECT for EXEC argument list for Interval Control user exits. On entry to the XICEREQ and XICEREQC user exits, the EXEC parameter list is pointed to by UEPLPS.

Internal Control block changes

With CICS TS for z/VSE 2.2, no internal CICS control blocks were changed.

With CICS TS for z/VSE 2.1, various internal CICS control blocks were changed and the changes are not transparent. User exit programs which use these control blocks have to be recompiled. For more information, refer to the section [Chapter 42, “Changed internal CICS control blocks ,”](#) on page 265.

Chapter 44. Changes to User-Replaceable Programs

No changes were introduced with CICS TS for z/VSE 2.2.

With CICS TS for z/VSE 2.1, changes were introduced affecting the following user-replaceable programs:

- DFHCNV
- DFHUCNV

DFHCNV or DFHUCNV programs that are migrated from CICS TS for z/VSE 2.1 (z/VSE 6.1) will run unchanged on CICS TS for z/VSE 2.2 (z/VSE 6.2). The following considerations apply for users migrating from a CICS TS for VSE/ESA (z/VSE 5.2) or older.

DFHCNV

CICS TS for z/VSE 2.1 introduced a new DFHCNV macro parameter operand. The new operand SYSDEF has been added to the TYPE=INITIAL and TYPE=ENTRY macro parameters CLINTCP and SRVERCP. These macros define the user-replaceable data conversion table DFHCNV. When upgrading from CICS TS for VSE/ESA to CICS TS for z/VSE the DFHCNV TYPE=INITIAL macro defines the beginning of the conversion table. It gives a list of valid code pages. The DFHCNV TYPE=ENTRY macro specifies a name and type to uniquely identify a data resource. There must be one for each resource for which conversion is required.

The DFHCNV data conversion table has to be reassembled using the CICS TS for z/VSE macros in sublibrary PRD1.BASE. CICS initialization fails (with message DFHPC0401 and abend ACN4) when trying to load a DFHCNV table assembled using macros from CICS TS for VSE/ESA (instead of earlier release).

DFHUCNV

If you specify USREXIT=YES on the DFHCNV TYPE=ENTRY macro, CICS loads the DFHUCNV user-replaceable program.

With CICS TS for z/VSE 2.1 there are changes to the offset and length parameters passed to the DFHUCNV program. The changes to offsets, and the use of fullword instead of halfword values, means that DFHUCNV has to be modified and reassembled using CICS TS for z/VSE macros in sublibrary PRD1.BASE (when upgrading from CICS TS for VSE/ESA only).

Chapter 45. Upgrade Dynamic Transaction Routing Programs

This section describes the changes introduced with CICS TS for z/VSE 2.1.

The DFHDYPDS communication area was changed with CICS TS for z/VSE 2.1, but was not changed with CICS TS for z/VSE 2.2. Users migrating from CICS TS for z/VSE 2.1 (z/VSE 6.1) are not affected.

Users migrating from a z/VSE 5.2 or older system: If you use a user-written dynamic transaction routing program you might need to update this program to handle the increased length (DYRCLEN) or the new value that is passed in the DYRVER field of the DFHDYPDS communication area. Depending on your usage of the DYRVER field, or DYRCLEN this is required whether or not you plan to implement channels and containers in your applications.

Note: EIBCALEN contains the increased DYRCLEN value.

Information Passed to the Dynamic Routing Program

The CICS relay program, DFHAPRT, passes information to the dynamic transaction routing program by means of a communications area. The communications area contains fields that are mapped by the DSECT DFHDYPDS. The DFHDYPDS DSECT is described in detail in the [CICS Customization Guide](#), SC33-1652-10. These are the changes that are not reflected in the *CICS Customization Guide*:

- **DYRVER**

Is the version number of the dynamic routing program interface.

- For CICS TS for z/VSE 2.2 the number is 4.
- For CICS TS for z/VSE 2.1 the number is 4.
- For CICS for VSE/ESA the number is 3.

- **DYRACMAA**

- If the user application employs a COMMAREA, the length, in bytes, of the application's COMMAREA.
- If the user application employs a channel and has created, within the channel, a container named DFHROUTE, the length, in bytes, of the data in the DFHROUTE container.
- If the user application has no COMMAREA and no DFHROUTE container, zero.

- **DYRCHANL**

Is a 16-bytes field containing the channel name associated with the application program, if any. This field is added at the end.

Chapter 46. Monitoring and Statistics

This topic discusses the following topics:

- Monitoring
- Statistics

Summary of CICS monitoring

CICS monitoring collects data about the performance of user- and CICS-supplied transactions during online processing for later offline analysis. The records produced by CICS® monitoring are of System Management Facility (SMF) type 110, and are written to a DMF data set by the DMF utility. You can request CICS® to collect two types, or classes, of monitoring data: performance class data and exception class data. You can choose which classes of monitoring data you want to be collected.

Performance class monitoring

You can enable performance class monitoring by coding `MNPER=ON` (together with `MN=ON`) as a system initialization parameter.

Alternatively, you can use either the `CEMT SET MONITOR ON PERF` or `EXEC CICS SET MONITOR STATUS(ON) PERFCLASS(PERF)` commands.

When you change the status of CICS monitoring from ON to OFF, CICS stops writing monitoring data immediately. No monitoring data is accumulated for tasks that start after the change is made. If you turn off monitoring, also set `NOPERF` to ensure that buffers containing recorded data for completed tasks are flushed; otherwise, some of this data might be lost.

Exception class monitoring

You can enable exception class monitoring by coding `MNEXC=ON` (together with `MN=ON`) as a system initialization parameter.

Alternatively, you can use either the `CEMT SET MONITOR ON EXCEPT` or `EXEC CICS SET MONITOR STATUS(ON) EXCEPTCLASS(EXCEPT)` commands.

Monitoring control table (MCT)

If you specify the system initialization parameter `MCT=NO`, CICS monitoring builds dynamically a default MCT, ensuring that default monitoring control table entries are always available for use when monitoring is on and a monitoring class (or classes) is active.

Alternatively, you can create your own monitoring control table `DFHMCT`. It controls, for example, which system-defined performance class data fields are recorded or which system-defined fields should be excluded from being recorded.

Note: If you create your own monitoring control table, it has to be reassembled using the macros in sublibrary `PRD1.BASE`.

Three sample monitoring control tables are also provided in the `z/VSE` sublibrary `PRD1.BASE` :

- For terminal-owning regions (TORs) - `DFHMCTT$`
- For application-owning regions (AORs) - `DFHMCTA$`
- For file-owning regions (FORs) - `DFHMCTF$`

These samples show how to use the EXCLUDE and INCLUDE operands to reduce the size of the performance class record.

See also the [“Statistical changes with CICS TS for z/VSE 2.1”](#) on page 275.

The DFHMCT TYPE=RECORD Macro

You can specify some new values on the INCLUDE and EXCLUDE operands of the DFHMCT TYPE=RECORD macro. These values allow you to include or exclude specific fields from performance-class monitoring records. The new values are shown in the following table. These are all TYPE-A data.

Table 23. New values on INCLUDE and EXCLUDE operands of the DFHMCT TYPE=RECORD introduced with CICS TS for z/VSE 2.1

Group Name	Field Id	Field Name	Description
DFHCHNL	321	PGTOTCCT	Number of CICS requests for channel containers
DFHCHNL	322	PGBRWCCT	Number of STARTBROWSE CONTAINER requests for channel containers
DFHCHNL	323	PGGETCCT	Number of GET CONTAINER requests for channel containers
DFHCHNL	324	PGPUTCCT	Number of PUT 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 commands
DFHCHNL	327	PGPUTCDL	Length of data in the containers of all PUT CONTAINER CHANNEL commands
DFHCHNL	328	PGCRECCT	The number of containers created by PUT CONTAINER requests for channel containers commands.
DFHPROG	306	PCLNKCCT	Number of local LINK requests with CHANNEL option
DFHPROG	307	PCXCLCCT	Number of XCTL requests with CHANNEL option
DFHPROG	309	PCRTNCCT	Number of remote RETURN requests with CHANNEL option
DFHPROG	310	PCRTNCDL	Length of data in the containers of all remote RETURN CHANNEL commands
DFHTASK	65	ICSTACCT	Number of local START requests with CHANNEL option
DFHTASK	345	ICSTACDL	Length of data in the containers of all locally-executed START CHANNEL requests

Table 23. New values on INCLUDE and EXCLUDE operands of the DFHMCT TYPE=RECORD introduced with CICS TS for z/VSE 2.1 (continued)

Group Name	Field Id	Field Name	Description
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

Monitoring utility program

CICS® provides two programs for processing any CICS monitoring data that is written to DMF data sets.

These two programs are:

DFHMNDUP

A utility program that generates a performance dictionary record for use with monitoring data extracted from DMF data sets for printing and formatting.

Note: You have to create a performance dictionary record with the DFHMNDUP utility program shipped with your version of CICS/TS for z/VSE. See also skeleton DFHMOLS in ICCF library 59.

DFH\$MOLS

A print program for CICS monitoring data. DFH\$MOLS is a sample program which you can modify or adapt to your own purposes — it is intended to show how you can code your own monitoring utility program to print CICS monitoring data.

See also skeleton DFHMOLS in ICCF library 59.

Statistical changes with CICS TS for z/VSE 2.1

There have been changes made to certain fields in the statistics report. New fields were added to the intersystem communication (ISC) or interregion communication (IRC) system statistics A14 record, described by DSECT DFHA14DS. The utility program DSFHSTUP has been adapted to show the channel related fields.

Table 24. New fields in A14 record

	Field ID	Description
Program Control	A14ESTPC	The total number of program control requests function shipped across the connection.
	A14ESTPC_CHANNEL	Number of program control link requests, with channels, for function shipping
	A14ESTPC_CHANNEL_SENT	Number of bytes sent on link channel requests
	A14ESTPC_CHANNEL_RCVD	Number of bytes received on link channel requests
Terminal sharing	A14ESTTC_CHANNEL	Number of terminal-sharing channel requests
	A14ESTTC_CHANNEL_SENT	Number of bytes sent on terminal-sharing channel requests
	A14ESTTC_CHANNEL_RCVD	Number of bytes received on terminal-sharing channel requests

Table 24. New fields in A14 record (continued)

	Field ID	Description
Interval control	A14ESTIC_CHANNEL	Number of interval control start requests, with channels, for function shipping
	A14ESTIC_CHANNEL_SENT	Number of bytes sent on start channel requests
	A14ESTIC_CHANNEL_RCVD	Number of bytes received on start channel requests

Statistics utility program (DFHSTUP)

The statistics utility program, DFHSTUP, prepares and prints reports offline, using the CICS® statistics data recorded on the DMF data sets. Always use the version of the DFHSTUP program from the same release of CICS as the data that it is to process.

These are the newly added and renamed fields in the statistic report that were introduced with CICS TS for z/VSE 2.1:

- Renamed "Interval control function shipping requests" to "Interval control (IC) - Total"
- Added "Interval control (IC) - Channel"
- Added "Program control (PC) - Total"
- Added "Program control (PC) - Channel"
- Renamed "Terminal sharing requests" to "Terminal sharing requests - Total"
- Added "Terminal sharing requests - Channel"
- Added "Bytes Sent by Program Channel requests"
- Added "Bytes Received by Program Channel requests"
- Added "Bytes Sent by Interval Channel requests"
- Added "Bytes Received by Interval Channel requests"
- Added "Bytes Sent by Transaction Routing requests"
- Added "Bytes Received by Transaction Routing requests"

The figure below shows a sample statistic report that includes channels and containers.

Interval Report 1 (00:01:00) Collection Date-Time 09/30/2014-07:49:00 Last Reset
00:00:00

Report Date 11/25/2014 Page 78
Applid DBDCCICS Jobname CICSICCF ISC/IRC SYSTEM AND MODE ENTRIES

System Entry

Connection name	:	CBPS
Aids in chain	:	0
Generic aids in chain	:	0
ATIs satisfied by contention losers	:	0
ATIs satisfied by contention winners	:	0
Peak contention losers	:	0
Peak contention winners	:	0
Peak outstanding allocates	:	0
Total number of allocates	:	0
Queued allocates	:	0
Failed link allocates	:	0
Failed allocates due to sessions in use	:	0
Maximum queue time (seconds)	:	0
Allocate queue limit	:	0
Number of QUEUELIMIT allocates rejected	:	0
Number of MAXQTIME allocate queue purges	:	0
Number of MAXQTIME allocates purged	:	0
Number of XZIQUE allocates rejected	:	0
Number of XZIQUE allocate queue purges	:	0
Number of XZIQUE allocates purged	:	0
Total bids sent	:	0
Current bids in progress	:	0
Peak bids in progress	:	0
File control function shipping requests	:	0
Interval control (IC) - Total	:	0
Interval control (IC) - Channel	:	0
Program control (PC) - Total	:	0
Program control (PC) - Channel	:	0
TD function shipping requests	:	0
TS function shipping requests	:	0
DLI function shipping requests	:	0
Terminal sharing requests - Total	:	0
Terminal sharing requests - Channel	:	0
Bytes Sent by Transaction Routing requests	:	0
Bytes Received by Transaction Routing requests	:	0
Bytes Sent by Program Channel requests	:	0
Bytes Received by Program Channel requests	:	0
Bytes Sent by Interval Channel requests	:	0
Bytes Received by Interval Channel requests	:	0

Figure 38. Statistic report example

Chapter 47. CICS-supplied Utility Programs

Note:

- The internal level number of CICS TS for z/VSE 2.2 is "430".
- The internal level number of CICS TS for z/VSE 2.1 is "420".
- The internal level number of CICS TS for VSE/ESA 1.1.1 is "410".

The internal level number is used within the CICS-supplied utility programs.

The CICS TS utility programs require that the parameter OS390 is specified on the EXEC statement. For example: EXEC DFHPD420,OS390.

Changes to the trace formatting utility program, DFHTU430

z/VSE 6.2z/VSE 6.2The trace formatting utility program is renamed to DFHTU430. Always ensure you use the trace program with the correct level number for the release of CICS TS that created the trace data set you are formatting.

Changes to the dump exit routine, DFHPD430

The dump formatting utility program is renamed to DFHPD430. Always ensure you use the dump formatting program with the correct level number for the release of CICS TS that created the dump data set you are formatting.

Changes to the CICS transaction dump utility program, DFHDU430

The transaction dump utility program is renamed to DFHPDU430. Always ensure you use the transaction dump utility program with the correct level number for the release of CICS TS that created the dump data set you are formatting.

Changes to other internally used programs

DFHTT430 is the CICS module used for trace interpretation.

Chapter 48. How to migrate RPG II online applications to CICS TS for z/VSE

Note that this description applies both to CICS TS for z/VSE 2.1 and CICS TS for z/VSE 2.2. Users who already migrated their RPG II online applications on z/VSE 6.1 and performed a FSU to z/VSE 6.2 are not affected.

Initially, RPG online applications were restricted to the CICS/VSE environment and could not be run with CICS Transaction Server for VSE/ESA. This changed with the end of service of CICS/VSE 2.3 in October 2012. An RPG PTF (UK60655) was provided, that allows to run RPG online applications with CICS TS for VSE/ESA and also with CICS TS for z/VSE. Please verify, if there are additional RPG PTFs available.

This topic describes:

- The prerequisites to run RPG online applications with CICS TS for z/VSE
- CICS TS program definitions to run RPG online applications
- Migration aspects

Prerequisites

1. Install the RPG compiler DOS/VS RPG II Release 1.3.0 from the optional product tape. It is installed in sublibrary PRD2.PROD.

Ensure, that PTFs (including PTFs UK60655 and UK97635) are applied. If the z/VSE 6.1 optional product tape is used, all known PTFs at GA are already included.

2. Run the jobs RPGINST and RPGSAMPL, which are provided in ICCF library 59. These jobs install CICS parts, among others the CICS RPG II pre-translator, in a newly created sublibrary PRD2.PRGUII.

CICS TS RPG program definitions

- Define the RPG program in CICS TS with Language=Assembler
- Define the RPG program in CICS TS with Reload=Yes

CSD updates

- Add program RPGIICLN to the CSD:

```
DEFINE PROG(RPGIICLN) GROUP(group) DA(BELOW) LANG(A)
```

- Add transaction CRPG to the CSD:

```
DEFINE TRANS(CRPG)GROUP(group)PROG(RPGIICLN)
```

Transaction security for CRPG has to be allowed.

Users migrating from CICS TS for VSE/ESA to CICS TS for z/VSE

The RPG applications will run unchanged on CICS TS for z/VSE.

Users migrating from CICS/VSE to CICS TS for z/VSE

RPG online applications have to be recompiled and re-linked for CICS TS for z/VSE on your z/VSE 6.1 system. Modify your compile and link jobs:

- Add sublibrary PRD2.RPGII in the LIBDEF SEARCH chain statement
- Add PARM='CICSTS' in the EXEC RPGIAUTO statement:

```
// EXEC RPGIAUTO,SIZE=256K,PARM='CICSTS'
```

- Link-edit including object ILNERI. INLERI is used in place of DFHERI. DFHERI must no longer be included in the link job. ILNERI must be the first object included in linking the phase. For example:

```
PHASE RPGPROG,*  
  INCLUDE ILNERI  
  INCLUDE . . . .
```

A sample compile and link job is provided in ICCF library 2, member C\$\$RPONL. The same job is generated when you are using IUI compile option 8 with SOURCE TYPE 1 (Online Program) and LANGUAGE 5 (RPG II).

Chapter 49. Changes to the EXEC CICS application programming interface

- CICS TS for z/VSE 2.1 introduced the channels and containers API. Refer to [Chapter 33, “CICS API commands for using channels and containers,”](#) on page 191.
- Changes to rounding for ASKTIME and FORMATTIME commands.
- Changes to INQUIRE SYSTEM command.
- New options of the FORMATTIME command.
- New CONVERTTIME command.
- New condition of the FORMATTIME command.

Changes to INQUIRE SYSTEM command

The OSLEVEL option was added to the INQUIRE SYSTEM command. This option returns a 6-byte field that shows the version, release, and modification level of the z/VSE product on which CICS TS for z/VSE is running. For example, z/VSE Version 6 Release 1 Modification Level 0 returns the value 060100.

CONVERTTIME

CONVERTTIME

➤ CONVERTTIME — DATESTRING(*data-area*) — ABSTIME(*data-area*) ➤

Conditions: INVREQ, LENGERR

Description

CONVERTTIME analyzes three different date and time stamp formats which are commonly used on the Internet, and converts them to the ABSTIME (absolute date and time) format.

ABSTIME format gives the time, in packed decimal, since 00:00 on 1 January 1900 (in milliseconds rounded to the nearest hundredth of a second). The FORMATTIME command can be used to change this into other formats.

The architected date and time stamp string formats recognized by the CONVERTTIME command are:

RFC 1123 format

The preferred standard format for date and time stamps for the HTTP protocol, as specified in RFC 1123. An example of a date and time stamp in this format is "Tue, 01 Apr 2003 10:01:02 GMT".

RFC 850 format

An older date and time stamp format for the Internet. An example of a date and time stamp in this format is "Tuesday, 01-Apr-03 10:01:02 GMT".

Important: Because the year has only two digits in this format, CICS® uses the assumption that the years are in the range 1970 to 2069. In the example above, CICS would assume that the date of the document was 1 April 2003. Given the date and time stamp "Thursday, 13-Feb-98 15:30:00 GMT", CICS would assume that the date of the document was 13 February 1998. Be aware of this when coding your application, if you think that you could receive date and time stamps in this format.

ASctime format

A date and time stamp format output from the C ASctime function. An example of a date and time stamp in this format is "Tue Apr 1 10:01:02 2003".

Options

DATESTRING(data-area)

Specifies a 64-character data-area to contain the architected date and time stamp string. You can supply a string in any of the formats recognized by the command, and you do not need to specify which format is used. If the date and time stamp string is in the RFC 1123 format, which is always at GMT, the date and time are converted to local time for the ABSTIME which is returned.

ABSTIME(data-area)

Specifies a data-area to receive the converted date and time stamp in ABSTIME format. For the format of this data-area, see the description of the ASKTIME command. If the date and time stamp was not in a recognized format, no ABSTIME is returned.

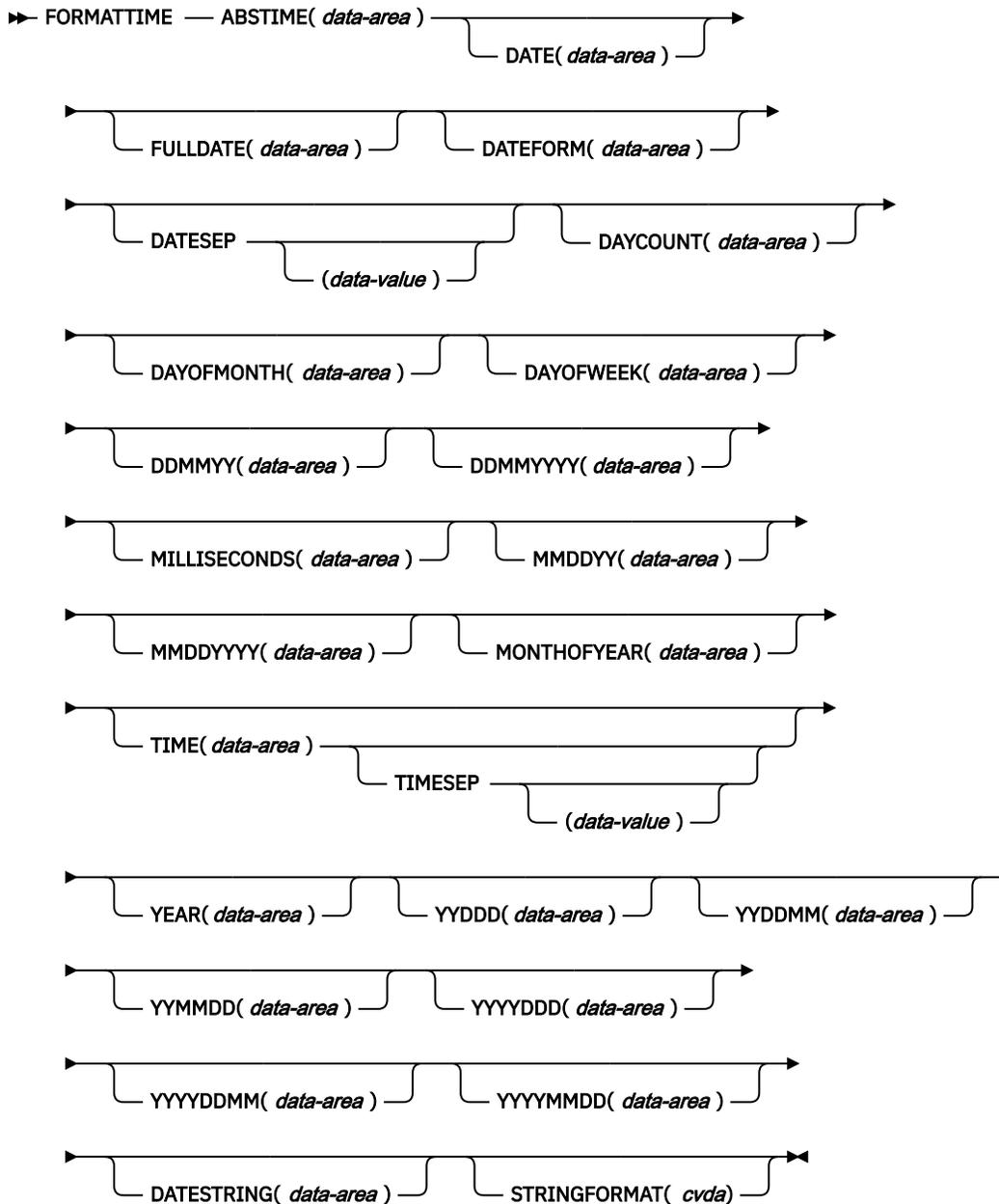
Conditions

INVREQ

occurs for the following conditions. RESP2 values are:

- 1** Format of date and time stamp string not recognized. (This error can be caused by a year value that has more or less than the correct number of digits for the identified format.)
- 2** Invalid time.
- 3** Invalid month.
- 4** Invalid year (includes years before 1900).
- 5** Invalid day name.
- 6** Invalid day number for month and year specified.
- 7** GMT was not stated (required for RFC 1123 and RFC 850 formats).

FORMATTIME



Conditions: INVREQ

Description

FORMATTIME transforms the absolute date and time into any of a variety of formats. Normally, the ABSTIME argument is the value returned by an ASKTIME ABSTIME command.

To obtain an elapsed time in a particular format, the ABSTIME data value can be the difference between two values returned by ASKTIME, and options such as DAYCOUNT(d) and TIME(t) can be specified.

Options

ABSTIME(data-area)

specifies the data value for the time, in packed decimal, since 00:00 hours on 1 January 1900 (in milliseconds rounded to the nearest hundredth of a second) that is to be converted to an alternative format.

The format of the parameter is:

```
COBOL: PIC S9(15) COMP-3
C:      char data_ref[8];
PL/I:   FIXED DEC(15);
ASM:    PL8
```

DATE(data-area)

specifies the variable that is to receive the date in the format specified in the DATFORM system initialization parameter. A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 8-character user field. You should normally use this option only when a date is needed for output purposes. Where a date is needed for analysis, you should request the date in explicit form, for example, using the MMDDYY option.

DATEFORM(data-area)

specifies the format of the installation-defined date. CICS® returns YYMMDD, DDMMYY, or MMDDYY (six characters) according to the DATFORM system initialization parameter.

DATESEP(data-value)

specifies the character to be inserted as the separator between the year and the month, and between the day and the month; or between the year and the day if form YYDDD is specified. If you omit this option, no separator is supplied.

If you omit “data-value”, a slash (/) is assumed as the separator.

DATESTRING(data-area)

specifies the 64-character user field where CICS is to return the architected date and time stamp string in the format specified by the STRINGFORMAT option. If STRINGFORMAT is not specified, the default format provided is the RFC 1123 format (RFC1123).

Note: If you are using the DATESTRING option, run the ASKTIME ABSTIME command first to obtain a value for the ABSTIME option. If the value for the ABSTIME option is from any other source, the architected date and time stamp string which is returned by the FORMATTIME command might be incorrect.

DAYCOUNT(data-area)

returns the number of days since 1 January 1900 (day 1), as a fullword binary number. This is useful if you need to compare the current date with a previous date that has, for example, been stored in a data set.

DAYOFMONTH(data-area)

returns the number of the day in the month as a fullword binary number.

DAYOFWEEK(data-area)

returns the relative day number of the week as a fullword binary number: Sunday=0, Saturday=6. This number can be converted to a textual form of day in any language.

DDMMYY(data-area)

specifies the 8-character user field where CICS is to return the date, in day/month/year format (for example, 21/10/98). A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 8-character user field.

DDMMYYYY(data-area)

specifies the 10-character user field where CICS is to return the date, in day/month/year format (for example 17/06/1995). A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 10-character user field.

FULLDATE(data-area)

specifies the 10-character user field where CICS is to return the date, in the format specified in the DATFORM system initialization parameter, with the year expanded to 4 digits. A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 10-character user field. You should normally use this option only when a date is needed for output purposes. Where a date is needed for analysis, you should request the date in explicit form, for example, using the MMDDYYYY option.

MILLISECONDS(data-area)

Returns the number of milliseconds in the current second specified by ABSTIME, as a fullword binary integer in the range 0 - 999.

MMDDYY(data-area)

specifies the 8-character user field in which CICS is to return the date, in month/day/year format (for example, 10/21/95). A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 8-character user field.

MMDDYYYY(data-area)

specifies the 10-character user field where CICS is to return the date, in month/day/year format (for example 11/21/1995). A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 10-character user field.

MONTHOFYEAR(data-area)

“data-area” is set to the relative month number of the year as a fullword binary number (January=1, December=12). You can convert this number, in your application program, to the name of the month in any language.

STRINGFORMAT(cvda)

specifies the format for the architected date and time stamp string returned in DATESTRING. The only CVDA value available at present is:

RFC1123

specifies the RFC 1123 format, which is suitable for use on the Internet. This date and time stamp string contains the day, date, and 24-hour clock time at GMT, for example "Tue, 01 Apr 2003 10:01:02 GMT".

TIME(data-area)

“data-area” is set as an 8-character field to the current 24-hour clock time in the form hh:mm:ss, where the separator is specified by the TIMESEP option.

TIMESEP(data-value)

specifies the character to be used as the separator in the returned time. If you omit this option, no separator is assumed and six bytes are returned in an 8-character field. If you omit the “data-value”, a colon (:) is used as a separator.

YEAR(data-area)

specifies the full 4-figure number of the year as a fullword binary number (for example, 1995, 2001).

YYDDD(data-area)

specifies the 6-character user field where CICS is to return the date, in year/day format (for example, 95/301). A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 6-character user field.

YYDDMM(data-area)

specifies the 8-character user field where CICS is to return the date, in year/day/month format (for example, 95/30/10). A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 8-character user field.

YYMMDD(data-area)

specifies the 8-character user field where CICS is to return the date, in year/month/day format (for example, 95/10/21). A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 8-character user field.

YYYYDDD(data-area)

specifies the 8-character user field where CICS is to return the date, in year/day format (for example 1995/200). A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 8-character user field.

YYYYDDMM(data-area)

specifies the 10-character user field where CICS is to return the date, in year/day/month format (for example 1995/21/06). A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 10-character user field.

YYYYMMDD(data-area)

specifies the 10-character user field where CICS is to return the date, in year/month/day format (for example 1995/06/21). A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 10-character user field.

Conditions**INVREQ**

RESP2 values:

1

The ABSTIME value is less than zero or not in packed-decimal format.

Default action: terminate the task abnormally.

Examples

The following example shows the effect of some of the options of the command. Let “utime” contain the value 002837962864828 in milliseconds.

```
EXEC CICS ASKTIME ABSTIME(utime)
EXEC CICS FORMATIME ABSTIME(utime)
          DATESEP('-') DDMMYY(date)
          TIME(time) TIMESEP
```

This gives the values 06-12-89 for “date” and 19:01:05 for “time”.

WAIT JOURNALNUM

WAIT JOURNALNUM

➔ WAIT JOURNALNUM(*data-value*)

Conditions: INVREQ, IOERR, JIDERR, NOTAUTH, NOTOPEN

Description

WAIT JOURNALNUM synchronizes the task with the output of one or more journal records that have been created but whose output has been deferred; that is, with asynchronous journal output requests.

The journal records may already be written out from the journal buffer area to the corresponding system logger log stream, or the system logger output operation may be in progress. If the log stream output operation has already been completed, control returns immediately to the requesting task; if not, the requesting task waits until the operation has been completed. If STARTIO is specified, output is initiated immediately.

If the requesting program has made a succession of successful asynchronous output requests to the same journal, it is necessary to synchronize on only the last of these requests to ensure that all of the journal records have reached auxiliary storage. This may be done either by issuing a stand-alone WAIT JOURNALNUM command, or by making the last output command itself synchronous (by specifying the WAIT option in the WRITE JOURNALNUM command).

Options

JOURNALNUM(data-value)

Specifies the journal identifier as a halfword numeric value in the range 1–99. Journal numbers correspond to journal names as follows:

- Journal numbers in the range 2–99 refer to user journals named DFHJ02 through DFHJ99.
- Journal number 1 refers to the system log.

REQID(data-value)

Specifies a fullword binary variable set to a number that refers to the journal record that has been created but possibly not yet written out.

If REQID is not specified, the task is synchronized with the output of the last record created for the journal specified by JOURNALNUM.

STARTIO

Specifies that output of the journal record is to be initiated immediately.

Conditions

INVREQ

Occurs if a WAIT JOURNALNUM command is issued before any WRITE JOURNALNUM command has been issued in the same task.

Default action: terminate the task abnormally.

IOERR

Occurs if the physical output of a journal record was not accomplished because of an unrecoverable I/O error.

Default action: terminate the task abnormally.

JIDERR

Occurs if the specified journal identifier does not exist in the journal control table (JCT).

Default action: terminate the task abnormally.

NOAUTH

Occurs when a resource security check has failed on JOURNALNUM(data-value).

Default action: terminate the task abnormally.

NOTOPEN

Occurs if the WAIT JOURNALNUM command cannot be satisfied because the specified journal is not open.

Default action: terminate the task abnormally.

Examples

The following example shows how to request synchronization with the output of a journal record:

```
EXEC CICS WAIT JOURNALNUM(4)  
      REQID(ENTRYID)
```

Part 11. CICS System Initialization Changes

Chapter 50. CICS system initialization parameters

This section describes the CICS system initialization process; describes the DFHSIT macro and its parameters, and tells you how to supply initialization parameters to CICS.

System Initialization Parameters - New with CICS TS for z/VSE 2.2

This topic summarizes the new CICS TS for z/VSE 2.2 system initialization parameters. If you are using the default values you have no impact when you are migrating from an earlier release of CICS.

The table below shows the new system initialization parameters.

Keywords	Operands	Explanation
MAXSOCKETS	{ 512 number}	The MAXSOCKETS system initialization parameter specifies the maximum number of IP sockets that can be managed by the CICS sockets domain. The maximum number of sockets must be greater than the maximum number of sockets used by CICS including the number of TCPIP SERVICE resources in service. 1 <= number <=512. Default is 512.

System Initialization Parameters - New with CICS TS for z/VSE 2.1

This topic summarizes the new CICS TS for z/VSE system initialization parameters. If you are using the default values you have no impact when you are migrating from an earlier release of CICS.

The table below shows the new system initialization parameters.

Keywords	Operands	Explanation
CLINTCP	{ 437 codepage}	Specifies the default client code page to be used by the DFHCNV data conversion table but only if the CLINTCP parameter in the DFHCNV macro is set to SYSDEF
LOCALCCSID	{ 037 CCSID}	Specifies the default CCSID for the local region. The CCSID is a value of up to 8 characters. If CCSID value is not specified, the default LOCALCCSID is set to 037.
SRVERCP	{ 037 CCSID}	Specifies the default server code page to be used by the DFHCNV data conversion table but only if the SRVERCP parameter in the DFHCNV macro is set to SYSDEF.

Specifying system initialization parameters

The primary method of providing system initialization parameters is with a system initialization table (SIT). The parameters of the SIT, which you assemble as a load table, supply the system initialization program with most of the information necessary to initialize the system to suit your unique environment. You can generate more than one SIT, and (at the time of system initialization) select the one that is appropriate to your needs.

SIT Parameters

You can also specify other system initialization parameters, which cannot be coded in the SIT. You specify which SIT you want, and other system initialization parameters (with a few exceptions), using any of the following methods:

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

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

The information defined by system initialization parameters can be grouped into three categories:

1. Information used to initialize and control CICS system functions (for example, information such as the dynamic storage area limits and the region exit time interval)
2. Module suffixes used to load the user-specified version of the CICS modules (for example, DFHDBPxx) and tables (for example, DFHJCTxx)
3. Special information used to control the initialization process

The syntax of the various DFHSIT macro parameters is listed in [Table 27 on page 294](#). Except for those parameters marked "**SIT macro only**", all the system initialization parameters can be provided at run time, although there are restrictions in some cases. The restrictions are explained at the end of the description of the system initialization parameter to which they apply. See the CHKSTRM parameter in the [CICS System Definition Guide](#) for an example of such a restriction.

There are some other CICS system initialization parameters (and options of the parameters in [Table 27 on page 294](#)) that you cannot define in the DFHSIT macro. (See "[The system initialization parameter descriptions](#)" on page 316.) The parameters that you cannot define in the DFHSIT macro are shown in "[Parameters that you cannot code in the DFHSIT macro](#)" on page 301.

System initialization keywords grouped by function: For a list of all the system initialization keywords grouped by their functional area, see [CICS System Definition Guide](#). This ensures that you do not miss coding an important parameter relating to a particular CICS function. For details of how to code a parameter, you still have to refer to the parameter descriptions that are listed alphabetically in this section.

Migration considerations

If you have existing system initialization tables, you must modify them. Remove all obsolete parameters, and specify the required values for new or changed parameters if you want to run with other than the defaults. When you have made the necessary changes, reassemble the tables using the CICS Transaction Server for z/VSE macro libraries.

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

To avoid generating specific system initialization tables for each CICS system, a simple solution is to let CICS load the default, unsuffixed table (DFHSIT) at start-up, and supply the system initialization parameters for each CICS system in a SYSIPT data set.

The DFHSIT macro parameters

DFHSIT	[TYPE={CSECT DSECT}]
	[,ADI={30 number}]
	[,AIEXIT={DFHZATDX DFHZATDY name}]
	[,AILDELAY={@ hhmmss}]

Table 27. The DFHSIT macro parameters (continued)	
	[,AIQMAX={ <u>100</u> number}]
	[,AIRDELAY={ <u>700</u> hhmmss}]
	[,AKPFREQ={ <u>200</u> number}]
	[,APPLID={({ <u>DBDCCICS</u> name1},{name2})}]
	[,AUTCONN={ <u>0</u> hhmmss}]
	[,AUXTR={ <u>OFF</u> ON}]
	[,AUXTRSW={ <u>NO</u> ALL NEXT}]
	[,BMS={({ <u>MINIMUM</u> STANDARD <u>FULL</u> },{COLD})}]
	[,{ <u>UNALIGN</u> ALIGN}][,{ <u>DDS</u> NODDS}]]]
	[,CLINTCP={ <u>437</u> codepage}]
	[,CLSDSTP={ <u>NOTIFY</u> NONOTIFY}]
	[,CLT=xx]
	[,CMDPROT={ <u>YES</u> NO}]
	[,CMDSEC={ <u>ASIS</u> ALWAYS}]
	[,CONFDATA={ <u>SHOW</u> HIDETC}]
	[,CONFTXT={ <u>NO</u> YES}]
	[,CSDACC={ <u>READWRITE</u> READONLY}]
	[,CSDBKUP={ <u>STATIC</u> DYNAMIC}]
	[,CSDBUFND=number]
	[,CSDBUFNI=number]
	[,CSDFRLOG=number]
	[,CSDJID={ <u>NO</u> number}]
	[,CSDLSRNO={ <u>1</u> number NONE NO}]
	[,CSDRECOV={ <u>NONE</u> ALL BACKOUTONLY}]
	[,CSDSTRNO={ <u>2</u> number}]
	[,CWAKEY={ <u>USER</u> CICS}]
	[,DATFORM={ <u>MMDDYY</u> DDMMYY YYMMDD}]
	[,DBP={ <u>1</u> <u>2</u> xx YES}] (Must specify in the SIT macro)
	[,DBUFSZ={({ <u>500</u> number})}]
	[,DCT={({YES xx <u>NO</u> })}]
	[,DFLTUSER={ <u>CICSUSER</u> userid}]
	[,DIP={ <u>NO</u> YES}]
	[,DISMACP={ <u>YES</u> NO}]

Table 27. The DFHSIT macro parameters (continued)	
	[{,DLI DL1}={NO YES REMOTE},{COLD}]
	[,DLIOER={ABEND CONTINUE}]
	[,DOCCODEPAGE={037 codepage}]
	[,DSALIM={5M number}]
	[,DSHIPIDL={020000 hhmmss}]
	[,DSHIPINT={120000 hhmmss}]
	[,DTRPGM={DFHDYP program-name}]
	[,DTRTRAN={CRTX name NO}]
	[,DUMP={YES NO}]
	[,DUMPDS={AUTO A B}]
	[,DUMPSW={NO NEXT}]
	[,EDSALIM={20M number}]
	[,ENCRYPTION={STRONG SSLV3}]
	[,EODI={E0 xx}]
	[,ESMEXITS={NOINSTLN INSTLN}] (<i>SIT macro only</i>)
	[,FCT={YES xx NO}]
	[,FEPI={NO YES}]
	[,FLDSEP={'_____' 'xxx'}]
	[,FLDSTRT={'_' 'x'}]
	[,FSSTAFF={YES NO}]
	[,FTIMEOUT={30 nn}]
	[,GMTEXT={'WELCOME TO CICS' 'text'}]
	[,GMTRAN={CSGM CESN name}]
	[,GNTRAN={CESF transaction-id}]
	[,GRPLIST={DFHLIST name (name[,name2][,name3][,name4])}]
	[,ICP=COLD]
	[,ICV={1000 number}]
	[,ICVR={5000 number}]
	[,ICVTSD={500 number}]
	[,INITPARM=(pgmname_1='parmstring_1' [, ...,pgmname_n='parmstring_n'])]
	[,INTTR={ON OFF}]
	[,IRCSTRT={NO YES}]
	[,ISC={NO YES}]

Table 27. The DFHSIT macro parameters (continued)	
	[,JCT={YES xx NO}]
	[,KEYFILE=name]
	[,LEVSE={YES NO}]
	[,LGNMSG={NO YES}]
	[,LOCALCCSID={037 CCSID}]
	[,MAXSOCKETS={512 number}]
	[,MCT={NO YES xx}]
	[,MN={OFF ON}]
	[,MNCONV={NO YES}]
	[,MNEXC={OFF ON}]
	[,MNFREQ={0 hhmmss}]
	[,MNPER={OFF ON}]
	[,MNSYNC={NO YES}]
	[,MNTIME={GMT LOCAL}]
	[,MROBTCH={1 number}]
	[,MROFSE={ NO YES}]
	[,MROLRM={NO YES}]
	[,MSGCASE={MIXED UPPER}]
	[,MSGLVL={1 0}]
	[,MXT={5 number}]
	[,NATLANG=(E,x,y,z,...)]
	[,OPERTIM={120 number}]
	[,PARMERR={INTERACT IGNORE ABEND}]
	[,PDI={30 number}]
	[,PGAICTLG={MODIFY NONE ALL}]
	[,PGAIXIT={DFHPGADX name}]
	[,PGAIPGM={INACTIVE ACTIVE}]
	[,PGCHAIN=character(s)]
	[,PGCOPY=character(s)]
	[,PGPURGE=character(s)]
	[,PGRET=character(s)]
	[,PLTPI={NO xx YES}]
	[,PLTPISEC={NONE CMDSEC RESSEC ALL}]
	[,PLTPIUSR=userid]
	[,PLTSD={NO xx YES}]

Table 27. The DFHSIT macro parameters (continued)	
	[,PRGDLAY={0 hhmm}]
	[,PRINT={NO YES PA1 PA2 PA3}]
	[,PRTYAGE={32768 number}]
	[,PSDINT={0 hhmmss}]
	[,PVDELAY={30 number}]
	[,RAMAX={256 number}]
	[,RAPOOL={ 50 value1 (value1,value2)}]
	[,RENTPGM={PROTECT NOPROTECT}]
	[,RESP={FME RRN}]
	[,RESSEC={ASIS ALWAYS}]
	[,RMTRAN={({CSGM name1 [, {CSGM name2}]})]
	[,RUWAPool={NO YES}]
	[,SEC={YES NO}]
	[,SECPRFX={NO YES}]
	[,SKRxxxx='page-retrieval-command']
	[,SNSCOPE={NONE CICS VSEIMAGE}]
	[,SPCTR={({1,2 1[,2][,3]} ALL OFF}]
	[,SPOOL={NO YES}]
	[,SRT={YES xx NO}]
	[,SRVERCP={037 CCSID}]
	[,SSLDELAY={600 number}]
	[,START={AUTO COLD STANDBY}]
	[,STARTER={NO YES}] (SIT macro only)
	[,STATRCD={OFF ON}]
	[,STGPROT={NO YES}]
	[,STGRVCY={NO YES}]
	[,STNTR={1 (1[,2][,3]} ALL OFF}]
	[,SUFFIX=xx] (SIT macro only)
	[,SVA={NO YES{}}]
	[,SYDUMAX={999 number}]
	[,SYSIDNT={CICS name}]
	[,SYSTR={ON OFF}]
	[,TAKEOVR={MANUAL AUTO COMMAND}]

Table 27. The DFHSIT macro parameters (continued)	
	[,TBEXITS={nm1},nm2},nm3},nm4},nm5}]
	[,TCP={YES NO}]
	[,TCPIP={NO YES}]
	[,TCSACTN={NONE UNBIND FORCE}]
	[,TCSWAIT={4 number NO NONE 0}]
	[,TCT={YES xx NO}]
	[,TCTUAKEY={USER CICS}]
	[,TCTUALOC={BELOW ANY}]
	[,TD={({3 number1},{3 number2})}]
	[,TRAP={OFF ON}]
	[,TRDUMAX={999 number}]
	[,TRTABSZ={16 number}]
	[,TRTRANSZ={40 number}]
	[,TRTRANTY={TRAN ALL}]
	[,TS={({COLD},{0 3 value-1},{3 value-2})}]
	[,TMSGSET={4 number}]
	[,TST={NO YES xx}]
	[,USERTR={ON OFF}]
	[,USRDELAY={30 number}]
	[,VTAM={YES NO}]
	[,VTPREFIX={\ character}]
	[,WEBDELAY={5 time_out,60 keep_time}]
	[,WRKAREA={512 number}]
	[,XAPPC={NO YES}]
	[,XCMD={NO name YES}]
	[,XDCT={NO name YES}]
	[,XFCT={NO name YES}]
	[,XJCT={NO name YES}]
	[,XLT={NO xx YES}]
	[,XPCT={NO name YES}]
	[,XPPT={NO name YES}]
	[,XPSB={NO name YES}]
	[,XRF={NO YES}]
	[,XRFSOFF={NOFORCE FORCE}]

Table 27. The DFHSIT macro parameters (continued)	
	[,XRFSTME={ <u>5</u> number}]
	[,XRFTODI={ <u>30</u> number}]
	[,XSWITCH=({@ 1–254})[,{,?????? progrname}][,{A B})]
	[,XTRAN={YES name NO}]
	[,XTST={NO name YES}]
	[,XUSER={YES NO}]
	You must terminate your macro parameters with the following END statement.
END	DFHSITBA

Creating a system initialization table

Decide which parameters you want to use. List them, together with the appropriate setting, in a file named DFHSITxx. The suffix xx can be any two characters that you want (apart from the characters reserved solely for use by CICS), and identifies the file as your own SIT. If you do not specify a two-character suffix, CICS uses the default system initialization table, DFHSIT1\$.

See CICS Resource Definition Guide which summarizes the groups of system initialization parameters needed for different CICS functions.

If you specify more than one value for a system initialization parameter, separate the operand values by commas and enclose them all in parentheses. If you do not specify a parameter as a system initialization override, the system uses the value supplied in whichever SIT you specify.

End your system initialization table with an

```
END DFHSITBA
```

statement.

“The system initialization parameter descriptions” on page 316 describes all the system initialization parameters on an individual basis.

There are some parameters that cannot be coded in the DFHSIT macro. These are summarized in “Parameters that you cannot code in the DFHSIT macro” on page 301.

Coding more than one system initialization table

You can code several SITs, each reflecting different parameter combinations and settings, as long as you remember to give each DFHSITxx a different two-character suffix. CICS uses only one of these tables during initialization. Choose the SIT that best suits your needs at the time, and code its suffix on the SIT system initialization parameter.

Assemble your chosen SIT and link-edit it into one of the sublibraries defined in the LIBDEF statement for your CICS startup job stream.

CICS-supplied system initialization tables

The z/VSE sublibrary, PRD1.BASE, contains the sample system initialization tables, DFHSIT6\$ and DFHSIT\$\$\$. DFHSIT6\$ contains all the SIT values needed for a standard CICS system. DFHSIT\$\$\$ contains all the default values for the system initialization parameters.

System initialization tables supplied with z/VSE

z/VSE supplies two sample system initialization tables. DFHSITSP is provided for a standard CICS system, and DFHSITC2 is provided for a second predefined CICS system. There are listings for both these tables in the [z/VSE Planning, SC34-2681](#) publication.

Assembling the system initialization table

When you have defined the parameters you need, add your system initialization table to your working CICS system by assembling the DFHSIT macro. The macro assembly process produces the linkage-editor control statements needed to link-edit your table into your CICS (or private) sublibrary.

When you have assembled your SIT, make it known to CICS using one of the following methods:

- Including your sublibrary in the LIBDEF statement of the CICS startup job stream.
- Coding the suffix of your system initialization table on the SIT system initialization override parameter.
- Including the suffix of chosen system initialization table on the PARM parameter of the EXEC DFHSIP statement. [Figure 39 on page 306](#) gives an example of using the PARM parameter to specify a SIT with the suffix 1\$.

The samples DFHSITSP and DFHSITC2 are located in ICCF Library 59.

For information about assembling and link-editing CICS control tables, see [CICS Resource Definition Guide](#).

Examples of the syntax notation for describing CICS macros are given in the [CICS Resource Definition Guide](#).

Assembler errors from duplicated or undefined keywords

If you duplicate a keyword when using VSE 2.4 and the High Level Assembler, the following error messages are produced:

```
ASMA018S *** ERROR *** DUPLICATE KEYWORD IN MACRO CALL;
                      LAST VALUE IS USED - DFHSI/keyword
```

If you use an undefined keyword, the following error message is produced:

```
ASMA017W ** WARNING ** UNDEFINED KEYWORD PARAMETER; DEFAULT TO POSITIONAL,
          INCLUDING KEYWORD - DFHSI/keyword
```

However, be aware that because of work space limitations, there is a limit to the number of undefined keyword errors that the assembler can generate. This means that if your SIT contains more undefined keywords than the assembler can generate messages for, the excess errors will not be flagged until a second or even later assembly and, as you correct the currently flagged errors, other (previously unflagged) errors may appear during re-assembly.

Parameters that you cannot code in the DFHSIT macro

There are some CICS system initialization parameters that you cannot define in the DFHSIT macro. These are:

- CDSASZE={OK|number}
- CHKSTRM={CURRENT|NONE}
- CHKSTSK={ALL|CURRENT|NONE}
- ECDSASZE={OK|number}
- ERDSASZE={OK|number}
- ESDSASZE={OK|number}
- EUDSASZE={OK|number}

SIT Parameters

- JSTATUS=RESET
- NEWSIT={YES|NO}
- PRVMOD={name|(name,name...name)}
- RDSASZE={OK|number}
- SDSASZE={OK|number}
- SIT=xx
- SPCTRxx={(1[,2][,3])|ALL|OFF}
- START=LOGTERM
- START=(option,ALL)
- STNTRxx={(1[,2][,3])|ALL|OFF}
- UDSASZE={OK|number}

These parameters can only be supplied at CICS startup time using any of the following methods:

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

For information about coding system initialization parameters in PARM, SYSIPT or at the console, see [“System initialization control keywords” on page 303](#).

Overriding SIT parameters at system startup

There may be times when you need to change some of the parameters specified in your SIT, but don't want to have to modify your SIT and reassemble it; or you may need to supply parameters that cannot be coded in the DFHSIT macro (as listed in [“Parameters that you cannot code in the DFHSIT macro” on page 301](#)). To do this you supply system initialization parameters as **overrides** using the:

- PARM parameter on the EXEC DFHSIP statement
- SYSIPT data set
- Operator console.

You can use one, or a combination of the above methods. However, parameter manager domain processes these three sources in the following strict sequence:

1. The PARM parameter
2. The SYSIPT data set (but only if SYSIPT is coded in the PARM parameter)
3. The console (but only if CONSOLE is coded in either the PARM parameter or in the SYSIPT data set.

Note: If you supply duplicate system initialization parameters, either through the same or a different medium, CICS takes the **last** one that it reads. For example, if you specify DCT=1\$ in the PARM parameter, DCT=2\$ in the SYSIPT data set, and finally enter DCT=3\$ through the console, CICS loads DFHDCT3\$.

You complete the override process with the \$END statement.

During startup, CICS uses your chosen system initialization table and any system initialization parameter overrides you have specified to determine the CICS functions you require.

The CICS parameter manager domain

In addition to loading the system initialization table at the start of initialization, and reading any other parameters from PARM, SYSIPT, or the system console, the parameter manager domain is responsible for the management of the SIT. With the exception of the application domain (AP) which uses the SIT directly, the parameter manager domain passes system initialization parameters to the other CICS domains on request. The domain initialization process is as follows:

Query the type of startup

With the exception of the trace domain, each domain asks the parameter manager for the type of startup, cold or warm. (For this purpose, the parameter manager domain treats an emergency restart as a warm start.)

Startup is cold

If startup is cold, domains do not read their domain records from the catalogs. Instead, they request system initialization parameters from the parameter manager domain. Because it is a cold start, the parameter manager domain returns all system initialization parameters from the SIT, modified by any overrides.

Startup is warm

If startup is warm, domains try to perform a warm start by reading their domain records from the catalogs:

- If they succeed in reading their status records, domains perform a warm start. Where applicable, they also request system initialization parameters from the parameter manager domain. Because it is a warm start, the parameter manager domain returns only those system initialization parameters supplied as overrides via PARM, SYSIPT, or the system console.
- If they fail for any reason to read their status records, domains perform a cold start. They do this either by requesting **all** system initialization parameters from the parameter manager domain, or by using system default values if the domain does not have any system initialization parameters.

NEWSIT or new suffix

Although a START=AUTO may resolve to a warm start, parameter manager enforces most system initialization parameters if:

- You specify NEWSIT=YES as a system initialization parameter in PARM, SYSIPT, or through the console.
- You specify a different SIT suffix from the previous run of CICS. Parameter manager saves the suffix from each run in the global catalog, and, if it detects a new suffix, it forces the NEWSIT=YES option.

For details of the parameters that are ignored when you specify NEWSIT=YES, see the NEWSIT parameter description . [“NEWSIT={YES|NO}”](#) on page 343

Note: The trace domain is an exception to the above rules in that it always cold starts. Trace does not save its status at CICS shutdown like the other domains, and regardless of the type of startup, it requests **all** of its system initialization parameters from the parameter manager domain.

System initialization control keywords

There are three special control keywords that you can use at start up to control how CICS reads in system initialization parameters. These are:

- SYSIN (abbreviated to SI)
- CONSOLE (abbreviated to CN)
- .END

The SYSIN (SI) keyword

Code the SYSIN keyword to tell CICS to read system initialization parameters from the SYSIPT data set. You can code SYSIN (or SI) only on the PARM parameter of the EXEC DFHSIP statement. The keyword can appear once only and must be at the end of the PARM parameter. CICS does not read the SYSIPT data stream until it has finished scanning all of the PARM parameter, or until it reaches an \$END keyword before the end of the PARM parameter. An example of coding the SYSIN keyword is:

```
// EXEC DFHSIP,SIZE=DFHSIP,PARM=' SI ',0S390
```

The CONSOLE (CN) keyword

Use the CONSOLE (CN) keyword to tell CICS to read initialization parameters from the console. CICS prompts you with message DFHPA1104 when it is ready to read parameters from the console.

You can code the CONSOLE (or CN) keyword in the PARM parameter of the EXEC DFHSIP statement, or in the SYSIPT data stream. The keyword can appear anywhere in the PARM parameter or SYSIPT data stream, but you must code it in one place only.

If you code CONSOLE on the PARM parameter and the PARM parameter also contains the SYSIN keyword CICS does not begin reading parameters from the console until it has finished reading and processing the SYSIPT data stream. Similarly, if you code CONSOLE in the SYSIPT data stream, CICS does not begin reading parameters from the console until it has finished reading and processing the SYSIPT data stream.

In general, after a given CICS system has become the production system, you should perform startup with as little operator intervention as possible (that is, use PARM or SYSIPT, and **not** the console). Better still, use the SIT only, without any overrides. An example of coding the CONSOLE keyword is as follows:

```
// EXEC DFHSIP,SIZE=DFHSIP,PARM='CN',OS390
```

The .END keyword

The meaning of the .END keyword varies, depending on its context.

Note that \$END is also recognized for compatibility with earlier versions of CICS.

If you are using the PARM parameter, the use of the .END keyword is optional. If you omit it, CICS assumes it to be at the end of the PARM parameter. If you code .END in the PARM parameter it can have one of two meanings:

1. If you also code one, or both, of the other control keywords (CONSOLE and/or SYSIN) .END denotes the end of the PARM parameter only. For example:

```
// EXEC DFHSIP,SIZE=DFHSIP,PARM='SI,CN,SIT=6$, .END',OS390
```

2. If you code .END as the only control keyword in the PARM parameter, it denotes the end of all system initialization parameters, and CICS begins the initialization process. For example:

```
// EXEC DFHSIP,SIZE=DFHSIP,PARM='SIT=6$, .END',OS390
```

If .END is not the last entry in the PARM parameter, CICS truncates the PARM parameter and the parameters following the .END keyword are ignored.

If you are using the SYSIPT data stream, the use of the .END keyword is optional. If you omit it, CICS assumes it to be at the end of SYSIPT. If you code .END in the SYSIPT data stream its meaning depends on your use of the CONSOLE keyword, as follows:

- If you code the CONSOLE control keyword in the PARM parameter or in the SYSIPT data stream, .END denotes the end of the SYSIPT data stream only.
- If you do not code the CONSOLE control keyword in the PARM parameter or in the SYSIPT data stream, .END denotes the end of all CICS system initialization parameters, and CICS begins the initialization process.

If you code .END, and it is not the last entry in the SYSIPT data stream, or not at the end of a SYSIPT record, CICS initialization parameters following the .END are ignored. To avoid accidental loss of initialization parameters, ensure that the .END keyword is on the last record in the SYSIPT data stream, and that it is the only entry on that line. (However, if you want to remove some system initialization parameters from a particular run of CICS, you could position them after the .END statement just for that run.)

The following example shows the use of .END in a SYSIPT data set:

```
// EXEC DFHSIP,SIZE=DFHSIP,PARM='SI',OS390
* CICS system initialization parameters
```

```
SIT=6$, START=COLD,
AUXTR=OFF, CSDLRNO=15
:
.END
/*
```

If you are using the `CONSOLE` keyword, the meaning of `.END` through the console depends on whether you are entering new parameters or entering corrections. The two meanings are as follows:

1. If you are keying new parameters in response to message DFHPA1104, `.END` terminates parameter reading, and CICS starts initialization according to the SIT it has loaded, but modified by any system initialization parameters you have supplied. Until you enter the `.END` control keyword, CICS continues to prompt you for system initialization parameters.
2. If you have coded `PARMERR=INTERACT`, and CICS detects a parameter error, either in the keyword or in the value that you have assigned to it, CICS prompts you to correct the error with message DFHPA1912 or DFHPA1915. If you enter the correct keyword or value, initialization resumes with CICS continuing to process either the `PARM` parameter, the `SYSIPT` data set, or prompting for more system initialization parameters through the console, depending on where CICS detected the error. If you cannot correct the error, but want CICS to continue with the initialization process, you can enter `.END` to end the error correction phase.

Processing the PARM parameter

If you omit the `PARM` parameter from the `EXEC DFHSIP` statement CICS assumes that there are no SIT override or other initialization parameters, and attempts to load an unsuffixed version of `DFHSIT`. As a general rule, this is unlikely to be your intention, so the `PARM` parameter should at least specify the suffix of your system initialization table, using the `SIT` system initialization parameter.

Alternatively, you can code the special `SYSIN` keyword as the only `PARM` parameter, and supply the suffix of your SIT and the other system initialization parameters from the `SYSIPT` data set.

CICS scans the `PARM` string looking for a `SIT=` parameter, any of the special control keywords, or any system initialization parameters, and proceeds as follows:

- If CICS finds a `SIT=` parameter but no `SYSIN` keyword, CICS tries to load the SIT as soon as it has finished scanning the `PARM` parameter. Processing any CICS system initialization parameters that are also present in the `PARM` parameter takes place only after the SIT has been loaded.
- If CICS finds a `SIT=` parameter **and** a `SYSIN` keyword, CICS does not try to load the SIT until it has also finished scanning the `SYSIPT` data set. In this case, loading the SIT is deferred because there can be other `SIT=` parameters coded in the `SYSIPT` data set that override the one in the `PARM` parameter.

Processing any system initialization parameters that are also present in the `PARM` parameter takes place only after the SIT has been loaded.

Rules for coding PARM parameters

The rules for coding the `PARM` parameter on an `EXEC` job control statement are described fully in the [z/VSE System Control Statements, SC34-2679](#) publication.

Briefly, the maximum number of characters you can code on each `PARM` parameter is 100, excluding the opening and closing apostrophes. All CICS system initialization parameters must be separated by a comma, and the separating commas are included in the 100 character limit. You can code the `PARM` parameter up to three times on one `EXEC` statement. The syntax rules above apply to each `PARM` parameter separately. Because of this limiting factor, you might prefer to limit the use of the `PARM` parameter to specify the `SYSIN` control keyword only.

Coding the PARM parameter over two lines

If you need to continue your `PARM` parameter on a second line, indicate the continuation with a nonblank character in column 72, and continue in column 16 on the next line. Do not leave any blank columns between your SIT overrides and the continuation character in column 72; your overrides must be entered up to, and including, column 71.

Figure 39 on page 306 gives an example of the PARM parameter coded over two lines.

```
0-----1-----2-----3-----4-----5-----6-----7--
// EXEC DFHSIP,SIZE=DFHSIP,PARM='SIT=1$,DCT=1B,GRPLIST=DFHLIST,TCT=5$,T*
S=(,0),SPOOL=YES,START=AUTO',0S390
```

Figure 39. Example of a PARM parameter coded over two lines

Processing the SYSIPT data set

CICS scans the SYSIPT data set looking for a SIT= parameter and any of the special keywords, as well as system initialization parameters.

If CICS finds a SIT= parameter in the SYSIPT data set, it tries to load that SIT, overriding any that was specified in the PARM parameter. If CICS does not find a SIT= parameter in the SYSIPT data set, it tries to load any SIT specified in the PARM parameter.

However, if after scanning the PARM parameter and the SYSIPT data set CICS has not found a SIT= parameter, CICS does one of the following:

1. If you specified CONSOLE in the PARM parameter or in the SYSIPT data set, CICS prompts you with the following message to enter the SIT suffix as the first parameter through the console:

```
DFHPA1921 DBDCCICS PLEASE SPECIFY THE REQUIRED SIT SUFFIX, OR
SPECIFY 'NONE' (UNSUFFIXED).
```

2. If you did not specify CONSOLE, CICS automatically tries to load an unsuffixed SIT module (DFHSIT). If this load fails, CICS issues message DFHPA1106, requesting a SIT suffix in reply to the message.

Note: CICS does not process any system initialization parameters that are coded in the PARM parameter and the SYSIPT data set until after the SIT has been loaded.

Rules for coding parameters in the SYSIPT data set

There are a few simple rules to observe when coding CICS system initialization parameters in the SYSIPT data set. These are:

- Use a comma to separate parameters that are on the same line. The use of a comma at the end of a SYSIPT record is optional.
- You can use an asterisk in column 1 to code comments, or to remove temporarily an initialization parameter from a particular execution of CICS.
- You can also add comments after the parameters on a SYSIPT line, but they must be preceded by at least one blank character.
- The SYSIPT data set is an 80-byte file. Everything that appears in positions 1 through 80 is treated by CICS as input data.
- You can continue, on another line in SYSIPT, parameters that have multiple operands if you make the split immediately after a comma. CICS concatenates the operands, omitting the intervening blanks.
- As a general rule, you cannot split an individual operand between lines. However, in the case of the GMTEXT parameter, you can enter the operand over more than one line up to the maximum of 246 characters. The format of this parameter is:

```
GMTEXT='User's text'
```

You can use apostrophes to punctuate message text, provided that you code *two* successive apostrophes to represent a single apostrophe (as shown in the example above). The apostrophes delimiting the text are mandatory.

- You must take care when coding parameters that use apostrophes, parentheses, or commas as delimiters, because failure to include the correct delimiters is likely to cause unpredictable results.

Processing the console entries

Generally, CICS does not begin to read from the console until it has loaded the SIT and processed any initialization parameters that are coded in the PARM parameter and the SYSIPT data set. CICS accepts system initialization parameters from the console until you terminate the input with .END.

You can specify a SIT= parameter only as the first parameter through the console when prompted by message DFHPA1921, at which point CICS tries to load the specified SIT. If you try to specify a SIT= parameter after CICS has loaded the SIT it is rejected as an error.

Rules for coding parameters at the console

When it is ready to read parameters from the console, CICS displays the following message (where nn is the reply ID):

```
nn DFHPA1104 applid - SPECIFY ALTERNATIVE SIT PARAMETERS, IF ANY,
                        AND THEN TYPE '.END'.
```

You can enter as many initialization parameters as you can get on one line of the console, but you must use a comma to separate parameters. CICS continues to prompt for system initialization parameters with displays of message DFHPA1105 until you terminate console input by entering the .END control keyword.

Entering corrections to parameters at the console

If you have coded PARMERR=INTERACT, and CICS detects a parameter error, either in the keyword or in the value you have assigned to it, CICS prompts you to correct the error with message DFHPA1912 or DFHPA1915:

```
DFHPA1912 applid SIT OVERRIDE 'keyword' IS NOT RECOGNIZED.
                SPECIFY CORRECT SIT OVERRIDE.
```

```
DFHPA1915 applid INVALID DATA HAS BEEN DETECTED FOR SIT OVERRIDE
                'keyword'. RESPECIFY THE OVERRIDE.
```

CICS prompts you to enter corrections to any errors it find in the PARM parameter or the SYSIPT data set **after** it has loaded the SIT, and as each error is detected. This means that if there is an APPLID parameter **following** the parameter that is in error, either in the PARM parameter or in the SYSIPT data set, it is the APPLID coded in the SIT that CICS displays in messages DFHPA1912 and DFHPA1915.

Notes on CICS resource table and module keywords

Table 28 on page 307 shows the system initialization keywords for those CICS resources that:

- Have a suffix option
- Result in a dummy program or table if you code resource=NO
- You can COLD start individually.

<i>Table 28. Summary of resources with a suffix, a dummy load module, or a COLD option</i>				
DFHSIT keyword	Default 1	Suffix 2	Dummy 3	COLD start 4
BMS	FULL	-	-	COLD
CLT	-	xx	-	-
DBP	-	xx	-	-
DCT	YES	xx	-	COLD
DIP	NO	-	program	-
DL1 or DLI	NO	-	-	COLD
FCT	YES	xx	-	-

DFHSIT keyword	Default 1	Suffix 2	Dummy 3	COLD start 4
ICP	-	-	-	COLD
JCT	YES	xx	program	-
MCT	NO	xx	5	-
PLTPI	NO	xx	-	-
PLTSD	NO	xx	-	-
SRT	YES	xx	-	-
TCT	YES	xx	table	-
TS	-	-	-	COLD
TST	NO	xx	-	-
XLT	NO	xx	-	-

Notes for Table 28 on page 307:

1 The **Default** column indicates the default value for the keyword in the DFHSIT macro.

If you code YES in the SIT for those keywords with the suffix option, an unsuffixed version of the table or program is loaded. For example, DCT=YES results in a table called DFHDCT being loaded.

You can also select an unsuffixed module or table at CICS startup by specifying **keyword=**, or **keyword=YES**. For example, if you code:

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

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

The result of specifying **keyword=**, as a system initialization parameter through PARM, SYSIPT, or CONSOLE is not necessarily the same as in the DFHSIT macro. For example, TST=, (or omitted altogether) when coding the DFHSIT macro is taken to mean TST=NO, but TST=, through any of the other three methods is the same as TST=YES.

2 The **Suffix** column indicates whether you can code a suffix. (xx indicates that a suffix can be coded.)

The DBP keyword is mandatory; you must code either a suffix or YES for DBP. For more information about the DBP system initialization parameter, see [“The system initialization parameter descriptions”](#) on page 316.

A suffix can be any 1 or 2 characters, but you must not use DY. You cannot use NO as a suffix.

If you code a suffix, a table or program with that suffix appended to the standard name is loaded. For example, DBP=2\$ causes DFHDBP2\$ dynamic backout program to be included in your CICS region.

When the suffix option is specified with other values, the two values must be enclosed within parentheses: for example, DCT=(xx,COLD).

3 The **Dummy** column indicates whether a dummy version of the program or table is loaded if you code NO. In some cases, coding NO for the operand associated with the **table** results in a dummy **program**. For more information about the effect of this option, see [“Selecting versions of CICS programs and tables”](#) on page 309.

4 The **COLD start** column indicates whether the resource can be forced to start COLD. (COLD indicates that the resource can be cold started individually).

If COLD is coded, it can be overridden only by coding START=(...,ALL) as a system initialization parameter. For more information about this option, see [“SRT={YES|NO|xx}”](#) on page 355 .

For more information about CICS table and program selection, see [“Selecting versions of CICS programs and tables”](#) on page 309.

5 If you code MCT=NO, the CICS monitoring domain builds dynamically a default monitoring control table. This ensures that default monitoring control table entries are always available for use when monitoring is on and a monitoring class is active.

Selecting versions of CICS programs and tables

A CICS program is usually made up from a group of related CICS functional modules, one example of which is the terminal control program. For most CICS programs you can only have one version, which is supplied with CICS. However, for some CICS programs you can create more than one version; for example, with different service levels. To select a particular version of a program, you can include the load library containing that version in the CICS startup JCL. For the following programs, however, you can select from different versions, by specifying the version you require at system initialization:

1. The dynamic backout program (DBP).
2. The basic mapping support (BMS) program suite.

There are two ways of selecting the versions you need:

1. Using a suffix. Use this method for DBP.
2. Explicitly selecting the level of function needed. Use this method for BMS.

You can also specify that a program is **not** needed (see [“Excluding unwanted programs”](#) on page 309 for details).

You can use these methods **only** for the programs referred to in this section and in [“Excluding unwanted programs”](#) on page 309, by coding system initialization parameters.

Using a suffix to select the dynamic backout program

Suffixes are used to distinguish different versions of the dynamic backout program (DBP), a CICS management program. Two versions of DBP are supplied with CICS, both in pregenerated format. These two suffixed versions are:

Suffix

Suffix	Description
1\$	CICS local DL/I is not supported.
2\$	CICS local DL/I is supported.

Using an explicit level of function to select programs

You use an explicit level of function to select the BMS suite of programs. When you specify your BMS requirement on the BMS system initialization parameter, you can select one of three versions. The BMS level of function is selected by the parameter options MINIMUM, STANDARD, or FULL, from which the system initialization program loads the set of programs you require.

Excluding unwanted programs

The three ways of excluding programs that are not required are by specifying one of programname=NO, tablename=NO, or function=NO.

Specifying programname=NO

If you code programname=NO in your system initialization table (for example, DIP=NO), or as a SIT override parameter, you exclude the named management program at CICS system initialization.

The programs that you can exclude by coding programname=NO are:

- Batch data interchange program (DIP)
- Terminal control program (TCP).

Note: In the case of DIP, you get a dummy version of the management program, which is supplied on the distribution tape with a suffix of **DY**.

Specifying tablename=NO for the program's control table

Not all of the CICS programs have a programname parameter in the SIT. The alternative method of excluding them is to code NO against the table name for these programs. This has the same effect as coding NO against a program name parameter, and the associated CICS program is excluded at system initialization, either by loading a dummy program, or by some other technique.

The tables that can be used in this way, and their associated management programs, are shown in [Table 29 on page 310](#).

<i>Table 29. SIT control tables with a NO option</i>	
Control tables	Associated management modules
Journal control table (JCT)	Journal control program (JCP)
System recovery table (SRT)	System recovery program (SRP)

A dummy version of the management program (suffixed **DY**) is supplied for journal control. The dummy program is DFHJCPDY. because you don't need DFHJCP if you specify no journaling.)

The dummy TCT, DFHTCTDY:

- There is a special case where you can also specify tablename=NO, but this does not load a dummy terminal control program. You specify TCT=NO when you are using resource definition online, and all of your terminal resource definitions are in the CSD.
- When you specify TCT=NO, CICS loads a dummy TCT named DFHTCTDY. If you specify TCT=NO, a generated table of this name must be available in a sublibrary of the LIBDEF PHASE, SEARCH chain for the CICS job when you start CICS. A pregenerated dummy table, and its source, are provided in the z/VSE sublibrary, PRD1.BASE.
- The dummy TCT provides **only** the CICS and VTAM control blocks that you need if you are using VTAM terminals and using the CSD for storing terminal definitions. You define your VTAM terminals using the RDO transaction, CEDA, a user application program using EXEC CICS CREATE TERMINAL commands, or the DEFINE command of the DFHCSDUP utility program.

Specifying function=NO

If you code function=NO as a system initialization parameter (for example, XRF=NO), you exclude the management program associated with the named function at CICS system initialization.

You can exclude CICS DL/I support, intersystem communication (ISC), the 3270 print-request facility, the system spooling interface, or the extended recovery facility (XRF), in this way.

Classes of start and restart

The type of initialization that CICS performs is not only determined by the START parameter. The CICS local and global catalogs also play a major role in the initialization process, together with any system

initialization parameters that you provide, either in the SIT or at run time by one of the three methods described in this section.

The global catalog

CICS uses the global catalog to save all resource definitions that are installed at CICS shutdown. These are:

- Programs
- Transactions and transaction profiles
- Transaction classes
- Terminals, including any which are autoinstalled
- Typeterms
- Connections and sessions
- BMS maps sets and partition sets
- Files.

The resource definitions that CICS saves at its shutdown may have been installed during a cold start (from a list of groups specified by a group list system initialization parameter), or during CICS execution (by CEDA INSTALL or EXEC CICS CREATE commands).

If you run CICS with START=AUTO, and a warm or emergency restart results, CICS restores all the installed resource definitions as they were at normal CICS shutdown, or at the time of system failure. The general rule is that you cannot alter installed resource definitions during a restart except by coding START=COLD.

The CICS domains also use the global catalog to save their domain status between runs. In some cases this information can be overridden during a restart by supplying system initialization parameters. For example, CICS monitoring uses the cataloged status at a restart, but modified by any system initialization parameters you provide. In other cases the domain information saved in the catalog is always used in a restart.

For example, CICS statistics interval time is always restored from the catalog in a warm or emergency restart, because the statistics domain does not have this as a system initialization parameter. To change this you must use CEMT or EXEC CICS commands after control is given to CICS. Alternatively, you can enforce system defaults by performing a cold start.

Note: If you need to reinitialize the CICS global catalog for any reason, you must also reinitialize the local catalog.

The local catalog

The CICS domains use the local catalog to save some of their information between CICS runs. If you delete and redefine the local catalog, you must:

- Initialize the CICS local catalog with an initial set of domain records.
- Use the CICS-supplied utility program, DFHSMUTL, to re-add records to enable the CICS self-tuning mechanism for storage manager domain subpools. For details of how to do this, see the [CICS Operations and Utilities Guide](#).
- Delete and reinitialize the CICS global catalog.

For more information about initializing the local catalog, see the [CICS System Definition Guide](#).

Some of the information that is saved in the local catalog can be overridden at CICS system initialization by system initialization parameters, such as CICS transaction dump data set status.

Note: If you need to reinitialize the local catalog for any reason, you must also reinitialize the global catalog.

The START system initialization parameter

You can influence the type of startup that CICS performs, by specifying the START system initialization parameter, as follows:

START=AUTO

If you code AUTO as the START operand, CICS determines, by inspecting the control record in the global catalog, which of the following three types of start to perform.

1. WARM

If the control record in the global catalog indicates that the previous run of CICS terminated normally with a successful warm keypoint, CICS performs a warm restart. The local catalog must also contain the information saved by the CICS domains during the previous execution for the warm restart to be successful. A warm start restores CICS to the state it was in at the previous shutdown.

If you are using disk journaling, the status of the disk journals is also saved in the global catalog. This information is used by CICS at startup to determine which journal data set is to be opened. You can use the JSTATUS=RESET startup parameter to cause the status in the global catalog to be ignored. During CICS startup, the status of all journal data sets is set to “ready for use”. For more information about using JSTATUS=RESET, see the [CICS System Definition Guide](#) and the description of the JSTATUS parameter “JSTATUS=RESET ” on page 338 .

You can modify a warm restart by coding the NEWSIT system initialization parameter. This has the effect of enforcing the system initialization parameters coded in the SIT, overriding any cataloged status from the previous CICS shutdown.

The exceptions to this are the system initialization parameters FCT, the CSDxxxxx group (for example CSDACC), and GRPLIST, which are always ignored in a warm restart, even if you specify NEWSIT=YES. Specifying NEWSIT=YES causes, in effect, a partial cold start.

2. COLD

If there is no control record in the CICS global catalog, CICS assumes that the catalog has been newly initialized, and forces a cold start. If you have recreated the global catalog for some reason, you must also reinitialize the local catalog.

3. EMERGENCY

If the control record in the global catalog indicates that the previous run of CICS terminated in an immediate or uncontrolled shutdown, CICS performs an emergency restart.

The emergency restart procedure uses the system log at the time of the failure to return recoverable resources to their committed states.

START=AUTO should be the normal mode of operation, with the choice of start being made by CICS automatically.

START=COLD

If you code COLD as the START operand, CICS initializes using the resource definitions specified by the system initialization parameters, ignoring any previously installed resource definitions saved in a warm keypoint in the global catalog. This includes all the groups of resources specified by the GRPLIST= system initialization parameter, and those resources specified in CICS control tables.

Be aware, however, that a start initiated by START=COLD is not entirely without reference to the previous run of a CICS system using the same global catalog.

For example, CICS checks the ‘global catalog’ for any dataset name block entries for VSAM data sets for which backout failures have occurred previously.

You can perform a fully cold start of CICS, without reference to any previous execution, only by reinitializing both CICS catalogs. Generally, do this only when you are starting your CICS system for the first time.

There may be times when it is necessary to restart CICS with START=COLD, irrespective of the type of system termination that has been recorded in the global catalog.

START=STANDBY

The STANDBY option is for use only with XRF=YES. START=STANDBY initializes an alternate CICS region. If you code START=STANDBY with XRF=NO, initialization fails with message DFHXA6530, and CICS terminates abnormally with a dump.

CICS initializes as an alternate CICS region by beginning to perform an emergency restart, which is then suspended until it needs to perform a takeover. During the period when initialization is suspended, the alternate CICS region is in standby mode and monitors the active CICS region. When it takes over, the alternate CICS region completes the emergency restart and becomes the active CICS region.

If you have specified a COLD start for other CICS resources, for example, DCT=(xx,COLD), they are cold started when the alternate CICS region (with START=STANDBY specified) takes over. This may cause CICS to lose data on an XRF takeover; for example, coding ICP=COLD results in outstanding STARTs being lost. You are recommended to code START=(STANDBY,ALL) to ensure a full emergency restart during takeover, unless you wish to specifically cold start individual resources.

For information about operating a CICS region with XRF, see the [CICS Operations and Utilities Guide](#).

START=LOGTERM

The LOGTERM operand on the START parameter is a special restart option. It is for use only when the system log is defined on **disk** data sets, to cater for an abnormal termination of a CICS system that does not close the system log. The LOGTERM option causes a CICS restart to put only an end of file label on the log, and then terminate **before** doing any backout processing. You can use START=LOGTERM, for example, if you need to close the system log to perform offline file recovery, particularly in those cases when you know (or suspect) that emergency restart won't work.

LOGTERM is available only as a system initialization parameter at run time, and cannot be coded in the system initialization table. It is also intended for use only when you are running CICS with XRF=NO.

START= parameter	State of the CICS catalogs	Result at restart
COLD	Local and global catalogs are both newly initialized.	CICS performs a fully cold start. All domains are initialized using system default values, modified by any system initialization parameters. All resources are installed as specified by system initialization parameters.
COLD	The global catalog contains a successful warm keypoint from previous run, and the local catalog contains information saved by the CICS domains.	CICS performs a cold restart, installing the resource definitions specified by system initialization parameters. The domains initialize according to system initialization parameters, or using system default values where there are no parameters (for example the statistics domain). CICS also checks the global catalog for any data set name block entries for VSAM data sets for which backout failures were recorded in the system log.
AUTO	Local and global are both newly initialized.	CICS enforces a fully cold start. All domains are initialized with system default values, modified by any system initialization parameters. All resources are installed as specified by system initialization parameters.

START= parameter	State of the CICS catalogs	Result at restart
AUTO	The global catalog contains a successful warm keypoint from the previous run. The local catalog is newly initialized.	CICS begins to perform a warm restart, but fails during the initialization process because of absence of expected records in the local catalog. (Do not reinitialize only one of the catalogs.)
AUTO	The global catalog contains a successful warm keypoint from the previous run, and the local catalog contains information saved by the domains.	CICS performs a warm restart, restoring all of your CICS system except the trace domain to the same status it was in at CICS shutdown. (CICS trace domain does not save the status of the various trace options at CICS shutdown, and always uses the system initialization parameters.) Only those resources that have a COLD option on their system initialization parameter can be cold started (for example, the destination control table or auxiliary temporary storage).

Table 31 on page 314 shows the effect of the various START options, combined with system initialization parameters where applicable, on the CICS trace, monitoring, statistics, and dump domains:

Domain	State of the CICS catalogs	Result at startup if START=AUTO	Result at startup if START=COLD
Trace	Not relevant	Domain initializes according to the system initialization parameters.	Domain initializes according to the system initialization parameters.
Monitoring	The global catalog is newly initialized.	Domain initializes according to the system initialization parameters.	Domain initializes according to the system initialization parameters.
Monitoring	The global catalog contains status of monitoring at the previous CICS shutdown.	Domain uses monitoring status from the catalog, but modified by any system initialization override parameters.	Domain initializes according to the system initialization parameters.
Statistics	The global catalog is newly initialized.	Domain initializes according to CICS-defined system default values.	Domain initializes according to CICS-defined system default values.
Statistics	The global catalog contains status of statistics at CICS shutdown.	Domain uses statistics status from the catalog.	Domain initializes according to CICS-defined system default values.
Dump	The global catalog is newly initialized.	Domain initializes the dump table according to CICS-defined system default values. Other dump attributes are set by system initialization parameters.	Domain initializes an empty dump table, and takes CICS-defined default action for all dump requests. Other dump attributes are set by system initialization parameters.

Table 31. Effect of the START= parameter on the CICS domains at initialization (continued)

Domain	State of the CICS catalogs	Result at startup if START=AUTO	Result at startup if START=COLD
Dump	The global catalog contains dump status at CICS shutdown.	Domain reads the dump table and dump status from the catalog, but the dump status is modified by any system initialization parameters.	Domain initializes an empty dump table, and takes CICS-defined default action for all dump requests. Other dump attributes are set by system initialization parameters.

CICS startup and the VTAM session

In a VTAM network, the session between CICS and VTAM is started automatically if VTAM is started before CICS. EXEC=IESWAITT is a procedure that waits until VTAM is open. If VTAM is not active when you start CICS, you receive the following messages:

```
+DFHSI1589D 'applid' VTAM is not currently active.
+DFHSI1572 'applid' Unable to OPEN VTAM ACB - RC=xxxxxxxx, ACB CODE=yy.
```

If you receive messages DFHSI1589D and DFHSI1572, and if the CICS region is not initializing as an alternate CICS region, you can start the CICS-VTAM session manually when VTAM is eventually started, by means of the CEMT SET VTAM OPEN command from any VSE console defined to CICS.

If VTAM is active, but CICS still cannot open the VTAM ACB because VTAM does not recognize the CICS APPLID, you receive the following messages:

```
+DFHSI1592I 'applid' CICS applid not (yet) active to VTAM.
+DFHSI1572 'applid' Unable to OPEN VTAM ACB - RC=00000008, ACB CODE=5A.
```

This may be caused by an error in the value of APPLID operand, in which case you must correct the error and restart CICS. For information about other causes and actions, see the [CICS Messages and Codes](#) publication.

Concurrent initialization of VTAM and XRF alternate CICS regions

An XRF alternate CICS region cannot initialize properly until it has successfully opened the VTAM ACB.

Because VTAM and the alternate CICS region may be initialized concurrently, it is possible that several tries may have to be made to open the VTAM ACB. If VTAM is not active, the following message is written to the system console every 15 seconds:

```
DFHSI1589D 'applid' VTAM is not currently active.
```

If VTAM is active, but CICS cannot open the VTAM ACB, the following messages are written to the system console:

```
+DFHSI1572 'applid' Unable to OPEN VTAM ACB - RC=xxxxxxxx, ACB CODE=yy.
DFHSI1590 'applid' XRF alternate cannot proceed without VTAM.
```

CICS abends with a dump (abend code 1590).

End of CICS startup

Whichever type of startup is performed, when the message:

```
DFHSI1517 'applid' Control is being given to CICS.
```

is displayed on the operating system console, CICS is ready to process terminal requests (*applid* is the value of the specific APPLID system initialization parameter).

When the startup process is completed, users are able to enter transactions from any terminals that are connected to CICS. For information about the CICS-supplied transactions, see the [CICS Supplied Transactions](#) publication.

The system initialization parameter descriptions

Unless otherwise stated, all of the system initialization parameters described here can be defined to CICS by any of these four ways:

1. In a DFHSIT macro
2. In a PARM parameter on the DFHSIP statement
3. In the SYSIPT data set of the CICS startup job stream
4. Through the system console.

Default notation: Default values are underscored; for example, TYPE=CSECT. This notation applies to the SIT macro parameters only.

TYPE={CSECT|DSECT}

Indicates the type of SIT to be generated.

CSECT

A regular control section that is normally used.

DSECT

A dummy control section.

ADI={30|number}

Specifies, when you are running CICS with XRF, the alternate delay interval in seconds. The minimum delay that you can specify is 5 seconds. This is the time that must elapse between the (apparent) loss of the surveillance signal in the active CICS system, and any reaction by the alternate CICS system. The corresponding parameter for the active is PDI. ADI and PDI need not have the same value.

AIEXIT={DFHZATDX|DFHZATDY|name}

Specifies the name of the autoinstall user-replaceable program that you want CICS to use when autoinstalling local VTAM terminals, APPC connections, and remote terminals. Autoinstall is the process of installing resource definitions automatically, using VTAM logon or BIND data, model definitions, and an autoinstall program.

Important:

- You can specify only one user-replaceable program on the AIEXIT parameter. Which of the CICS-supplied programs (or customized versions thereof) that you choose depends on what combination of resources you need to autoinstall.
- For background information about autoinstall, see the [CICS Resource Definition Guide](#).

DFHZATDX

A CICS-supplied autoinstall user program. This is the default. It installs definitions for locally-attached VTAM terminals, remote shipped terminals, and remote shipped connections.

DFHZATDY

A CICS-supplied autoinstall user program. It installs definitions for both locally-attached VTAM terminals, local APPC connections, remote shipped terminals, and remote shipped connections.

name

The name of your own customized autoinstall program, which may be based on one of the supplied sample programs. For programming information about writing your own autoinstall program, see the [CICS Customization](#).

AILDELAY={0|hhmmss}

Specifies the delay period that elapses after a session between CICS and an autoinstalled terminal is ended, before the terminal entry is deleted. A session is ended when a terminal logs off or when a transaction disconnects a terminal from CICS.

hhmss

Specify a 1-to 6-digit number. The default is 0, meaning the terminal entry is deleted as soon as the session is ended. If you leave out the leading zeros, they are supplied (for example, 123 becomes 000123, that is, 1 minute 23 seconds).

Note: The AIRDELAY parameter does not apply to autoinstall of APPC connections, because they are not deleted.

AIQMAX={100|number}

Specifies the maximum number of VTAM terminals and APPC connections that can be queued concurrently for autoinstall.

number

A number in the range 0 through 999. The default is 100.

A zero value disables the autoinstall function.

You should specify a number that is large enough to allow for both APPC connections and terminals.

Note: This value does not limit the total number of terminals that can be autoinstalled. If you have a large number of terminals autoinstalled, shutdown can fail due to the MXT system initialization parameter being reached or CICS becoming short on storage. For information about preventing this possible cause of shutdown failure, see the [CICS Performance Guide](#).

AIRDELAY={700|hhmss}

Specifies the delay period that elapses after an emergency restart before autoinstalled terminal entries that are not in session are deleted.

hhmss

Specify a 1-to 6-digit number. If you leave out the leading zeros, they are supplied. The default is 700, meaning a delay of 7 minutes. A value of 0 means that autoinstalled terminal definitions are not written to the global catalog and therefore are not restored at an emergency restart. For guidance about the performance implications of setting different AIRDELAY values, see the [CICS Performance Guide](#).

Note: The AIRDELAY parameter does not apply to autoinstall of APPC connections, because they are not cataloged.

XRF restriction:

- If you are running CICS with XRF, set the same value on the AIRDELAY parameter for both the active and the alternate CICS systems. It is particularly important, if you want autoinstall sessions to be reestablished after a takeover, that you avoid coding a zero on this parameter for either the active or the alternate CICS systems.
- For background information, see the [CICS XRF Guide](#).

AKPFREQ={200|number}

If AKPFREQ is a number other than zero, it specifies the number of consecutive blocks, written by DFHJCP to the system log data set, that triggers the activity keypoint function. The minimum number that should be coded is 200 (the default) and the maximum number is 65535. (The CICS region must support activity keypointing; that is, the CSKP transaction and DFHAKP program must be defined. For information about supporting activity keypointing, see the [CICS Recovery and Restart Guide](#) and the [CICS Performance Guide](#).)

If AKPFREQ=0 is coded, no activity keypoints are taken and a subsequent emergency restart is not possible.

APPLID={DBDCCICS|applid}

The VTAM application identifier for this CICS system.

applid

This name, 1 through 8 characters, identifies the CICS system in the VTAM network. It must match the name field specified in the APPL statement of the VTAM VBUILD TYPE=APPL definition.

When you define this CICS system to another CICS system, in a CONNECTION definition, you specify the applid as the NETNAME.

If the CICS system uses XRF, the form of the APPLID parameter is:

APPLID=(generic_applid,specific_applid)

Specifies the generic and specific XRF applids for the CICS system. Both applids must be 1 through 8 characters.

generic_applid

This is the generic applid for both the active and the alternate CICS systems. Therefore, you must specify the same name for *generic_applid* on the APPLID system initialization parameter for both CICS systems. Because IRC uses *generic_applid* to identify the CICS systems, there can be no IRC connection for an alternate CICS system until takeover has occurred and the alternate CICS system becomes the active CICS system.

When you define this XRF pair to another CICS system, in a CONNECTION definition, you specify the generic applid as the NETNAME.

specific_applid

This identifies the CICS system in the VTAM network. It must match the label specified in the VTAM VBUILD TYPE=APPL definition. You must specify a different *specific_applid* on the APPLID system initialization parameter for the active and for the alternate CICS system. Also, *generic_applid* and *specific_applid* must be different.

The active and alternate CICS systems use the VTAM MODIFY USERVAR command to set a user application name variable, so end users do not need to know which CICS system is active at any instant. For background information about using this command, see the [CICS XRF Guide](#).

AUTCONN={@|hhmmss}

Specify this to delay the reconnection of terminals after an XRF takeover, to allow time for manual switching. The delay is hh hours, mm minutes, ss seconds. The default value of zero means that there is no delay in the attempted reconnection.

The interval specified is the delay before the CXRE transaction runs. CXRE tries to reacquire terminals that were in session at the time of the takeover.

Note that the same delay interval applies to the connection of terminals with AUTOCONNECT(YES) specified in the RDO TYPETERM definition, at a warm or emergency restart, whether or not you have coded XRF=YES.

AUXTR={OFF|ON}

Indicates if the auxiliary trace destination is to be activated at system initialization. This parameter controls whether any of the three types of CICS trace entry are written to the auxiliary trace data set. The three types are: CICS system trace (see the SYSTR parameter), user trace (see the USERTR parameter), and exception trace entries (that are always made and are not controlled by a system initialization parameter).

OFF

Do not activate auxiliary trace.

ON

Activate auxiliary trace.

For details of internal tracing in main storage, see [“INTTR={ON|OFF}”](#) on page 338 .

AUXTRSW={NO|ALL|NEXT}

Specifies whether you want the auxiliary trace autoswitch facility.

NO

Disables the autoswitch facility.

NEXT

Enables the autoswitch facility to switch to the next data set at end of file of the first data set used for auxiliary trace. Coding NEXT permits one switch only, and when the second data set is full, auxiliary trace is switched off.

ALL

Enable the autoswitch facility to switch to the inactive data set at every end of file. Coding ALL permits continuous switching between the two auxiliary trace data sets, DFHAUXT and DFHBUXT, and whenever a data set is full, it is closed and the other data set is opened.

BMS=({MINIMUM|STANDARD|FULL}, COLD)[,{UNALIGN|ALIGN}], { DDS|NODDS}}

Specifies which version of basic mapping support you want to be included. The function included in each version of BMS is shown in [Table 32 on page 319](#). The parameter BMS can be overridden during CICS initialization.

You need full or standard function BMS, if you are using XRF and have specified MESSAGE for RECOVNOTIFY on any of your RDO TYPETERM resource definitions.

MINIMUM

The minimum version of BMS is included.

STANDARD

The standard version of BMS is included.

FULL

The full version of BMS is included. This is the default in the SIT.

COLD

CICS deletes delayed messages from temporary storage, and destroys their interval control elements (ICEs).

UNALIGN

Specifies that all BMS maps assembled before CICS/DOS/VS Version 1 Release 6 are unaligned. Results are unpredictable if the stated alignment does not match the actual alignment.

ALIGN

Code this to indicate that all BMS maps assembled before CICS/DOS/VS Version 1 Release 6 are aligned.

DDS

BMS is to load suffixed versions of map sets and partition sets. BMS first tries to load a version that has the alternate suffix (if the transaction uses the alternate screen size). If the load fails, BMS tries to load a version that has the default map suffix. If this fails too, BMS tries to load the unsuffixed version. DDS, which stands for “device dependent suffixing”, is the default.

You need to use map suffixes only if the same transaction is to be run on terminals with different characteristics (in particular, different screen sizes). If you do not use suffixed versions of map sets and partition sets, CICS need not test for them.

NODDS

BMS is not to load suffixed versions of map sets and partition sets. Specifying NODDS avoids the search for suffixed versions, saving processor time.

BMS version	Devices supported	Function provided
MINIMUM	All 3270 system display units and printers except SNA character string printers, which are defined as DEVICE(SCSPRINT) on the RDO TYPETERM definition or as TRMTYPE=SCSPRT in DFHTCT	SEND MAP command, RECEIVE MAP command, SEND CONTROL command. Default and alternate screens; extended attributes; map set suffixes; screen coordination with null maps; and block data

<i>Table 32. Versions of BMS (continued)</i>		
BMS version	Devices supported	Function provided
STANDARD	All devices are supported by BMS. These are listed in the CICS Application Programming Guide	All function of MINIMUM, plus outboard formats, partitions, controlling a magnetic slot reader, NLEOM mode for 3270 system printers, SEND TEXT command, and Subsystem LDC controls
FULL	All devices supported by BMS. These are listed in the CICS Application Programming Guide	Same as STANDARD, plus terminal operator paging, cumulative mapping, page overflow, cumulative text processing, routing, message switching returning BMS-generated data stream to program before output.

CDSASZE={0K|number}

Specifies the size of the CDSA. The default size is 0 indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

number

Specify number as an amount of storage in the range 0 through 16777215 bytes in multiples of 262144 bytes (256KB). If the size specified is not a multiple of 256KB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4194304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

Restrictions: You can code the CDSASZE parameter in PARM, SYSIPT, or CONSOLE only.

CHKSTRM={CURRENT|NONE}

Activate, or deactivate, terminal storage-violation checking. The operands have the following meanings:

CURRENT

Code CURRENT to check for TIOA storage violations.

NONE

Code NONE to deactivate TIOA storage-violation checking.

You can also use the CICS-supplied transaction, CSFE, to switch terminal storage-violation checking on and off.

For information about checking for storage violations, see the [CICS Problem Determination Guide](#).

Restrictions: You can code the CHKSTRM parameter in PARM, SYSIPT, or CONSOLE only.

CHKSTSK={ALL|CURRENT|NONE}

Activate, or deactivate, task storage-violation checking at startup. The operands have the following meanings:

ALL

Code ALL to check all storage areas on the transaction storage chains for all tasks.

CURRENT

Code CURRENT to check all storage areas on the transaction storage chain for the current task only.

NONE

Code NONE to deactivate task storage-violation checking.

You can also use the CICS-supplied transaction, CSFE, to switch task storage-violation checking on and off.

For information about checking for storage violations, see the [CICS Problem Determination Guide](#).

Restrictions: You can code the CHKSTSK parameter in PARM, SYSIPT, or CONSOLE only.

CLINTCP={437|codepage}

Specifies the default client code page to be used by the DFHCNV data conversion table but only if the CLINTCP parameter in the DFHCNV macro is set to SYSDEF.

CLSDST={NOTIFY|NONOTIFY}

Specifies the notification required for an EXEC CICS ISSUE PASS command. This parameter is applicable to both autoinstalled and non-autoinstalled terminals. You can use the notification in a user-written node error program to reestablish the CICS session when a VTAM CLSDST PASS request resulting from an EXEC CICS ISSUE PASS command fails. For more information about the EXEC CICS ISSUE PASS command, see the [CICS Application Programming Reference](#).

NOTIFY

CICS requests notification from VTAM when the EXEC CICS ISSUE PASS command is executed.

NONOTIFY

CICS does not request notification from VTAM.

CLT=xx

Specify the suffix for the command list table (CLT), if this SIT is used by an alternate XRF system. The name of the table is DFHCLTxx.

For information about coding the macros for this table, see the [CICS Resource Definition Guide](#).

CMDPROT={YES|NO}

Specifies whether to allow, or inhibit, CICS validation of start addresses of storage referenced as output parameters on EXEC CICS commands.

YES

If you specify YES, CICS validates the initial byte at the start of any storage that is referenced as an output parameter on EXEC CICS commands to ensure that the application program has write access to the storage. This ensures that CICS does not overwrite storage on behalf of the application program when the program itself cannot do so. If CICS detects that an application program has asked CICS to write into an area to which the application does not have addressability, CICS abends the task with an AEYD abend.

The level of protection against bad addresses depends on the level of storage protection in the CICS environment. The various levels of protection provided when you specify CMDPROT=YES are shown in [Table 33 on page 321](#).

NO

If you specify NO, CICS does not perform any validation of addresses of the storage referenced by EXEC CICS commands. This means an application program could cause CICS to overwrite storage to which the application program itself does not have write access.

Environment	Execution key of affected programs	Types of storage referenced by applications that cause AEYD abends
Read-only storage (RENTPGM=PROTECT)	CICS-key and user-key	CICS key 0 read-only storage (RDSA and ERDSA).
Subsystem storage protection (STGPROT=YES)	User-key	All CICS-key storage (CDSA and ECDSA)
Base CICS (all storage is CICS key storage) (RENTPGM=NOPROTECT; and STGPROT=NO)	CICS-key and user-key	VSE storage only

CMDSEC={ASIS|ALWAYS}

Specifies whether or not you want CICS to honor the CMDSEC option specified on a transaction's resource definition.

ASIS

means that CICS honors the CMDSEC option defined in a transaction's resource definition. CICS calls its command security checking routine only when CMDSEC(YES) is specified in a RDO TRANSACTION resource definition.

ALWAYS

means that CICS overrides the CMDSEC option, and always calls its command security checking routine to issue the appropriate call to the System Authorization Facility (SAF) interface.

Note:

1. Specify ALWAYS when you want to control the use of the SPI in all your transactions. Be aware that this might incur additional overhead. The additional overhead is caused by CICS issuing the command security calls on every eligible EXEC CICS command, which are *all* the system programming interface (SPI) commands.
2. If you specify ALWAYS, command checking applies to CICS-supplied transactions such as CESN and CESF. You must authorize all users of CICS-supplied transactions to use the internal CICS resources for the transactions, otherwise you will get unexpected results in CICS-supplied transactions.

Restrictions: You can code the CMDSEC parameter in the SIT, PARM, or SYSIPT only.

CONFDATA={SHOW|HIDETC}

Specifies whether CICS is to suppress (hide) user data that might otherwise appear in CICS trace entries or in dumps that contain the VTAM receive any input area (RAIA.). This option applies to initial input data received on a VTAM RECEIVE ANY operation, the initial input data received on an MRO link, and FEPI screens and RPLAREAs.

SHOW

Data suppression is not in effect. User data is traced regardless of the CONFDATA option specified in transaction resource definitions. This option overrides the CONFDATA option in RDO TRANSACTION resource definitions.

HIDETC

Specifies that you want CICS to 'hide' user data from CICS trace entries. It also indicates that VTAM RAIAs are to be suppressed from CICS dumps. The action actually taken by CICS is subject to the individual CONFDATA attribute on the RDO TRANSACTION resource definition (see [Table 34 on page 323](#)).

If you specify CONFDATA=HIDETC, CICS processes VTAM, MRO, and FEPI user data as follows:

• VTAM

CICS clears the VTAM RAIA containing initial input as soon as it has been processed, and before the target transaction has been identified.

The normal trace entries (FC90 and FC91) are created on completion of the RECEIVE ANY operation with the text "SUPPRESSED DUE TO CONFDATA=HIDETC IN SIT" replacing all the user data except the first 4 bytes of normal data, or the first 8 bytes of function management headers (FMHs).

CICS then identifies the target transaction for the data. If the RDO TRANSACTION resource definition specifies CONFDATA(NO), CICS traces the user data that it suppressed from the FC90 trace in the trace entry AP FC92. This trace entry is not created if the transaction is defined with CONFDATA(YES) on the RDO TRANSACTION resource definition.

• MRO

CICS does not trace the initial input received on an MRO link.

The normal trace entries (DD16, DD23, and DD25) are created with the text "SUPPRESSED DUE TO CONFDATA=HIDETC IN SIT" replacing all the user data.

CICS then identifies the target transaction for the data. If the transaction definition specifies CONFDATA(NO), CICS traces the user data that it suppressed from DD16 in the trace entry AP FC92. This special trace entry is not created if the transaction is defined with CONFDATA(YES).

- **FEPI**

FEPI screens and RPL data areas (RPLAREAs) areas are suppressed from all FEPI trace points if CONFDATA(YES) is specified in the transaction resource definition. The user data in the FEPI trace points AP 1243, AP 1244, AP 145E, AP 145F, AP 1460, AP 1461, AP 1595, AP 1596, AP 1597, AP 1598, and AP 1599 is replaced with the message "SUPPRESSED DUE TO CONFDATA=HIDETC IN SIT". If the RDO TRANSACTION resource definition specifies CONFDATA(NO), the FEPI trace entries are created with the user data as normal.

Mirror transactions

The CICS-supplied mirror RDO TRANSACTION resource definitions are specified with CONFDATA(YES). This ensures that, when you specify CONFDATA=HIDETC as a system initialization parameter, CICS regions running mirror transactions suppress user data as described for VTAM and MRO data.

Modified data

By waiting until the transaction has been identified to determine the CONFDATA option, VTAM or MRO data may have been modified (for example, it may have been translated to upper case).

The interaction between the CONFDATA system initialization parameter and the CONFDATA attribute on the RDO TRANSACTION resource definition is shown in [Table 34 on page 323](#).

CONFDATA on RDO TRANSACTION resource definition	CONFDATA system initialization parameter	
	SHOW	HIDETC
NO	Data not suppressed	VTAM RAIAs are cleared. Initial input of VTAM and MRO data is suppressed from the normal FC90, FC91, DD16, DD23, and DD25 trace entries. For FC90 and DD16 traces only, suppressed user data is traced separately in an FC92 trace entry. FEPI screens and RPLAREAs are traced as normal.
YES	Data not suppressed	VTAM RAIAs are cleared. All VTAM, MRO, and FEPI user data is suppressed from trace entries.

You cannot modify the CONFDATA option while CICS is running. You must restart CICS to make such a change.

Restrictions: You can code the CONFDATA parameter in the SIT, PARM, and SYSIPT only.

CONFTXT={NO|YES}

Specifies whether CICS is to prevent VTAM from tracing user data.

NO

CICS does not prevent VTAM from tracing user data.

YES

CICS prevents VTAM from tracing user data.

Restrictions: You can code the CONFTXT parameter in the SIT, PARM, and SYSIPT only.

CSDACC={READWRITE|READONLY}

Specifies the type of access to the CSD to be permitted to this CICS region. Note that this parameter is effective only when you start CICS with a START=COLD parameter. If you code START=AUTO, and CICS performs a warm or emergency restart, the file resource definitions for the CSD are recovered from the CICS global catalog. However, you can redefine the type of access permitted to the CSD dynamically with a CEMT SET FILE, or an EXEC CICS SET FILE, command.

READWRITE

Read/write access is allowed, permitting the full range of CEDA, CEDB, and CEDC functions to be used.

READONLY

Read access only is allowed, limiting the CEDA and CEDB transactions to only those functions that do not require write access.

CSDBUFND=number

Specifies the number of buffers to be used for CSD data. The minimum you should specify is the number of strings coded on the CSDSTRNO parameter plus 1, up to a maximum of 32768. Note that this parameter is used only if you have also coded CSDLRNO=NONE; if you have coded CSDLRNO=number, CSDBUFND is ignored.

If you specify a value for CSDBUFND that is less than the required minimum (the CSDSTRNO value plus 1), VSAM automatically changes the number of buffers to the number of strings plus 1 when CICS issues the OPEN macro for the CSD.

This parameter is effective only on a CICS cold start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog.

CSDBUFNI=number

Specifies the number of buffers to be used for the CSD index. The minimum you should specify is the number of strings coded on the CSDSTRNO parameter, up to a maximum of 32768. This parameter is used only if you have also coded CSDLRNO=NONE; if you have coded CSDLRNO=number, CSDBUFNI is ignored.

If you specify a value for CSDBUFNI that is less than the required minimum (the CSDSTRNO value), VSAM automatically changes the number of buffers to the number of strings when CICS issues the OPEN macro for the CSD.

This parameter is effective only on a CICS cold start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the CICS global catalog.

CSDFRLOG=number

Specifies a journal identifier to indicate the journal that you want to use for forward recovery of the CSD. This parameter is used only if CSDRECOV=ALL is specified, otherwise it is ignored. If you omit CSDFRLOG, but specify CSDRECOV=ALL, CSDFRLOG defaults to 1, which indicates that the CICS system log is to be used for forward recovery of the CSD.

The CSDRECOV, and CSDFRLOG system initialization parameters interact according to how they are specified. For information about their effects when the SIT is assembled and during CICS override processing, see the [CICS System Definition Guide](#).

This parameter is effective only on a CICS cold start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog.

number

The journal identification of the journal that is to be used for forward recovery. The number must be in the range 1 through 99. The number 1 indicates the CICS system log, and any other number refers to a user journal.

CSDJID={NO|number}

Specifies the journal identifier of the journal that you want CICS to use for automatic journaling of file requests against the CSD.

This parameter is effective only on a CICS cold start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog.

NO

Code NO if you do **not** want automatic journaling for the CSD. This is the default.

number

A number in the range 1 through 99 to identify the journal that CICS is to use for automatic journaling for the CSD. The number 1 indicates the CICS system log, and any other number refers to another CICS journal.

The automatic journaling options enforced for the CSD when you code CSDJID=number are JNLADD=BEFORE and JNLUPDATE=YES. These options are sufficient to record enough information for a user-written forward recovery utility. No other automatic journaling options are available for the CSD. For information about the options JNLADD=BEFORE and JNLUPDATE=YES, see the [CICS Resource Definition Guide](#).

CSDLSRNO={1|number|NONE|NO}

Specifies whether the CSD is to be associated with a local shared resource (LSR) pool.

This parameter is effective only on a CICS cold start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog. However, you can redefine the LSR pool attribute for the CSD dynamically with an EXEC CICS SET FILE command.

1

The default LSR pool number is 1.

number

The number of the LSR pool the CSD is to be associated with. The number of the pool must be in the range 1 through 15.

NONE|NO

Code NONE (or NO) if the CSD is not to be associated with an LSR pool.

CSDRECOV={NONE|ALL|BACKOUTONLY}

Specifies whether the CSD is a recoverable file.

The CSDRECOV, and CSDFRLOG system initialization parameters interact according to how they are specified. For information about their effects when the SIT is assembled and during CICS override processing, see the [CICS System Definition Guide](#).

This parameter is effective only on a CICS cold start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog.

NONE

The default is that the CSD is not recoverable.

ALL

Code ALL to specify that you want both forward recovery and backout for the CSD. If you code ALL, also specify CSDFRLOG with the journal identification of the journal to be used for forward recovery of the CSD. If you do not code the CSDFRLOG parameter, CICS uses the system log for forward recovery.

Note: For backout purposes, CICS always uses the system log. (See the BACKOUTONLY option.)

BACKOUTONLY

Code BACKOUTONLY to limit CSD recovery to file backout only. If you specify backout for the CSD, CICS uses the system log to record before images for backout purposes.

CSDSTRNO={2|number}

Specifies the number of concurrent requests that can be processed against the CSD. When the number of requests reaches the STRNO value, CICS automatically queues any additional requests until one of the active requests terminates.

CICS requires two strings per CSD user, and you can increase the CSDSTRNO value, in multiples of two, to allow more than one concurrent CSD user.

See section “Multiple Users of the CSD Within a CICS Region” in the [CICS System Definition Guide](#), before you code this parameter.

This parameter is effective only on a CICS cold start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog. However, you can redefine the number of strings for the CSD dynamically with an EXEC CICS SET FILE command.

2

The minimum number of concurrent requests for the CSD is 2.

number

This number must be a multiple of 2, in the range 2 through 254.

CWAKEY={USER|CICS}

Specifies the storage key for the common work area (CWA) if you are operating CICS with storage protection (STGPROT=YES). (You specify how much storage you want for the CWA on the WRKAREA parameter.) The permitted values are USER (the default), or CICS:

USER

If you specify USER, or allow this parameter to default, CICS obtains storage for the CWA in user key. This allows a user program executing in any key to modify the CWA.

CICS

If you specify CICS, CICS obtains storage for the CWA in CICS key. This means that only programs executing in CICS key can modify the CWA, and user-key programs have read-only access.

If CICS is running without storage protection, the CWAKEY parameter is ignored, and the CWA is always allocated from CICS-key storage.

DATFORM={MMDDYY|DDMMYY|YYMMDD}

Specifies the external date display standard that you want to use for CICS date displays. An appropriate indicator setting is made in the CSA. It is examined by CICS supplied system service programs that display a Gregorian date. CICS maintains the date in the form OCYYDDD in the CSA (where C=0 for years 19xx, 1 for years 20xx, and so on; YY=year of century; and DDD=day of year), and converts it to the standard you specify for display.

MMDDYY

The date is in the form of month-day-year.

DDMMYY

The date is in the form of day-month-year.

YYMMDD

The date is in the form of year-month-day.

DBP={1\$|2\$|xx|YES}

Specifies which version of the dynamic transaction backout program is to be part of the system. This DBP parameter is mandatory, and has no default. (See page [“Excluding unwanted programs”](#) on page 309 for more information about coding this parameter.)

If you are using local DL/I support, specify DBP=2\$. If you have generated a user-defined DFHDBP program for use with local DL/I support, specify either DBP=xx, where xx is the suffix of your program, or DBP=YES if you have defined an unsuffixed version of DFHDBP.

If you are not using local DL/I support, specify DBP=1\$.

For background information about dynamic transaction backout, see the [CICS Recovery and Restart Guide](#).

The dynamic transaction backout program needs a program resource definition entry in the CSD. This entry must be included in your GRPLIST list at CICS initialization. The CICS-supplied programs DFHDBP1\$ and DFHDBP2\$ have resource definition entries in the CSD group DFHBACK, which is included in the default group list DFHLIST. If you use your own dynamic backout program, you must create a RDO PROGRAM resource definition entry for it in the CSD, and include the entry in one of your GRPLIST lists at CICS initialization. You are recommended to code RESIDENT(YES) on your RDO PROGRAM definition. For information about the required program resource definition entry, see the [CICS Resource Definition Guide](#).

Restrictions: You must code a DBP parameter in the system initialization table, although you can override that DBP parameter in PARM, SYSIPT, or CONSOLE.

DBUFSZ={500|number}

Specifies the maximum size of the dynamic log buffer (needed for dynamic transaction backout) for each transaction. CICS initially allocates for each buffer half the maximum size that you specify, and subsequently uses the value specified to calculate the actual size of the dynamic buffer. The value can be in the range 6 through 32000; the default is 500.

If the resultant dynamic buffer is too small, CICS spills the dynamic log data to a further area of main storage. (CICS allocates all dynamic log storage, including spilled buffers, above the 16MB line.)

For information on the dynamic log, and how you can tune this parameter, see *Dynamic transaction backout statistics* and *Dynamic log buffer size (DBUFSZ)* in the CICS Performance Guide.

DCT={YES|xx|NO},COLD]

Specifies the destination control table suffix, for more information, refer to “Notes on CICS resource table and module keywords” on page 307. For information about coding the macros for this table, see the CICS Resource Definition Guide.

DFLTUSER={CICSUSER|userid}

Specifies the external security manager (ESM) userid with the security attributes to be used for all terminal users who have not explicitly signed on (by the CESN transaction, the EXEC CICS SIGNON command, or by the preset security options of the TERMINAL resource definition).

The specified userid must be defined to your ESM if you are using external security (that is, you have coded SEC=YES).

The specified userid is signed on during CICS initialization. If it cannot be signed on, CICS fails to initialize.

Restrictions: You can code the DFLTUSER parameter in the SIT, PARM, or SYSIPT only.

DIP={NO|YES}

Code YES to include the batch data interchange program, DFHDIP, which supports the batch controller functions of the IBM 3790 Communication System and the IBM 3770 Data Communication System. For more information, refer to “Notes on CICS resource table and module keywords” on page 307. (Support is provided for the transmit, print, message, user, and dump data sets of the 3790 system.)

DISMACP={YES|NO}

DISMACP=YES allows CICS to disable any transaction that terminates abnormally with an ASRD abend (caused by a user program invoking a CICS macro, or referencing the CSA or the TCA).

Note: DISMACP=YES has no effect if the ASRD abend is handled by an active abend exit.

{DLI|DL1}={NO|xx|YES},COLD]

Specifies whether DL/I databases are to be accessed during this run of CICS.

This parameter selects only DL/I support that runs within the CICS address space.

NO

DL/I is not to be used.

xx

DL/I is used. xx is the 1- or 2-character suffix of the DL/I application control table (ACT) specified on the DLZACT TYPE=INITIAL macro, which results in the module DLZNUCxx.

YES

DL/I is to be used. You must also specify DBP=2\$.

COLD

If you code DLI=YES, you can specify the COLD option if you want to cold start the DL/I resources. The default is the start option coded on the START parameter. If you specify COLD, DL/I databases are not backed out in the event of an emergency restart.

Note: On initialization of an XRF alternate system, the DLI COLD option is overridden so that DL/I is emergency restarted, and so DL/I databases are backed out.

DLIOER={ABEND|CONTINUE}

Code the DLIOER parameter to tell CICS what to do in the event of DL/I detecting an I/O error on a database.

ABEND

CICS abends. In an XRF configuration, the alternate system is canceled. Global user exit, XDBDERR, is not invoked.

CONTINUE

CICS continues to run. DL/I flags the database as unusable and allows the use of DL/I utilities to perform database recovery and to restart the database.

DOCCODEPAGE={037|codepage}

specifies the default host code page to be used by the document domain. The *codepage* is a field of up to 8 characters. If *codepage* value is not specified, the default *doccodepage* is set to 037.

DSALIM={5M|number}

Specifies the upper limit of the total amount of storage within which CICS can allocate the individual dynamic storage areas (DSAs) that reside below the 16MB boundary.

5M

The default DSA limit is 5MB (5 242 880).

number

Specify *number* as an amount of storage in the range 2MB to 16MB (2 097 152 bytes to 16 777 216 bytes) in multiples of 262 144 bytes (256KB). If the size specified is not a multiple of 256KB, CICS rounds the value up to the next multiple.

You can specify *number* in bytes (for example, 4 194 304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

From the storage size that you specify on the DSALIM parameter, CICS allocates the following dynamic storage areas:

The user DSA (UDSA)

The user-key storage area for all user-key task-lifetime storage below the 16MB boundary.

The read-only DSA (RDSA)

The key-0 storage area for all reentrant programs and tables below the 16MB boundary.

The shared DSA (SDSA)

The user-key storage area for any non-reentrant user-key RMODE(24) programs, and also for any storage obtained by programs issuing EXEC CICS GETMAIN commands for storage below the 16MB boundary with the SHARED option.

The CICS DSA (CDSA)

The CICS-key storage area for all non-reentrant CICS-key RMODE(24) programs, all CICS-key task-lifetime storage below the 16MB boundary, and for CICS control blocks that reside below the 16MB boundary.

Note: CICS allocates the UDSA and the other DSAs below 16MB in multiples of 256KB. The maximum you can specify depends on a number of factors, such as how you have configured your VSE storage (which governs how much private storage remains below the line) and how much private storage you must leave free to satisfy VSE GETVIS requests for storage outside the DSAs.

For information about calculating the amount of storage to specify on the DSALIM parameter, see the [CICS Performance Guide](#).

DSHIPIDL={020000|hhmmss}

Specifies the minimum time, in hours, minutes, and seconds, that an *inactive* shipped terminal definition must remain installed in this region. When the timeout delete mechanism is invoked, only those shipped definitions that have been inactive for longer than the specified time are deleted.

You can use this parameter in a transaction routing environment, on the application-owning and intermediate regions, to prevent terminal definitions having to be reshipped because they have been deleted prematurely.

The default minimum idle time is 2 hours.

hhmmss

Specify a 1 to 6 digit number in the range 0 995959. Numbers that have fewer than six digits are padded with leading zeros.

DSHIPINT={120000|0|hhmmss}

Specifies the interval between invocations of the timeout delete mechanism. The timeout delete mechanism removes any shipped terminal definitions that have not been used for longer than the time specified by the DSHIPIDL parameter.

You can use this parameter in a transaction routing environment, on the application-owning and intermediate regions, to control:

- How often the timeout delete mechanism is invoked.
- The approximate time of day at which a mass delete operation is to take place, relative to CICS startup.

Note: For more flexible control over when mass delete operations take place, you can use a CEMT SET DELETSHIPED or EXEC CICS SET DELETSHIPED command to reset the interval. (The revised interval starts *from the time the command is issued*, **not** from the time the remote delete mechanism was last invoked, nor from CICS startup.)

0

The timeout delete mechanism is not invoked. You might set this value in a terminal-owning region, or if you are not using shipped definitions.

hhmmss

Specify a 1 to 6 digit number in the range 1 995959. Numbers that have fewer than six digits are padded with leading zeros.

DTRTRAN={CRTX|name|NO}

This is the name of the transaction definition that you want CICS to use for dynamic transaction routing. This is intended primarily for use in a CICS terminal-owning region, although you can also use it in an application-owning region when you want to daisy-chain transaction routing requests. In a dynamic transaction routing environment, the transaction named on DTRTRAN must be installed in the CICS terminal-owning regions if you want to eliminate the need for resource definitions for individual transactions.

CRTX

This is the default dynamic transaction definition. It is the name of the CICS-supplied sample transaction resource definition provided in the CSD group DFHISC.

name

The name of your own dynamic transaction resource definition that you want CICS to use for dynamic transaction routing.

NO

The dynamic transaction routing program is not invoked when a transaction definition cannot be found.

For information about the CICS-supplied sample transaction resource definition, CRTX, and about defining your own dynamic transaction routing definition, see the [CICS Resource Definition Guide](#).

DTRPGM={DFHDYP|program-name}

Specifies the name of the dynamic transaction routing program you want to use for routing transactions that are defined with the DYNAMIC attribute. DFHDYP, the default, is the name of the CICS-supplied version.

DUMP={YES|NO}

Specifies whether the CICS dump domain is to take SDUMPs.

YES

SDUMPs are produced, unless suppressed by the options specified in the CICS system dump table.

NO

SDUMPs are suppressed.

Note: This does not prevent the CICS kernel from taking SDUMPs.

For more information about SDUMPs, see the [CICS System Definition Guide](#).

DUMPDS={AUTO|A|B}

Specifies the transaction dump data set that is to be opened during CICS initialization.

AUTO

For all types of start, CICS opens the transaction dump data set that was **not** in use when the previous CICS run terminated. This information is obtained from the CICS local catalog (DFHLCD).

If you specify AUTO, or let it default, code DLBL statements for both of the transaction dump data sets, DFHDMPA and DFHDMPB, in your CICS startup job stream.

A

CICS opens transaction dump data set DFHDMPA.

B

CICS opens transaction dump data set DFHDMPB.

DUMPSW={NO|NEXT}

Specifies whether you want CICS to switch automatically to the next dump data set when the first is full.

NO

Disables the CICS autoswitch facility. If the transaction dump data set opened during initialization becomes full, CICS issues a console message to notify the operator. If you want to switch to the alternate data set, you must do so manually using the CEMT or EXEC CICS SET DUMPDS SWITCH command.

NEXT

Enables the autoswitch facility to switch to the next data set at end of file of the data set opened during initialization. Coding NEXT permits one switch only. If you want to switch to the alternate data set again, you must do so manually using CEMT or EXEC CICS SET DUMPDS SWITCH command. If you specify NEXT, code DLBL statements for both of the transaction dump data sets, DFHDMPA and DFHDMPB, in your CICS startup job stream.

For more information about transaction dump data sets, see the [CICS System Definition Guide](#)

ECDSASZ={0K|number}

Specifies the size of the ECDSA. The default size is 0 indicating that the DSA size can be changed dynamically. A non-zero value indicates that the DSA size is fixed.

number

Specify number as an amount of storage in the range 0 to 1073741824 bytes in multiples of 1048576 bytes (1MB). If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4194304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

EDSALIM={20M|number}

Specifies the upper limit of the total amount of storage within which CICS can allocate the individual extended dynamic storage areas (EDSAs) that reside above the 16MB boundary.

20M

The default EDSA limit is 20MB (20 971 520 bytes).

number

Specify *number* as a value in the range 10MB to 2047MB, in multiples of 1MB. If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify *number* in bytes (for example, 33 554 432), or as a whole number of kilobytes (for example, 32 768K), or a whole number of megabytes (for example, 32M).

The maximum value allowed depends on a number of factors, such as:

- The size of the CICS partition
- How much storage you require for the CICS internal trace table.
- How much private storage you must leave free to satisfy VSE GETVIS requests for storage above the 16MB boundary outside the DSAs.

From the storage value that you specify on the EDSALIM parameter, CICS allocates the following extended dynamic storage areas:

The extended user DSA (EUDSA)

The user-key storage area for all user-key task-lifetime storage above the 16MB boundary.

The extended read-only DSA (ERDSA)

The key-0 storage area for all reentrant programs and tables above the 16MB boundary.

The extended shared DSA (ESDSA)

The user-key storage area for any non-reentrant user-key RMODE(ANY) programs, and also for any storage obtained by programs issuing CICS GETMAIN commands for storage above the 16MB boundary with the SHARED option.

The extended CICS DSA (ECDSA).

The CICS-key storage area for all non-reentrant CICS-key RMODE(ANY) programs, all CICS-key task-lifetime storage above the 16MB boundary, and CICS control blocks that reside above the 16MB boundary.

CICS allocates all the DSAs above the 16MB boundary in multiples of 1MB.

Note: For information about calculating the amount of storage to specify on the EDSALIM parameter, see the [CICS Performance Guide](#). For example, the value that you specify must allow for all temporary storage MAIN requests, for which CICS uses the ECDSA.

ENCRYPTION={STRONG|SSLV3}

Specifies the level of encryption you want to use for TCP/IP connections using the secure sockets layer. When a secure connection is established, the most secure cipher suite that is supported by both sides is used. You can specify an option only if you have the underlying encryption support in the z/VSE operating system. Possible values are:

STRONG

Allows the use of TLS version 1.0 or higher.

SSLV3

Allows the use of SSL version 3.0 or higher. SSL version 3.0 should only be used for a migration period while clients that still require this protocol are upgraded. If any of the clients that connect do not support TLS 1.0 or higher, you can select SSLV3. CICS TS will then accept less secure SSL V3 client connections.

EODI={E0|xx}

Specifies the end-of-data indicator for input from sequential devices. The characters "xx" represent two hexadecimal digits in the range 01 through FF. The default value is X'E0', which represents the standard EBCDIC backslash symbol (\).

ERDSASIZE={0K|number}

Specifies the size of the ERDSA. The default size is 0 indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

number

Specify number as an amount of storage in the range 0 to 1073741824 bytes in multiples of 1048576 bytes (1MB). If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4194304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4MB).

ESDSASZE={@K|number}

Specifies the size of the EDSA. The default size is 0 indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

number

Specify number as an amount of storage in the range 0 to 1073741824 bytes in multiples of 1048576 bytes (1MB). If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4194304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

ESMEXITS={NOINSTLN|INSTLN}

Specifies whether installation data is to be passed via the RACROUTE interface to the external security manager (ESM) for use in exits written for the ESM.

NOINSTLN

Specifies that the INSTLN parameter is not used in RACROUTE macros.

INSTLN

Specifies that CICS-related and installation-supplied data is passed to the ESM using the INSTLN parameter of the RACROUTE macro. For programming information, including the format of the data passed, see the [CICS Customization Guide](#). This data is intended for use in exits written for the ESM.

Restrictions: You can code the ESMEXITS parameter in the SIT only.

EUDSASZE={@K|number}

Specifies the size of the EDSA. The default size is 0 indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

number

Specify number as an amount of storage in the range 0 to 1073741824 bytes in multiples of 1048576 bytes (1MB). If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4194304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

FCT={YES|xx|NO}

Specifies the suffix of the file control table (FCT) to be used.

This parameter is effective only on a CICS cold start. CICS does not load an FCT on a warm or emergency restart, and all file resource definitions are recovered from the CICS global catalog.

For information about coding the macros for this table, see the [CICS Resource Definition Guide](#).

You can use a mixture of macro definitions and RDO definitions for files in your CICS region. However, your FCT should not contain definitions for files that are also defined in an RDO group specified in the group list for the CICS startup, and the converse. In a cold start, CICS first loads the file definitions from the FCT, and adds any files that are defined in RDO groups later (using the groups specified in the GRPLIST parameter). If CICS tries to install a file definition from a group list for a file already defined by an entry from the FCT, the install may fail because of conflicting file attributes. You can avoid this potential conflict by ensuring that you do not have duplicate file entries in the FCT and in your group list.

FEPI={NO|YES}

Specifies whether or not you want to use the Front End Programming Interface feature (FEPI).

NO

Means that FEPI support is not required. You should specify NO on this parameter (or allow it to default) if you do not have the feature installed, or if you do not require FEPI support.

YES

Means you require FEPI support, and causes CICS to start the CSZI transaction.

This publication does not contain any information about the installation process for the Front End Programming Interface feature. Installation information can be found in the [CICS Front End Programming Interface User's Guide](#).

FLDSEP={ ' ' | 'xxxx' }

Specifies 1 through 4 field-separator characters, each of which indicates end of field in the terminal input data. The default is four blanks.

The field separator allows you to use transaction identifications of less than four characters followed by one of the separator characters. When less than four characters are coded, the parameter is padded with blanks, so that the blank is then a field separator. None of the specified field separator characters should be part of a transaction identification; in particular, the use of alphabetic characters as field separators is not recommended.

The character specified in the FLDSEP parameter must not be the same as any character specified in the FLDSTRT parameter. This means that it is invalid to allow both parameters to take the default value.

Restrictions:

- If you specify FLDSEP in the SIT, the characters must be enclosed in single quotation marks.
- If you specify FLDSEP as a PARM, SYSIPT, or CONSOLE parameter, do **not** enclose the characters in quotation marks, and the characters you choose must not include an embedded blank, or any of these characters:

```
( ) ' = ,
```

FLDSTRT={ ' ' | 'x' }

Specifies a single character to be the field-name-start character for free-form input for built-in functions. The default is a blank.

The character specified should not be part of a transaction identification; in particular, the use of alphabetic characters is not recommended.

The character specified in the FLDSTRT parameter must not be the same as any character specified in the FLDSEP parameter. This means that it is invalid to allow both parameters to take the default value.

Restrictions:

- If you specify FLDSTRT in the SIT, the parameter must be enclosed in single quotation marks.
- If you specify FLDSTRT as a PARM, SYSIPT, or CONSOLE parameter, do **not** enclose the character in quotation marks, and the character you choose must not be a blank or any of the following characters:

```
( ) ' = ,
```

FSSTAFF={YES|NO}

specify this parameter in an application-owning region (AOR) to prevent transactions initiated by function-shipped EXEC CICS START requests being started against incorrect terminals.

You may need to code the function-shipped START affinity (FSSTAFF) parameter in an AOR if all of the following are true:

1. The AOR is connected to two or more terminal-owning regions (TORs) that use the same, or a similar, set of terminal identifiers.
2. One or more of the TORs issues EXEC CICS START requests for transactions in the AOR.
3. The START requests are associated with terminals.
4. You are using shippable terminals, rather than statically defining remote terminals in the AOR.

Consider the following scenario:

Terminal-owning region TOR1 issues an EXEC CICS START request for transaction TRAR, which is owned by region AOR1. It is to be run against terminal T001. Meanwhile, terminal T001 on region

TOR2 has been transaction routing to AOR1; a definition of T001 has been shipped to AOR1 from TOR2. When the START request arrives at AOR1, it is shipped to TOR2, rather than TOR1, for transaction routing from terminal T001.

To prevent this situation, code YES on the FSSTAFF parameter in the AOR.

YES

When a START request is received from a terminal-owning region, and a shipped definition for the terminal named on the request is already installed in the AOR, the request is always shipped back to a TOR, for routing, *across the link it was received on*, unless the request supplies a TOR_NETNAME and a remote terminal with the correct TOR_NETNAME is located.

If the TOR to which the START request is returned is **not** the one referenced in the installed remote terminal definition, a definition of the terminal is shipped to the AOR, and the autoinstall user program is called. Your autoinstall user program can then allocate an *alias* termid in the AOR, to avoid a conflict with the previously installed remote definition. For information about writing an autoinstall program to control the installation of shipped definitions, see the [CICS Customization](#).

NO

When a START request is received from a terminal-owning region, and a shipped definition for the named terminal is already installed in the AOR, the request is shipped to the TOR referenced in the definition, for routing.

Note:

1. FSSTAFF has no effect:
 - On statically-defined (hard-coded) remote terminal definitions in the AOR. If you use these, START requests are always shipped to the TORs referenced in the definitions.
 - On START requests issued in the local region. It affects only START requests shipped from other regions.
 - When coded on intermediate regions in a transaction-routing path. It is effective only when coded on an application-owning region.
2. If the AOR contains no remote definition of a terminal named on a shipped START request, the “terminal not known” global user exits, XICTENF and XALTENF, are called. For details of these exits, see the [CICS Customization](#).

GMTEXT={'WELCOME TO CICS' | 'text'}

Specifies whether the default logon message text (WELCOME TO CICS) or your own message text is to be displayed on the screen by the CSGM (good morning) transaction when a terminal is logged on to CICS through VTAM, by the CESN transaction if used to sign on to CICS, or by your own transactions using the EXEC CICS INQUIRE SYSTEM GMMTEXT command.

You can use apostrophes to punctuate your message, in addition to using them as message delimiters. However, you must code *two* successive apostrophes to represent a single apostrophe in your text. For example,

```
GMTEXT='User''s logon message text.'
```

The whole message must still be enclosed by a pair of single delimiting apostrophes.

Your message text can be from 1 through 246 characters (bytes), and can extend over two lines by extending the text to column 80 on the first line, and continuing in column 1 of the second line. For example, the following might be used in the SYSIPT data set:

```
GMTEXT='An Information Development CICS Terminal-Owning Region (TOR) - CICSIDC.  
This message is to show the use of continuation lines when creating a GMTEXT par  
ameter in the SYSIPT data set'
```

The CSGM transaction displays this as follows (with the time appended to the end of message):

An Information Development CICS Terminal-Owning Region (TOR) - CICSIDC. This message is to show the use of continuation lines when creating a GMTEXT parameter in the SYSIPT data set 09:56:14

The CESN transaction displays this as follows:

Signon to CICS	APPLID CICSHTH1
An Information Development CICS Terminal-Owning Region (TOR) - CICSIDC. This message is to show the use of continuation lines when creating a GMTEXT parameter in the SYSIPT data set	

For any transaction other than CESN that displays the text specified by this parameter, you must use a RDO TYPETERM with LOGONMSG(YES) for all terminals requiring the logon message. For information about using RDO TYPETERM, see the [CICS Resource Definition Guide](#).

GMTRAN={CSGM|CESN|transaction-id}

Specifies the name of the transaction that is initiated when terminals are logged on to CICS by VTAM. Do not specify the name of a remote transaction. The transaction must be capable of being automatically initiated (ATI). The default is the transaction CSGM, that displays the text specified in the GMTEXT parameter. Alternatively, you can specify the CICS signon transaction, CESN, which also displays the text specified in the GMTEXT parameter. The GMTRAN parameter can be used with the LGNMSG parameter to retrieve VTAM logon data.

GNTRAN={CESF|transaction_id}

Specifies the transaction that you want CICS to invoke when a user's terminal-timeout period expires.

CESF

The default, CESF, is the basic CICS signoff transaction without any options. This transaction attempts to sign off a terminal, subject to the SIGNOFF attribute of the RDO TYPETERM resource definition for that terminal.

transaction_id

The name of an alternative timeout transaction to signoff the user at the timed-out terminal. Specifying your own transaction allows you to specify functions in addition to, or instead of, signoff. For example, your own transaction could issue a prompt for the terminal user's password, and allow the session to continue if the correct password is entered.

The transaction to be used must have been specially written to handle the GNTRAN commarea that is passed to it. Of the CICS-supplied transactions, only CESF has been written to handle the GNTRAN commarea. For more information about writing your own transactions for GNTRAN, see the [CICS Customization Guide](#).

Note: When either the default CESF transaction, or your own transaction, attempts to sign off a terminal, the result is subject to the SIGNOFF attribute of the RDO TYPETERM resource definition for the terminal, as follows:

SIGNOFF

Effect

YES

The terminal is signed off, but not logged off.

NO

The terminal remains signed on and logged on.

LOGOFF

The terminal is both signed off and logged off.

Note: If GNTRAN fails to attach, and SIGNOFF(LOGOFF) has been specified, the terminal which has reached timeout will be signed off and logged off. GNTRAN will not run and will have no effect.

GRPLIST={DFHLIST|name|(name[,name2][,name3][,name4])}

Specifies the names (each 1 through 8 characters) of up to four lists of resource definition groups on the CICS system definition (CSD) file. The resource definitions in all the groups in the specified lists are loaded during initialization when CICS performs a cold start. If a warm or emergency start

is performed, the resource definitions are derived from the CICS global catalog, and the GRPLIST parameter is ignored.

Each name can be either a real group list name or a generic group list name that incorporates global filename characters (+ and *). If you specify more than one group list (either by specifically coding two or more group list names or by coding a group list name with global filename characters), the later group lists are concatenated onto the first group list. Any duplicate resource definitions in later group lists override those in earlier group lists.

Use the CEDA command LOCK to protect the lists of resource groups specified on the GRPLIST parameter.

The default is DFHLIST, the CICS-supplied list that specifies the set of resource definitions needed by CICS. If you create your own group list, either add to it the groups specified in DFHLIST (omitting only those for CICS functions that you know you do not need) or specify the DFHLIST name on the GRPLIST parameter. Do not code GRPLIST=NO unless you have a group list named NO.

Note:

1. Group lists specified by a generic group list name are concatenated in alphabetic then numeric order. For example, the generic list name CICSHT*, would concatenate the group lists CICSHT#1, CICSHTAP, CICSHTSD, and CICSHT3V in that order. If the order of concatenation is important (for example, to ensure that a particular resource definition overrides another), you should consider coding real group list names.
2. If a group list contains resource definitions that are needed by another group list, the prerequisite group list must be installed first. For example, if list A has TYPETERM definitions needed for TERMINAL definitions in list B, list A must be installed first. This may mean that you have to specifically name the prerequisite group on the GRPLIST parameter.
3. Take care when using generic group list names because, if a group list on your CSD satisfies the generic name, it will be installed. This means that a group list can be installed more than once; for example, if you specify the real group list name and a generic group list name that it satisfies, or if you specify two generic group list names that the group list name satisfies.
4. To override one or more of the group lists specified on the GRPLIST system initialization parameter, you must specify all list names (both real and generic) that you want to use, even if you are not changing the names.

For example, if you want to use the four group lists CICSHT#1, CICSHTAP, CICSHT3V, and CICSHTSD, you could specify either of the following system initialization parameters:

```
GRPLIST=(CICSHT#1,CICSHTAP,CICSHT3V,CICSHTSD)
GRPLIST=(CICSHT*)
```

In the first example GRPLIST, the group lists are loaded in the order specified, and resource definitions installed from the CICSHTSD group list will override any duplicate definitions installed by the other groups.

In the second example GRPLIST, the group lists are loaded in the order CICSHT#1, CICSHTAP, CICSHTSD, then CICSHT3V, and resource definitions installed from the CICSHT3V group list will override any duplicate definitions installed by the other groups.

If your SIT contains the parameter:

```
GRPLIST=(CICSHT#1,CICSAP*,CICSHT3V,CICSHTSD)
```

and you want to replace the list CICSHT3V with the list ANOLST05, you should specify the override:

```
GRPLIST=(CICSHT#1,CICSAP*,ANOLST05,CICSHTSD)
```

In general, any required resource definitions should appear in *one of* the group lists specified on the GRPLIST system initialization parameter.

For information about resource definitions, groups, lists, and the CSD, see the [CICS Resource Definition Guide](#).

ICP=COLD

Code this parameter if you want to cold start the interval control program. For more information, refer to “Notes on CICS resource table and module keywords” on page 307.

ICV={1000|number}

Code this parameter with the region exit time interval in milliseconds. The ICV system initialization parameter specifies the maximum time in milliseconds that CICS releases control to the operating system when there are no transactions ready to resume processing. This time interval can be any integer in the range 100 through 3600000 milliseconds (specifying an interval up to 60 minutes). A typical range of operation might be 100 through 2000 milliseconds.

A low value interval can enable much of the CICS nucleus to be retained in dynamic storage, and not be paged-out at times of low terminal activity. This reduces the amount of dynamic storage paging necessary for CICS to process terminal transactions (thus representing a potential reduction in response time), sometimes at the expense of concurrent batch region throughput. Large networks with high terminal activity are inclined to run CICS without a need for this value, except to handle the occasional, but unpredictable, period of inactivity. These networks can usually function with a large interval (10000 to 3600000 milliseconds). Once a task has been initiated, its requests for terminal services and the completion of the services are recognized by the system and this maximum delay interval is overridden.

Small systems, or those with low terminal activity, are subject to paging introduced by other jobs running in competition with CICS. By specifying a low value interval, key portions of the CICS nucleus are referenced more frequently, thus reducing the probability of these pages being paged-out. However, the execution of the logic without performing productive work might be considered wasteful. The need to increase the probability of residency by frequent but unproductive referencing must be weighed against the overhead and response time degradation incurred by allowing the paging to occur. By increasing the interval size, less unproductive work is performed at the expense of performance if paging occurs during the periods of CICS activity. For information about the effect of ICV on performance, see the [CICS Performance Guide](#).

Note: The region exit time interval process contains a mechanism to ensure that CICS does not constantly set and cancel timers (thus degrading performance) while attempting to meet its objectives for a low region exit time interval. This mechanism can cause CICS to release control to the operating system for up to 0.5 seconds when the interval has been set at less than 250; and up to 0.25 seconds more than the region exit time interval when the interval has been set greater than 250.

ICVR={5000|number}

Specifies the default runaway task time interval in milliseconds as a decimal number. You can code zero, or a number in the range 500 through 2 700 000, in multiples of 500. CICS rounds down values that are not multiples of 500. This is the RUNAWAY interval used by transactions defined with RUNAWAY(SYSTEM) (see the [CICS Resource Definition Guide](#)). CICS may purge a task if it has not given up control after the RUNAWAY interval for the transaction (or ICVR if the transaction definition specified RUNAWAY(SYSTEM)). If you code ICVR=0, runaway task control is inoperative for transactions specifying RUNAWAY(SYSTEM) in their RDO TRANSACTION definition (that is, tasks do not get purged if they appear to be looping). The ICVR value is independent of the ICV value, and can be less than the ICV value. Note that CICS runaway task detection is based upon task time, that is, the interval is decremented only when the task has control of the processor. For information about commands that reinitialize the ICVR value, see the [CICS Problem Determination Guide](#).

ICVTSD={500|number}

The terminal scan delay facility determines how quickly CICS deals with some terminal I/O requests made by applications. The range is 0 through 5000 milliseconds, with a default of ICVTSD=500.

There is an overhead in dealing with such requests. By specifying a nonzero value, the overhead may be spread over several transactions. A value close to zero (for example 200) would be adequate.

INITPARM=(pgmname_1='parmstring_1', ..., pgmname_n='parmstring_n')

Code this parameter in order to pass parameters to application programs that use the EXEC CICS ASSIGN INITPARM command. For example, you can use INITPARM to pass parameters to PLTPI programs to be executed in the final stages of system initialization. The area giving access to the parameters is specified by the EXEC CICS ASSIGN INITPARM command. For

programming information about the EXEC CICS ASSIGN INITPARM command, see the [CICS Application Programming Reference](#) publication.

pgmname

Code pgmname with the name of a program. This name must be 1 through 8 alphanumeric or national language characters.

parmstring

Code parmstring with the parameter string (up to 60 characters enclosed by single quotation marks) to be passed to the associated program. Any quotation marks imbedded in the string must be duplicated. For information on coding INITPARM in the SYSIPT data set, see [“Rules for coding parameters at the console” on page 307](#).

You can specify up to 255 pgmname='parmstring' sets.

INTTR={ON|OFF}

Code this parameter to specify whether the internal CICS trace destination is to be activated at system initialization.

This parameter controls whether any of the three types of CICS trace entry are written to the internal trace table. The three types are: CICS system trace (see the SYSTR parameter), user trace (see the USERTR parameter), and exception trace entries (which are always made and not controlled by a system initialization parameter).

ON

Activate main storage trace.

OFF

Do not activate main storage trace.

IRCSTRT={NO|YES}

Code this parameter to indicate whether IRC is started up at system initialization. If IRCSTRT=YES is not coded, IRC can be initialized by issuing a CEMT or EXEC CICS SET IRC OPEN command.

ISC={NO|YES}

Code YES to include the CICS programs required for interregion or intersystem communication.

JCT={YES|xx|NO}

Code this parameter with the suffix of the journal control table to be used. For more information, refer to [“Notes on CICS resource table and module keywords” on page 307](#). This indicates whether journaling and volume control are to be used. Note that you must have journaling if you code XRF=YES. If you code JCT=NO, a dummy journal control program (DFHJCPDY) is loaded.

For information about coding the macros for this table, see the [CICS Resource Definition Guide](#).

JSTATUS=RESET

Code JSTATUS=RESET to reset the journal status of all journal data sets that are defined on disk (JTYPE=DISK1|DISK2), but are *not* specified with JOUROPT=AUTOARCH.

This parameter resets the journal status held in the global catalog to “ready for use”. Before using the JSTATUS option at startup, ensure that any disk journals that contain essential information are archived, either to tape or another disk data set.

Restrictions: You can code the JSTATUS parameter in PARM, SYSIPT, or CONSOLE only. It is not applicable if you are using the CICS automatic archiving facility.

KEYFILE=name

Specifies the name of the VSE Keyring Library used for storing keys and certificates. *Your TCP/IP and CICS regions must be authorized to read the VSE Keyring Library specified here.* The IBM-supplied VSE Keyring Library has the name CRYPTO.KEYRING. Therefore the KEYFILE parameter would be coded as:

```
KEYFILE=CRYPTO.KEYRING
```

The *library* name (in the above example, CRYPTO) can have a maximum of 7 characters. For further details, refer to "Preparing Your System to Use SLL" in the publication [z/VSE Administration](#), SC34-2692.

LEVSE={YES|NO}

Specifies whether you want to initialize the language environment under CICS Transaction Server for z/VSE. If you specify NO, you will not be able to execute and PL/I for VSE/ESA, COBOL for VSE/ESA or C for VSE/ESA programs during this execution of CICS.

LGNMSG={NO|YES}

Code this to indicate whether VTAM logon data is made available to an application program.

NO

VTAM logon data is not available to an application program.

YES

VTAM logon data is available to an application program. The data can be retrieved with an EXEC CICS EXTRACT LOGONMSG command. For programming information about this command, see the [CICS Application Programming Reference](#) publication.

You can use this parameter with the GMTRAN parameter to retrieve the VTAM logon data at the time a terminal is logged on to CICS by VTAM.

LOCALCCSID={037|CCSID}

Specifies the default CCSID for the local region. The CCSID is a value of up to 8 characters. If CCSID value is not specified, the default LOCALCCSID is set to 037.

MCT={NO|YES|xx}

Code this parameter to specify the monitoring control table suffix. For more information, refer to ["Notes on CICS resource table and module keywords"](#) on page 307.

If you specify MCT=NO, CICS monitoring builds dynamically a default MCT, ensuring that default monitoring control table entries are always available for use when monitoring is on and a monitoring class (or classes) is active.

For information about coding the macros for this table, see the [CICS Resource Definition Guide](#).

MN={OFF|ON}

Code this parameter to indicate whether monitoring is to be switched on or off at initialization, and use the individual monitoring class parameters to control which monitoring classes are to be active. (See the MNEXC, and MNPER parameter descriptions.) The default status is that the CICS monitoring facility is *off*. The monitoring status is recorded in the CICS global catalog for use during warm and emergency restarts.

OFF

Switch off monitoring.

ON

Switch on monitoring. However, unless at least one individual class is active, no monitoring records are written. For details of the effect of monitoring status being on or off, in conjunction with the status of the various monitoring classes, see the following notes:

Note: If the monitoring status is ON, CICS accumulates monitoring data continuously. For the performance and exception monitoring classes, CICS writes the monitoring data for each class that is active to a CICS Data Management Facility (DMF) data set.

If the monitoring status is OFF, CICS does not accumulate or write any monitoring data, even if any of the monitoring classes are active.

You can change the monitoring status and the monitoring class settings at any time, as follows:

- During a warm restart by coding an MN system initialization parameter in PARM, SYSIPT, or through the system console.
- While CICS is running, by either of:
 - The CEMT SET MONITOR command

- The EXEC CICS SET MONITOR command.

When you change the status of monitoring, the change takes effect immediately. If you change the monitoring status from OFF to ON, monitoring starts to accumulate data and write monitoring records to DMF for all tasks that start after the status change is made **for all active monitoring classes**.

If the status is changed from ON to OFF, monitoring stops writing records immediately and does not accumulate monitoring data for any tasks that start after the status change is made.

The monitoring status operand can be manipulated independently of the class settings. This means that, even if the monitoring status is OFF, you can change the monitoring class settings and the changes take effect for all tasks that are started after the monitoring status is next set to ON.

For programming information about controlling CICS monitoring, see the [CICS System Programming Reference](#) publication.

MNCONV={NO|YES}

Specifies whether or not conversational tasks are to have separate performance class records produced for each pair of terminal control I/O requests.

Any clock (including user-defined) that is active at the time such a performance class record is produced is stopped immediately before the record is written. After the record is written, such a clock is reset to zero and restarted. Thus a clock whose activity spans more than one recording interval within the conversational task appears in multiple records, each showing part of the time, and the parts adding up to the total time the clock is active. The high-water-mark fields (which record maximum levels of storage used) are reset to their current values. All other fields are set to X'00', except for the key fields (transid, termid). The monitoring converse status is recorded in the CICS global catalog for use during warm and emergency restarts.

MNEXC={OFF|ON}

Code this parameter to indicate whether the monitoring exception class is to be made active during initialization. The monitoring exception class status is recorded in the CICS global catalog for use during warm and emergency restarts.

OFF

Set the exception monitoring class to "not active".

ON

Set the exception monitoring class to "active".

For programming information about exception monitoring records, see the [CICS Customization Guide](#).

MNFREQ={@|hhmmss}

Specifies the interval for which CICS automatically produces a transaction performance class record for any long-running transaction. The monitoring frequency value is recorded in the CICS global catalog for use during warm and emergency restarts.

0

means that no frequency monitoring is active.

hhmmss

is the interval for which monitoring produces automatically a transaction performance class record for any long-running transaction. Specify a 1 to 6 digit number in the range 001500–240000. Numbers that are fewer than six digits are padded with leading zeroes.

MNPER={OFF|ON}

Code this parameter to indicate whether the monitoring performance class is to be made active during CICS initialization. The monitoring performance class status is recorded in the CICS global catalog for use during warm and emergency restarts.

OFF

Set the performance monitoring class to "not active".

ON

Set the performance monitoring class to "active".

For programming information about performance monitoring records, see the [CICS Customization Guide](#).

MNSYNC={NO|YES}

Specifies whether or not you want CICS to produce a transaction performance class record when a transaction takes an implicit or explicit syncpoint (unit-of-work). No action is taken for syncpoint rollbacks. The monitoring syncpoint status is recorded in the CICS global catalog for use during warm and emergency restarts.

MNTIME={GMT|LOCAL}

Specifies whether you want the time stamp fields in the performance class monitoring data to be returned to an application using the EXEC CICS COLLECT STATISTICS MONITOR(taskno) command in either GMT or local time. The monitoring time value is recorded in the CICS global catalog for use during warm and emergency restarts.

For programming information on the EXEC CICS COLLECT STATISTICS command, see the [CICS System Programming Reference](#).

MROBTCH={1|number}

Code this parameter to specify the number of events that must occur before CICS is posted for dispatch due to the batching mechanism. The number can be in the range 1 through 255, and the default is 1.

Use this batching mechanism to spread the overhead of dispatching CICS over several tasks. If the value is greater than 1 and CICS is in a system wait, CICS is not posted for dispatch until the specified number of events has occurred. Events include MRO requests from connected systems or DASD I/O. For these events, CICS is dispatched as soon as one of the following occurs:

- The current batch fills up (the number of events equals MROBTCH).
- An ICV interval expires.

Therefore, ensure that the time interval you specify in the ICV parameter is low enough to prevent undue delay to the system.

If CICS is dispatched for another reason, the current batch is dealt with in that dispatch of CICS.

Note: During periods of low utilization, a value greater than 1 specified on the MROBTCH parameter may result in increased transaction response times. Transactions issuing file I/O requests may be delayed due to increased FCIOWAIT. For further guidance information about the effect of MROBTCH on performance, see the [CICS Performance Guide](#).

MROFSE={NO|YES}

specifies whether you want to extend the lifetime of the long-running mirror to keep it allocated until the end of the task rather than after a user syncpoint for Function Shipping applications.

NO

The lifetime of the MRO long-running mirror is not extended.

YES

The mirror task remains available to the application until the end of the application's task. This extended long-running mirror saves the overhead of re-attaching the mirror task following a user syncpoint.

This parameter is ignored for DPL requests (that is a DPL causes the session to be freed at the next syncpoint even if it has been kept for a previous sequence of syncpoints).

It should be used with caution. For additional information, see the Long Running Mirror sections of the *CICS Intercommunication Guide* and the [CICS Performance Guide](#).

MROLRM={NO|YES}

Code this parameter to specify whether you want to establish an MRO long-running mirror task.

NO

The MRO long-running mirror task is not required.

YES

The mirror transaction remains available to the application issuing the remote request. This long-running mirror saves the overhead of re-establishing communication with the mirror transaction if the application makes more function shipping requests in this unit of work.

For information about long-running mirror tasks, see the [CICS Intercommunication Guide](#).

MSGCASE={MIXED|UPPER}

CICS messages handled by the CICS message domain are in mixed case. Code the MSGCASE parameter to indicate how you want the message domain to display these mixed case messages.

MIXED

This is the default in the SIT; all messages displayed by the CICS message domain remain in mixed case,

UPPER

The message domain displays all mixed case messages in uppercase only.

Note: Mixed case output is not displayed correctly on Katakana display terminals and printers. Uppercase English characters appear correctly as uppercase English characters, but lowercase appears as Katakana symbols. If you have any Katakana terminals connected to your CICS region, specify MSGCASE=UPPER.

MSGLVL={1|0}

Code this parameter with the message level that controls the generation of messages to the console.

1

All messages are to be printed.

0

Only critical errors or interactive messages are to be printed.

MXT={5|number}

Specifies the maximum number, in the range 1 through 999, of *user* tasks CICS allows to exist at any time. CICS queues requests for tasks above this number but does not action (attach) them until the number of tasks attached drops below the MXT limit.

Note: The MXT value does **not** include CICS system tasks.

NATLANG=(E,x,y,z,...)

Specify on this parameter the single-character codes for the languages to be supported in this CICS run. For more information see [Figure 14 on page 161](#).

E

Code E for English, which is the *system* default (that is, is provided even if you do not specifically code E).

x,y,z,...

Code the appropriate letters for the other supported languages that you require.

For the codes that you specify on this parameter, you must ensure that a DFHMET1x module (where x is the language code) is in a sublibrary in the LIBDEF PHASE,SEARCH chain for the CICS job.

English language support is provided, even if you do not specifically code E for English.

The first language code specifies the default language for those elements of CICS enabled to receive NLS messages, such as some destinations used for CICS messages, and the terminals or users not signed-on with an NLS code. The other language codes are provided to specify the language to be used for messages sent to terminals that are defined with the appropriate language support code.

For example, coding NATLANG=(G,C,K) has the same effect as coding NATLANG=(G,C,E,K); that is, in both cases the default NLS language is German (G), and the languages English, Chinese (C), and Kanji (K) are supported.

CICS console messages are available in English and German only.

NEWSIT={YES|NO}

Specify NEWSIT=YES to cause CICS to load the specified SIT, and enforce the use of all system initialization parameters, modified by any system initialization parameters provided via PARM, SYSIPT, or the system console, even in a warm start. Enforcing the use of system initialization parameters in this way overrides any parameters that may have been stored in a warm keypoint at shutdown.

However, there are some exceptions; the following system initialization parameters are always ignored in a warm start, even if they are supplied via PARM, SYSIPT, or the console:

CSDACC
 CSDBUFND
 CSDBUFNI
 CSDFRLOG
 CSDJID
 CSDLRNO
 CSDRECOV
 CSDSTRNO
 FCT
 GRPLIST

In a warm restart CICS uses the *installed* resource definitions saved in the CICS global catalog at warm shutdown, and therefore the CSD, FCT, and GRPLIST parameters are ignored. (At CICS startup, you can only modify installed resource definitions, including file control table entries, or change to a new FCT, by performing a cold start of CICS with START=COLD.)

For more information about the use of the NEWSIT parameter, see [“Classes of start and restart” on page 310](#).

Restrictions: You can code the NEWSIT parameter in PARM, SYSIPT, or CONSOLE only.

OPERTIM={120|number}

Code this parameter with the write-to-operator timeout value, in the range 0 through 86400 seconds (24 hours). This is the maximum time (in seconds) that CICS waits for a reply before returning control to this transaction. For information about using the write-to-operator timeout value, see the [CICS Application Programming Reference](#) publication.

PARMERR={INTERACT|IGNORE|ABEND}

Code this parameter to specify what action you want to follow if CICS detects incorrect system initialization parameter overrides during initialization.

Note: When specified as an override, this parameter affects only subsequent system initialization parameter overrides. Errors in earlier system initialization parameter overrides are dealt with according to the PARMERR system initialization parameter value in the SIT.

INTERACT

Enables the operator to communicate with CICS via the console and correct parameter errors.

Note: INTERACT is overridden with IGNORE in the following cases:

- If errors are found in PARM or SYSIPT for system initialization parameter overrides that are not allowed to be entered from the console
- In certain circumstances, in response to invalid data when you have been trying to correct a previous invalid system initialization parameter keyword or value.

IGNORE

CICS ignores errors, and tries to complete initialization.

ABEND

CICS abends.

PDI={30|decimal-value}

Use this parameter in a SIT for an active CICS region. It specifies the XRF primary delay interval, in seconds. The minimum delay that you can specify is 5 seconds. This is the time that must elapse

between the (apparent) loss of the surveillance signal in the alternate CICS region, and any reaction by the active CICS region. The corresponding parameter for the alternate CICS region is ADI. PDI and ADI need not have the same value.

PGAICTLG={MODIFY|NONE|ALL}

Specifies whether autoinstalled program definitions should be cataloged. While CICS is running, you can set whether autoinstalled programs should be cataloged dynamically, by using either the EXEC CICS SET SYSTEM or CEMT SET SYSTEM command.

MODIFY

Autoinstalled program definitions are cataloged only if the program definition is modified by a SET PROGRAM command subsequent to the autoinstall.

NONE

Autoinstalled program definitions are not cataloged. This gives a faster CICS restart (warm and emergency) compared with the MODIFY or ALL options, because CICS does not reinstall definitions from the CICS global catalog (DFHGCD). Definitions are autoinstalled on first reference.

ALL

Autoinstalled program definitions are written to the global catalog at the time of the autoinstall, and following any subsequent modification.

PGAIXIT={DFHPGADX|name}

Specifies the name of the program autoinstall exit program. While CICS is running, you can set the name of the program autoinstall exit program dynamically, by using either the EXEC CICS SET SYSTEM or CEMT SET SYSTEM command.

PGAIPGM={INACTIVE|ACTIVE}

Specifies the state of the program autoinstall function at initialization. While CICS is running, you can set the status of program autoinstall dynamically, by using either the EXEC CICS SET SYSTEM or CEMT SET SYSTEM command.

INACTIVE

The program autoinstall function is disabled.

ACTIVE

The program autoinstall function is enabled.

PGCHAIN=character(s)

Code this parameter with the character string that is identified by terminal control as a BMS terminal page-chaining command. It can be 1 through 7 characters. For more information about the character string, see the notes under the PGRET parameter.

PGCOPY=character(s)

Code this parameter with the character string that is identified by terminal control as a BMS command to copy output from one terminal to another. It can be 1 through 7 characters. For more information about the character string, see the notes under the PGRET parameter.

PGPURGE=character(s)

Code this parameter with the character string that is identified by terminal control as a BMS terminal page-purge command. It can be 1 through 7 characters. For more information about the character string, see the notes under the PGRET parameter.

PGRET=character(s)

The character string that is recognized by terminal control as a BMS terminal page-retrieval command. It can be 1 through 7 characters.

Note:

1. Each character string is unique with respect to the leading characters of every other transaction identification defined in the CSD. A command requested by a single character precludes the use of all other transaction identifications starting with this character.
2. In pseudoconversational mode, each character string is unique with respect to the leading characters of any terminal input message.

3. A field-separator or other suitable delimiter may be specified in each character string to separate this command code from the remainder of the paging command when entered by an operator. For example:

```
PGCHAIN = X/
PGCOPY = C/
PGPURGE = T/
PGRET = P/
```

This reduces the risk of creating a nonunique command. (See Note 1.)

Restrictions:

- If you specify PGCHAIN, PGCOPY, PGPURGE, or PGRET in the SIT, the characters you choose must not include any of the following: () ' =
 - If you specify PGCHAIN, PGCOPY, PGPURGE, or PGRET as a PARM, SYSIPT, or console parameter, do not enclose the characters in quotation marks. The characters you choose must not include an embedded blank or any of the following: () ' =
4. PGCHAIN, PGCOPY, PGPURGE, and PGRET are required only if full function BMS is being used. For information about the BMS page retrieval transaction CSPG, see the [CICS Supplied Transactions](#) publication.
5. CICS always processes a paging command entered by the operator before initiating a transaction in response to a macro request, that consists of either DFHBMS or DFHPC with the TRANSID operand coded.

PLTPI={NO|xx|YES}

Code this parameter with a program list table that contains a list of programs to be executed in the final stages of system initialization. For more information, refer to [“Notes on CICS resource table and module keywords”](#) on page 307.

You can use the system initialization parameter INITPARM to pass parameters to those programs.

For information about coding the macros for this table, see the [CICS Resource Definition Guide](#) publication.

PLTPISEC={NONE|CMDSEC|RESSEC|ALL}

Specifies whether or not you want CICS to perform command security or resource security checking for PLT programs during CICS initialization. The PLT programs run under the authority of the userid specified on PLTPIUSR, which must be authorized to the appropriate resources defined by PLTPISEC.

NONE

Specifies that you do not want any security checking on on PLT initialization programs.

CMDSEC

Specifies that you want CICS to perform command security checking only.

RESSEC

Specifies that you want CICS to perform resource security checking only.

ALL

Specifies that you want CICS to perform both command and resource security checking.

Restrictions: You can code the PLTPISEC parameter in the SIT, PARM, or SYSIPT only.

PLTPIUSR=userid

Specifies the userid that CICS is to use for security checking for PLT programs that run during CICS initialization. All PLT programs run under the authority of the specified userid, which must be authorized to all the resources referenced by the programs, as defined by the PLTPISEC parameter.

PLT programs are run under the CICS internal transaction, CPLT. Before the CPLT transaction is attached, CICS performs a surrogate user check against the CICS region userid (the userid under which the CICS region is executing). This is to ensure that the CICS region is authorized as a surrogate for the userid specified on the PLTPIUSR parameter. This ensures that you cannot arbitrarily specify any PLT userid in any CICS region—each PLT userid must first be authorized to the appropriate CICS region.

If you do not specify the PLTPIUSR parameter, CICS runs PLTPI programs under the authority of the CICS region userid, in which case CICS does not perform a surrogate user check. However, the CICS region userid must be authorized to all the resources referenced by the PLT programs.

Restrictions: You can code the PLTPIUSR parameter in the SIT, PARM, or SYSIPT only.

PLTSD={NO|xx|YES}

Code this parameter with a program list table that contains a list of programs to be executed during system termination. For more information, refer to [“Notes on CICS resource table and module keywords” on page 307](#).

PRGDLAY={0|hhmm}

Code this parameter with the BMS purge delay time interval that is added to the specified delivery time to determine when a message is to be considered undeliverable and therefore purged. This time interval is coded in the form “hhmm” (where “hh” represents hours from 00 to 99 and “mm” represents minutes from 00 to 59). If PRGDLAY is not coded, or is given a zero value, a message remains eligible for delivery either until it is purged or until temporary storage is reinitialized.

Note: If you specify PRGDLAY as a SIT override, you must still specify a 4-character value (for example 0000).

The PRGDLAY facility requires the use of full function BMS. Note also that you must code a PRGDLAY value if you want the ERRTERM|ERRTERM(name) parameter on EXEC CICS ROUTE commands to be operative. For programming information about notification of undelivered messages, see the [CICS Application Programming Reference](#) publication.

The PRGDLAY value determines the interval between terminal page clean-up operations. A very low value causes the CSPQ transaction to be initiated continuously, and can have a detrimental effect on task-related resources.

A zero value stops CSPQ initiating terminal page clean-up. However, this can cause messages to stay in the system forever, resulting in performance problems with long AID queues or lack of temporary storage. The actual purge delay time interval specified is dependent on individual system requirements.

PRINT={NO|YES|PA1|PA2|PA3}

Code this parameter with the method of requesting printout of the contents of a 3270 screen.

NO

Screen copying is not required.

YES

Screen copying can be requested by terminal control print requests only.

PA1, PA2, or PA3

Screen copying can be requested by terminal control print request, or by using the PA (program attention) key specified.

The PA key specified by this parameter must not be specified by the TASKREQ option of the RDO TRANSACTION definition or be used for 3270 single keystroke retrieval.

When YES, PA1, PA2, or PA3 is specified, transaction CSPP is initiated which invokes program DFHP3270. The transaction and programs are defined in the CSD group DFHHARDC. In the case of 3270 and LUTYPE2 logical units, the resources defined in CSD group DFHVTAMP are required.

The 3270 print-request facility allows either the application program or the terminal operator to request a printout of data currently displayed on the 3270 display.

If CSPP is invoked to print the screen contents at an associated VTAM printer, the screen size of the printer is chosen according to the screen size defined in the profile for the transaction CSPP. The CICS-supplied definitions use the default screen size. Therefore, if you want DFHP3270 to use the alternate screen size of the printer, you must alter the screen size defined in the profile for the transaction CSPP. For information about defining profiles for transactions, see the [CICS Resource Definition Guide](#).

For a VTAM 3270 display without the printer-adapter feature, the PRINT request prints the contents of the display on the first available 3270 printer specified by PRINTER and ALTPRINTER options of the RDO TERMINAL definition. For a printer to be considered available, it must be in service and not currently attached to a task. It is not necessary for the printer to be on the same control unit.

In an MRO environment, the printer must be owned by the same system as the VTAM 3270 display.

For the 3275 with the printer-adapter feature, the PRINT request prints the data currently in the 3275 display buffer on the 3284 Model 3 printer attached to the 3275.

The format of the print operation depends on the size of the display buffer. For a 40-character wide display, the print format is a 40-byte line, and for an 80-character wide display the format is an 80-byte line.

For the 3270 compatibility mode logical unit of the 3790 (if the logical unit has the printer-adapter feature specified), the PRINT request prints the contents of the display on the first printer available to the 3790. The allocation of the printer to be used is under the control of the 3790.

For 3274, 3276, and LUTYPE2 logical units with the printer-adapter feature, the PRINT request prints the contents of the display on the first printer available to the 3270 control unit. The printer to be allocated depends on the printer authorization matrix. For information, refer to the *3270 Information Display System Component Description* manual.

For the 3270 compatibility mode logical unit without the printer-adapter feature, see the preceding paragraph on VTAM 3270 displays without the printer-adapter feature.

PRTYAGE={32768|value}

Code this parameter with the number of milliseconds to be used in the priority aging algorithm for incrementing the priority of a task. The value can be in the range 0 through 65535, and 32768 is the default.

The priority aging factor is used to increase the effective priority of a task according to the amount of time it is held on a ready queue. The value represents the number of milliseconds that must elapse before the priority of a waiting task can be adjusted upwards by 1. For example, if you code PRTYAGE=3000, a task has its priority raised by 1 for every 3000 milliseconds it is held on the ready queue. Thus a high value for PRTYAGE results in a task being promoted very slowly up the priority increment range, and a low value enables a task to have its priority incremented quickly.

If you specify a value of 0, the priority aging algorithm is not used (task priorities are not modified by age) and tasks on the ready queue are handled according to the user assigned priority.

PRVMOD={name|(name,name...name)}

Code the PRVMOD parameter with the name of those modules that are not to be used from the VSE shared virtual area (SVA).

The operand is a list of 1- to 8-character module names. This enables you to use a private version of a CICS nucleus module in the CICS address space, and not a version that might be in the SVA. For more information about preventing modules from being used from the SVA, see the [CICS System Definition Guide](#).

Restrictions: You can code the PRVMOD parameter in PARM, SYSIPT, or CONSOLE only.

PSDINT={0|hhmmss}

Specifies the persistent session delay interval. This delay interval specifies if, and for how long, VTAM is to hold sessions in a recovery-pending state if CICS fails. The value for hours can be in the range 0 through 23; the minutes and seconds in the range 00 through 59 inclusive.

This value can be overridden during CICS execution (and hence change the action taken by VTAM if CICS fails).

0

A zero value specifies that, if CICS fails, sessions are terminated. This is the default.

hhmss

Specifies a persistent session delay interval from 1 second up to the maximum of 23 hours 59 minutes and 59 seconds. If CICS fails, VTAM holds sessions in recovery pending state for up to the interval specified on the PSDINT system initialization parameter.

Specify a 1-to-6 digit time in hours, minutes and seconds, up to the maximum time. If you specify less than six digits, CICS pads the value with leading zeros. Thus a value of 500 is taken as five minutes exactly.

The interval you specify must cover the time from when CICS fails to when the VTAM ACB is opened by CICS during the subsequent emergency restart.

VTAM holds all sessions in recovery pending state for up to the interval specified (unless they are unbound through path failure or VTAM operator action, or other-system action in the case of intelligent LUs). The PSDINT value used must take account of the types and numbers of sessions involved.

You must exercise care when specifying large PSDINT values because of the problems they may give in some environments, in particular:

- Dial-up sessions—real costs may be incurred
- LU6.2 sessions to other host systems—such systems may become stressed.

Note:

1. When specifying a PSDINT value, you must consider the number and, more particularly, the nature of the sessions involved. If LU6.2 sessions to other host systems are retained in recovery pending state, the other host systems may experience excessive queuing delays. This point applies to LU6.1 sessions which are retained until restart (when they are unbound).
2. The PSDINT parameter is incompatible with the XRF=YES parameter. If XRF=YES is specified, the PSDINT parameter is ignored.

PVDELAY={30|number}

Code this with the persistent verification delay as a value in the range 0 through 10080 minutes (up to 7 days). PVDELAY defines how long entries can remain in the signed-on-from lists for those connections for which persistent verification is specified in a connection resource definition. If you specify PVDELAY=0, entries are deleted immediately after use.

For information about the use of PVDELAY, see the [CICS Performance Guide](#).

RAMAX={256|value}

Code this parameter with the size in bytes of the I/O area allocated for each RECEIVE ANY issued by CICS, in the range 0 through 32767 bytes.

Note:

1. If you are using APPC, do not code a value less than 256; otherwise, the results are unpredictable.
2. If you are using pipeline terminals, do not code a value that is less than the RUSIZE (from the CINIT), because pipelines cannot handle data longer than this.

For information about coding this parameter, see the [CICS Performance Guide](#).

RAPOOL={50|decimal-value}

Code this parameter to determine the size of the CICS receive any pool. It is the number of fixed request parameter lists (RPLs), receive any control elements (RACEs), and receive any input areas (RAIAs) that are to be generated.

Decimal-value can be in the range 1 through 999, and has a default of 2.

If you omit the RAPOOL parameter altogether, RAPOOL=50 is assumed. CICS maintains n VTAM RECEIVE ANYs, where n is either the RAPOOL "number active" value, or the MXT value minus the number of active tasks, whichever is the smaller. For example:

If RAPOOL=2, MXT=50, active tasks = 45 then RECEIVE ANY = 2
 If RAPOOL=10, MXT=50, active tasks = 45 then RECEIVE ANY = 5
 If RAPOOL=10, MXT=50, active tasks = 35 then RECEIVE ANY = 10

The number of RECEIVE ANYs needed depends on the expected activity of the system, the average transaction lifetime, and the MAXTASK value specified. For information about coding this parameter, see the [CICS Performance Guide](#).

RDSASZE={0K|number}

Code this parameter with the size of the RDSA. The default size is 0 indicating that the DSA size can be changed dynamically. A non-zero value indicates that the DSA size is fixed.

number

Specify number as an amount of storage in the range 0 to 16,777,215 bytes in multiples of 262,144 bytes (256KB). If the size specified is not a multiple of 256KB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4,194,304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

Restrictions: You can code the RDSASZE parameter in PARM, SYSIPT or CONSOLE only.

RENTPGM={PROTECT|NOPROTECT}

Code this parameter to specify whether you want CICS to allocate the read-only DSAs, RDSA and ERDSA, from read-only key-0 protected storage. The permitted values are PROTECT (the default), or NOPROTECT:

PROTECT

CICS obtains the storage for the read-only DSAs from key-0 protected storage.

NOPROTECT

CICS obtains the storage from CICS-key storage, effectively creating two more CICS DSAs (CDSA and ECDSA). This allows programs eligible for the read-only DSAs to be modified by programs that execute in CICS key.

You are recommended to specify RENTPGM=NOPROTECT for development regions only, and to specify RENTPGM=PROTECT for production CICS regions.

RESP={FME|RRN}

Code this parameter to specify the type of request that CICS terminal control receives from logical units.

FME

Function management end is the default.

RRN

Reached recovery node.

RESSEC={ASIS|ALWAYS}

Specifies whether or not you want CICS to honor the RESSEC option specified on a transaction's resource definition.

ASIS

means that CICS honors the RESSEC option defined in a transaction's resource definition. CICS calls its resource security checking routine only when RESSEC(YES) is specified in a transaction resource definition. This is normally a sufficient level of control, because often you will need only to control the ability to execute a transaction.

ALWAYS

means that CICS overrides the RESSEC option, and always calls its resource security checking routine to issue the appropriate call to the SAF interface.

Use this option only if you need to control or audit all accesses to CICS resources. You should be aware that using this option can significantly degrade performance.

Restrictions: You can code the RESSEC parameter in the SIT, PARM, or SYSIPT only.

RMTRAN={CSGM|name1},{CSGM|name2}}

Code this parameter with the name of the transaction you want to initiate when a terminal session is recovered at system restart for a CICS region running with VTAM persistent sessions support.

If you do not specify a name on the RMTRAN parameter, CICS uses the value specified on the GMTRAN system initialization parameter; the CSGM transaction is the default CICS good morning transaction.

name1

This is the transaction that CICS initiates at terminals that do **not** remain signed-on after system restart (that is, they are still connected to CICS, but are signed off).

name2

This is the transaction that CICS initiates at terminals that remain signed-on after system restart. If you specify only name1, CICS uses the CSGM transaction as the default for name2.

RUWAPool={NO|YES}

specifies the option for allocating a storage pool the first time an LE-conforming program runs in a task.

NO

CICS disables the option and provides no RUWA storage pool. Every EXEC CICS LINK to an LE-conforming application results in a GETMAIN for RUWA storage.

YES

CICS creates a pool of storage the first time an LE-conforming program runs in a task. This provides an available storage pool that reduces the need to GETMAIN and FREEMAIN run-unit work areas (RUWAs) for every EXEC CICS LINK request.

SDSASZ={OK|number}

Code this parameter with the size of the SDSA. The default size is 0 indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

number

Specify number as an amount of storage in the range 0 to 16,777,215 bytes in multiples of 262,144 bytes (256KB). If the size specified is not a multiple of 256KB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4,194,304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

Restriction: You can code the SDSASZ parameter in PARM, SYSIPT or CONSOLE only.

SEC={YES|NO}

Code this parameter to indicate what level of external security you want CICS to use.

YES

Code YES if you want to use full external security. CICS requires the appropriate level of authorization for the access intent: a minimum of READ permission for read intent, and a minimum of UPDATE permission for update intent.

Note: You must also ensure that the default userid (CICSUSER or another userid specified on the DFLTUSER system initialization parameter) has been defined to your external security manager (ESM).

If command security checking is defined for EXEC CICS SP-type commands, then specifying SEC=YES means that the appropriate level of authority is checked for; therefore:

- A check for READ authority is made for EXEC CICS INQUIRE and COLLECT commands
- A check for UPDATE authority is made for EXEC CICS SET, PERFORM, and DISCARD commands

For the results of the interaction between the access intent of the user application, and the permission defined to your ESM, see [Table 35 on page 351](#).

- A check for ALTER authority is made for EXEC CICS CREATE commands.

NO

Code NO if you do not want CICS to use an ESM. All users have access to all resources, whether determined by attempts to use them or by the QUERY SECURITY command. Users are not allowed to sign on or off.

Note: With MRO bind-time security, even if you specify SEC=NO, the CICS region userid is still sent to the secondary CICS region, and bind-time checking is still carried out in the secondary CICS region. For information about MRO bind-time security, see the [CICS Security Guide](#).

Define whether to use your ESM for resource level checking by using the XDCT, XFCT, XJCT, XPCT, XPPT, XPSB, and XTST system initialization parameters. Define whether to use an ESM for transaction-attach security checking by using the XTRAN system initialization parameter. Define whether ESM session security can be used when establishing APPC sessions by using the XAPPC system initialization parameter.

For information on defining command security checking for CICS SP-type commands, and about CICS security in general, see the [CICS Security Guide](#).

For programming information about the use of external security for CICS system commands, see the [CICS System Programming Reference](#) publication.

Access Permission defined to ESM for CICS user	Access intent in application		
	READ	UPDATE	ALTER
NONE	Refused	Refused	Refused
READ	Permitted	Refused	Refused
UPDATE	Permitted	Permitted	Refused
ALTER	Permitted	Permitted	Permitted

Restrictions: You can code the SEC parameter in the SIT, PARM, or SYSIPT only.

SECPRFX={NO|YES}

Specifies whether or not CICS prefixes the resource names in any authorization requests to the ESM with a prefix corresponding to the ESM userid for the CICS region. The prefix to be used (the userid for the CICS region) is obtained by the DFHIRP module.

NO

CICS does not prefix the resource names in any authorization requests to the ESM.

YES

The ESM userid is used as the prefix for CICS resources defined to the ESM. CICS prefixes the resource name in any authorization requests to the ESM with a prefix corresponding to the ESM userid of the CICS region, obtaining the userid as follows:

- If you start CICS as a job, the prefix corresponds to the USER operand coded on the //ID JOB statement of the CICS startup job stream.
- If you start a CICS job without an associated ESM userid, the prefix defaults to CICS.

For information about using the PREFIX option, see the *CICS Security Guide*.

Restrictions:

- You can code the SECPRFX parameter in the SIT, PARM, or SYSIPT only.
- The SECPRFX parameter is effective only if you specify YES for the SEC system initialization parameter.

SIT=xx

Code this parameter to specify the suffix, if any, of the system initialization table that you want CICS to load at the start of initialization. If you omit this parameter, CICS loads the unsuffixed table, DFHSIT,

which is pregenerated with all the default values. This default SIT named DFHSIT\$\$, and its source, is in the z/VSE sublibrary, PRD1.BASE.

Restrictions: You can code the system initialization parameter anywhere in PARM or SYSIPT, or as the *first* parameter entry at the CONSOLE.

SKRxxxx='page-retrieval-command'

Code this parameter if a single-keystroke-retrieval operation is required. 'xxxx' specifies a key on the 3270 keyboard which, during a page retrieval session, is to be used to represent a page retrieval command. The valid keys are PA1 through PA3, and PF1 through PF36. Thus up to 39 keys can be specified in this way (each by a separate command).

The 'page-retrieval-command' value represents any valid page retrieval command, and must be enclosed in apostrophes. It is concatenated to the character string coded in the PGRET parameter. The combined length must not exceed 16 characters.

Note: If full function BMS is used, all PA keys and PF keys are interpreted for page retrieval commands, even if some of these keys are not defined.

SNSCOPE={NONE|CICS|VSEIMAGE|}

Specifies whether or not a userid can be signed on to CICS more than once, within the scope of:

- A single CICS region
- A single VSE image

NONE

Each userid can be used to sign on for any number of sessions on any CICS region. This is the compatibility option, providing the same signon scope as in releases of CICS before CICS Transaction Server for z/VSE Version 2 Release 2.

CICS

Each userid can be signed on once only in the same CICS region. A signon request is rejected if the userid is already signed on to the same CICS region. However, the userid can be used to signon to another CICS region in the same, or another, VSE image.

VSEIMAGE

Each userid can be signed on once only, and to only one of the set of CICS regions in the same VSE image that also specify SNSCOPE=VSEIMAGE. A signon request is rejected if the user is already signed on to another CICS region in the same VSE image.

The signon scope (if specified) applies to all userids signing on by an explicit signon request (for example, by an EXEC CICS SIGNON command or the CESN transaction). SNSCOPE is restricted to users signing on at local terminals, or signing on after using the CRTE transaction to connect to another system.

Signon scope specified by SNSCOPE *does not* apply to:

- Non-terminal users.
- The CICS default userid, specified by the DFLTUSER system initialization parameter.
- Preset userids, specified in the USERID option of the RDO TERMINAL resource definition.
- Userids for remote users, received in attach headers.
- Userids for link security. For information about which userid is used for link security on a specific connection, see the [CICS Security Guide](#).
- The userid specified on the PLTPIUSR system initialization parameter.
- The CICS region userid.

Restrictions: You can code the SNSCOPE parameter in the SIT, PARM, or SYSIPT only.

SPCTR={{(1,2|1[,2][,3)]|ALL|OFF}}

Code this parameter to set the level of tracing for all CICS components used by a transaction, terminal, or both, selected for special tracing. If you want to set different tracing levels for an individual component of CICS, use the SPCTRxx system initialization parameter. You can select up to three levels of tracing, but some CICS components do not have trace points at all these levels. For

a list of all the available trace points and their level numbers, see the [CICS User's Handbook](#). For information about the differences between special and standard CICS tracing, see the [CICS Problem Determination Guide](#).

number

Code the level numbers for the level of special tracing you want for all CICS components. The options are: 1, (1,2), or (1,2,3). The default, (1,2), specifies special tracing for levels 1 and 2 for all CICS components.

ALL

Enables the special tracing facility for all available levels.

OFF

Disables the special tracing facility.

SPCTRxx={{(1,2|1[,2][,3)]|ALL|OFF}}

Code this parameter to set the level of tracing for a particular CICS component used by a transaction, terminal, or both, selected for special tracing. You identify the component by coding a value for xx in the keyword. You code one SPCTRxx keyword for each component you want to define selectively. For a CICS component being specially traced that does not have its trace level set by SPCTRxx, the trace level is that set by SPCTR (which, in turn, defaults to (1,2)). You can select up to three levels of tracing, but some CICS components do not have trace points at all these levels. The CICS component codes that you can code for xx on the SPCTRxx keyword are shown in [Table 36](#) on [page 353](#):

<i>Table 36. CICS component names and abbreviations</i>			
Code	Component name	Code	Component name
AP	Application domain	BF	Built-in function
BM	Basic mapping support	BR	3270 Bridge
CP	Common programming interface	DC	Dump compatibility layer
DD	Directory manager domain	DH	Document handler domain
DI	Batch data interchange	DM	Domain manager domain
DS	Dispatcher domain	DU	Dump domain
EI	Exec interface	FC	File control
GC	Global catalog domain	IC	Interval control
IS	Inter-system communication	JC	Journal control
KC	Task control	KE	Kernel
LC	Local catalog domain	LD	Loader domain
LM	Lock domain	ME	Message domain
MN	Monitoring domain	PA	Parameter domain
PC	Program control	PG	Program manager domain
RC	Report Controller	SC	Storage control
SM	Storage domain	SO	Sockets domain
SP	Sync point	ST	Statistics domain
SZ	Front end programming interface	TC	Terminal control
TD	Transient data	TI	Timer domain
TR	Trace domain	TS	Temporary storage
UE	User exit interface	US	User domain

<i>Table 36. CICS component names and abbreviations (continued)</i>			
Code	Component name	Code	Component name
WB	Web support	XM	Transaction manager domain
XS	Security manager domain		

Note: The component codes BF, BM, BR, CP, DC, DI, EI, FC, IC, IS, JC, KC, PC, RC, SC, SP, SZ, TC, TD, TS, and UE are sub-components of the AP domain. As such, the corresponding trace entries will be produced with a point ID of AP nnnn.

For details of using trace, see the [CICS Trace Entries](#).

number

Code the level numbers for the level of special tracing you want for the CICS component indicated by xx. The options are: 1, (1,2), or (1,2,3).

ALL

Code ALL to indicate that you want all the available levels of special CICS tracing switched on for the specified component.

OFF

Code OFF to switch off all levels of special CICS tracing for the CICS component indicated by xx.

Restrictions: You can code the SPCTRxx parameter in PARM, SYSIPT, or CONSOLE only.

SPOOL=({NO|YES}[,]{A|identifier}[,]{A|class}{YES|NO})

Specifies whether the system spooling interface is required. The identifier and class values are for the report controller.

Use the final YES|NO option to specify whether you want a formfeed (blank page) to precede the first report printed on a non-SCS printer started to the report controller.

The specified identifier is a single character used to identify the printer-owning CICS system to the cross-partition communication (XPCC) component in the device-driving system name (ddsname).

The specified class is the default output class to be used for reports that have no class specified.

NO

Specifies that the system spooling interface is not required.

YES

Specifies that the system spooling interface is required.

A

Specifies that the name SYSCICSA is used to identify this CICS system.

identifier

Specifies a single character identifier, which is appended to SYSCICSA to create a name used to identify this CICS system. This identifier should be unique for each CICS system running with the system spooling interface. If you do not provide a unique identifier, you might receive message DFHRC5301I (indicating that the POWER® interface has failed to initialize).

A

Class A is the default for the output class for reports and printers.

class

Specifies a single alpha-numeric character identifier (A to Z, 0-9 only), used as the default output class for reports and printers.

YES

Specifies a formfeed (blank page) is to precede the first report printed on a non-SCS printer started to the report controller.

NO

Specifies a formfeed (blank page) is not to precede the first report printed on a non-SCS printer started to the report controller. If you code NO, be aware that a report can overwrite the last line of the previous output when the printer is started to the report controller.

SRT={YES|NO|xx}

Specifies the system recovery table suffix. For more information, refer to [“Notes on CICS resource table and module keywords”](#) on page 307.

For information about coding the macros for this table, see the [CICS System Definition Guide](#).

If SRT=NO is coded, the system recovery program (DFHSRP) does not attempt to recover from a program check or from an operating system abend. However, CICS issues VSE ESTAEX macros to intercept program checks to perform clean-up operations before CICS terminates. Therefore, an SRT must be provided if recovery from either program checks or abnormal terminations, or both, is required.

SRVERCP={037|CCSID}

Specifies the default server code page to be used by the DFHCNV data conversion table but only if the SRVERCP parameter in the DFHCNV macro is set to SYSDEF.

SSLDELAY=600|number

Specifies the length of time in seconds for which CICS retains session IDs for secure socket connections. Session IDs are tokens that represent a secure connection between a client and an SSL server. While the session ID is retained by CICS within the SSLDELAY period, CICS can continue to communicate with the client without the significant overhead of an SSL handshake. The value is a number of seconds in the range 0 through 86400. The default value is 600.

START=({AUTO|COLD|STANDBY|LOGTERM}|,ALL)

Specifies the type of start for the system initialization program. The value specified for START, or the default of AUTO, becomes the default value for each resource.

AUTO

CICS performs either a warm or an emergency restart, according to the status of the control record written to the global catalog by the previous execution of CICS. If the global catalog is newly initialized, it does not contain a control record, and CICS forces a cold start for START=AUTO.

If you code START=AUTO, you must provide a restart data set for use in case CICS has to perform an emergency restart, and the system log from the previous execution of CICS must be available. You must also have coded an activity keypoint value (see [“AKPFREQ={200|number}”](#) on page 317) on the previous execution of CICS for an emergency restart to be successful.

COLD

Specifies a cold start. The status of CICS resource definitions saved in the global catalog at the previous shutdown is ignored, and all resource definitions are reinstalled, either from the CSD or CICS control tables. However, some information saved in the global catalog and local catalog is preserved across cold starts. For information about how CICS uses information saved in the global catalog and local catalog, see the [CICS Recovery and Restart Guide](#).

LOGTERM

LOGTERM is a special option for use as an alternative to a full emergency restart when the previous run terminated in an uncontrolled shutdown, and is available only if the CICS system log is defined on disk data sets. If you specify START=LOGTERM, CICS initializes up to the point where it writes an end of file on the system log. After it has written the end of file, CICS ends the emergency restart process and shuts down without doing any backout processing.

For background information about this restart process, see the [CICS Recovery and Restart Guide](#).

Restrictions: START=LOGTERM is applicable only if XRF=NO is coded. You can code START=LOGTERM in PARM, SYSIPT, or CONSOLE only.

STANDBY

Coding START=STANDBY, but only when you have also specified XRF=YES, defines this CICS as the alternate CICS region in an XRF pair. In other words, you *must* specify START=STANDBY for

the system that starts off as the alternate. (To start an active CICS region, specify AUTO or COLD, as you would without XRF.)

If you have specified a COLD start for other CICS resources, for example, DCT=(xx,COLD), they are cold started when the alternate CICS region (with START=STANDBY specified) takes over. This may cause CICS to lose data on an XRF takeover; for example, coding ICP=COLD results in outstanding STARTs being lost. You are recommended to code START=(STANDBY,ALL) to ensure a full emergency restart during takeover, unless you wish to specifically cold start individual resources.

(option,ALL)

Specifies that the ALL operand is a special option you can use on the START parameter when you supply it as a system initialization parameter at CICS startup; you cannot code it in the SIT. If you specify START=(AUTO,ALL), CICS initializes all resources according to the type of startup that it selects (warm, emergency, or cold). The ALL option overrides any individual settings in other system initialization parameters (for example, DCT=(xx,COLD)).

However, if you do not use the ALL option, you can individually cold start those resources that have a COLD operand. For details of resources that have a COLD option, see [Table 28 on page 307](#).

Restrictions: You can code START=(option,ALL) in PARM, SYSIPT, or CONSOLE only.

For more information about the types of CICS startup, see [“Classes of start and restart” on page 310](#).

STARTER={NO|YES}

Code YES to indicate that the generation of starter system modules (with \$ and # suffixes) is permitted, and various MNOTES are to be suppressed. This parameter should only be used when service is being performed on starter system modules.

Restrictions: You can code the STARTER parameter in the SIT only.

STATRCD=OFF|ON

Specifies the interval statistics recording status at CICS initialization. This status is recorded in the CICS global catalog for use during warm and emergency restarts. Statistics collected are written to the DMF data set.

OFF

Interval statistics are not collected (no action is taken at the end of an interval).

End-of-day, Unsolicited and requested statistics are written to DMF regardless of the STATRCD setting.

ON

Interval statistics are collected.

On a cold start of a CICS region, interval statistics are recorded by default at three-hourly intervals. All intervals are timed using the end-of-day time (midnight is the default) as a base starting time (*not* CICS startup time). This means that the default settings give collections at 00.00, 03.00, 06.00, 09.00, and so on, regardless of the time that you start CICS.

On a warm or emergency restart the statistics recording status is restored from the CICS global catalog.

You can change the statistics recording status at any time as follows:

- During a warm or emergency restart by coding the STATRCD system initialization parameter.
- While CICS is running by using the CEMT or EXEC CICS SET STATISTICS command.

Whatever the value of the STATRCD system initialization parameter, you can ask for requested statistics and requested reset statistics to be collected. You can get statistics "on demand" for all, or for specified, resource types by using the CEMT or EXEC CICS PERFORM STATISTICS command. The period covered for statistics requested in this way is from the last reset time (that is, from the beginning of the current interval or from when you last issued a CEMT or EXEC CICS statistics command specifying RESETNOW) up to the time that you issue the PERFORM STATISTICS command.

For information about using these CEMT commands, see [CICS Supplied Transactions](#) publication. For programming information about the EXEC CICS PERFORM commands, see the [CICS System Programming Reference](#) publication.

For information about the statistics utility program DFHSTUP, see [CICS Operations and Utilities Guide](#).

STGPROT={NO|YES}

Specifies whether you want storage protection in the CICS region. The permitted values are NO (the default), or YES:

NO

If you specify NO, or allow this parameter to default, CICS does not operate any storage protection, and runs in a single storage key as in earlier releases. See the [CICS System Definition Guide](#) for a summary of how STGPROT=NO affects the storage allocation for the dynamic storage areas.

YES

If you specify YES, and if you have the required hardware and software, CICS operates with storage protection, and observes the storage keys and execution keys that you specify in various system and resource definitions. See the [CICS System Definition Guide](#) for a summary of how STGPROT=YES affects the storage allocation for the dynamic storage areas.

If you do not have the required hardware and software support, CICS issues an information message during initialization, and operates without storage protection.

STGRVCY={NO|YES}

Specifies whether CICS should try to recover from a storage violation.

NO

CICS does not try to repair any storage violation that it detects.

YES

CICS tries to repair any storage violation that it detects.

In both cases, CICS continues unless you have specified in the dump table that CICS should terminate.

In normal operation, CICS sets up four task-lifetime storage subpools for each task. Each element in the subpool starts and ends with a 'check zone' that includes the subpool name. At each freemain, and at end-of-task, CICS checks the check zones and abends the task if either has been overwritten.

Terminal input-output areas (TIOAs) have similar check zones, which are set up with identical values. At each freemain of a TIOA, CICS checks the check zones and abends the task if they are not identical.

If you specify the STGRVCY(YES) system initialization parameter, CICS resets the check zones correctly and the task continues running.

STNTR={1|(1,2|,3)|ALL|OFF}

Specifies the level of standard tracing required for CICS as a whole. You can select up to three levels of tracing, but some CICS components do not have trace points at all these levels.

Note: Before globally activating tracing levels 3 and ALL for the storage manager (SM) component, read the warning given in the description for the STNTRxx system initialization parameter.

number

Code the level number(s) for the level of standard tracing you want for all CICS components. The options are: 1, (1,2), or (1,2,3). The default, 1, specifies standard tracing for level 1 for all CICS components.

ALL

Enables standard tracing for all levels.

OFF

Disables standard tracing.

For information about the differences between special and standard CICS tracing, see the [CICS Problem Determination Guide](#).

STNTRxx={ (1,2|1,2|,3) | ALL | OFF }

Specifies the level of standard tracing you require for a particular CICS component. You identify the component by coding a value for xx in the keyword. You code one STNTRxx keyword for each component you want to define selectively. For a CICS component being specially traced that does not have its trace level set by STNTRxx, the trace level is that set by STNTR (which, in turn, defaults to 1). You can select up to three levels of tracing, but some CICS components do not have trace points at all these levels.

The CICS component codes that you can code for xx on this STNTRxx keyword are shown in [Table 36 on page 353](#).

number

Code the level number(s) for the level of standard tracing you want for the CICS component indicated by xx. The options are: 1, (1,2), or (1,2,3).

ALL

Code ALL to indicate that you want all the available levels of standard tracing switched on for the specified component.

OFF

Code OFF to switch off all levels of standard CICS tracing for the CICS component indicated by xx.

Attention! If you select tracing levels 3 or ALL for the storage manager (SM) component, the performance of your CICS system will be degraded. This is because options 3 and ALL switch on levels of trace that are also used for field engineering purposes. See the [CICS Problem Determination Guide](#) for information about the effects of trace levels 3 and ALL.

Restrictions: You can code the STNTRxx parameter in PARM, SYSIPT, or CONSOLE only.

SUFFIX=xx

Specifies the last two characters of the name of this system initialization table.

The first 6 characters of the name of the SIT are fixed as DFHSIT. You can specify the last two characters of the name, using the SUFFIX parameter. Because the SIT does not have a TYPE=INITIAL macro statement like other CICS resource control tables, you specify its SUFFIX on the TYPE=CSECT macro statement.

The suffix allows you to have more than one version of the SIT. Any one or two characters (other than NO and DY) are valid. You select the version of the table to be loaded into the system during system initialization by coding SIT=xx, either in the PARM parameter or the SYSIPT data set. (You can, in some circumstances, specify the SIT using the system console, but this is not recommended.)

Restrictions: You can code the SUFFIX parameter in the SIT only.

SVA={NO|YES}

Specifies whether any CICS management modules can be used from the VSE shared virtual area (SVA).

NO

No CICS management modules are used from the SVA.

YES

CICS management modules installed in the SVA can be used from there, instead of being loaded into the CICS system.

A list of the CICS modules that are read-only, and hence eligible for residence in the SVA, is given in Appendix B of the [CICS System Definition Guide](#).

For details of the CICS system initialization parameter PRVMOD that you can use to override SVA=YES for selected modules, see page [“PRVMOD={name|\(name,name...name\)}” on page 347](#).

SYDUMAX={999|number}

Specifies the limit on the number of system dumps that may be taken per dump table entry. If this number is exceeded, subsequent system dumps for that particular entry will be suppressed.

number

Specifies a number in the range 0 through 999. The default, 999, enables an unlimited number of dumps to be taken.

SYSIDNT={CICS|name}

Specifies a 1- to 4-character name that is known only to your CICS region. If your CICS region also communicates with other CICS regions, the name you choose for this parameter to identify your local CICS region must not be the same name as an installed RDO CONNECTION resource definition for a remote region.

The value for SYSIDNT, whether specified in the SIT or as an override, can only be updated on a cold start. After a warm start or emergency restart, the value of SYSIDNT is that specified in the last cold start.

For information about the SYSIDNT of a local CICS region, see the [CICS Intercommunication Guide](#).

SYSTR={ON|OFF}

Specifies the master system trace flag.

Code ON to obtain trace entries of CICS system activity. Entries are written to all the trace destinations that are active.

TAKEOVR={AUTO|MANUAL|COMMAND}

Code the TAKEOVR parameter in the SIT for an alternate XRF system. It specifies the action to be taken by the alternate CICS, following the (apparent) loss of the surveillance signal in the active CICS system. In doing this, it also specifies the level of operator involvement.

AUTO

No operator approval or intervention is needed for a takeover.

MANUAL

Specifies that the operator is asked to approve a take over if the alternate system cannot detect the surveillance signal of the active system.

Note that the alternate system does not ask the operator for approval if the active system signs off abnormally, nor is there an operator or program command for take over. In these cases, there is no doubt that the alternate system should take over, and manual involvement by the operator would be an unnecessary overhead in the take over process.

For example, you could code MANUAL to ensure manual take over of a master or coordinator region in MRO.

COMMAND

Take over only occurs when a CEBT PERFORM TAKEOVER command is received by the alternate system. It ensures, for example, that a dependent alternate CICS (in MRO) is activated only if it receives the command from the operator, or from a master (coordinator) region.

TBEXITS=(*name1*],[*name2*],[*name3*],[*name4*],[*name5*])

Code the names of your transaction backout exit programs. These exits are used for resource backout during emergency restart processing. For programming information about transaction backout exit programs, see the [CICS Customization Guide](#). For background information about transaction backout, see the [CICS Recovery and Restart Guide](#).

The order in which you code the names is critical. If you do not want to use all five exits, code commas in place of the ones you miss out. For example:

```
TBEXITS=( , , EXITF , EXITV )
```

name1

The name of your initialization/termination exit program.

name2

The name of your input exit program.

name3

The name of your file error exit program.

name4

The name of your open error exit program.

name5

The name of your DL/I error exit program.

If no transaction backout exit programs are required, you can do one of the following:

- Omit the whole parameter from the SIT
- Code TBEXITS=(,,,,) as a SIT parameter
- Code TBEXITS=(,,,,) as a SIT override.

TCP={YES|NO}

Code TCP=YES to include the pregenerated non-VTAM terminal control program, DFHTCP.

You must code TCP=YES if you intend using card reader/line printer (sequential) devices.

TCPIP={NO|YES}

Specifies whether CICS TCPIP services are to be activated at CICS startup. The default is NO, meaning that these services cannot be enabled. If TCPIP is set to YES, the HTTP service can process work. SSL support can only be used when TCPIP=YES.

Note: The TCPIP system initialization parameter affects only CICS internal TCP/IP Services defined by TCPIPSERVICE resource definitions. It has nothing to do with the TCP/IP Socket Interface for CICS feature of TCP/IP for z/VSE.

TCSACTN={NONE|UNBIND|FORCE}

Specifies the required action that CICS terminal control should take if the terminal control shutdown wait threshold expires. For details of the wait threshold, see the TCSWAIT system initialization parameter. TCSACTN only takes effect when TCSWAIT is coded with a value in the range 1 through 99. This parameter only applies to VTAM terminals (including LU Type 6.2 single-session APPC terminals), not VTAM intersystem connections (LU Type 6.1 and LU Type 6.2 parallel connections). This is a global default action. On a terminal-by-terminal basis, you can code a DFHZNEP routine to override this action.

NONE

No action is taken. This can be overridden by DFHZNEP.

UNBIND

CICS terminal control attempts to force-close the session by issuing a VTAM CLSDST and sending an SNA UNBIND command to the hung terminal. This can be overridden by DFHZNEP.

FORCE

CICS terminal control attempts to forceclose the CICS VTAM ACB if there are any hung terminals. All CICS VTAM terminals and sessions are released and CICS normal shutdown continues.

TCSWAIT={4|number|NO|NONE|0}

Specifies the required CICS terminal control shutdown wait threshold. The wait threshold is the time, during shutdown, that CICS terminal control allows to pass before it considers terminal shutdown to be hung. If all VTAM sessions shutdown and close before the threshold expires then the CICS shutdown process moves on to its next stage, and the terminal control wait threshold then no longer applies. If, however, some of the VTAM session do not complete shutdown and close, then CICS takes special action with these sessions. For details of this special action see the description of the TCSACTN system initialization parameter. The wait threshold only applies to VTAM sessions; that is, VTAM terminals and VTAM intersystem connections. The wait time is specified as a number of minutes, in the range 1 through 99. As a special case, TCSWAIT=NO may be specified to indicate that terminal control shutdown is never to be considered hung, no matter how long the shutdown and close process takes. TCSWAIT=NONE and TCSWAIT=0 are alternative synonyms for TCSWAIT=NO, and all three have the same effect (internally they are held as the one value 0 (zero)).

TCT={YES|xx|NO}

Specifies which terminal control table, if any, is to be loaded. For more information, refer to [“Notes on CICS resource table and module keywords”](#) on page 307.

For guidance about coding the macros for this table, see the [CICS Resource Definition Guide](#)

If you reassemble the TCT after starting CICS, any changes are applied when you next start CICS, even if it is a warm or emergency startup.

If you have VTAM-connected terminals only, you can code TCT=NO. If you do this, note that a dummy TCT, called DFHTCTDY, is loaded during system initialization. For more information about DFHTCTDY, see [The dummy TCT, DFHTCTDY](#). (If you code TCT=NO, you must specify a CSD group list in the GRPLIST parameter.)

TCTUAKEY={USER|CICS}

Specifies the storage key for the terminal control table user areas (TCTUAs) if you are operating CICS with storage protection (STGPROT=YES). The permitted values are USER (the default), or CICS:

USER

If you specify USER, or allow this parameter to default, CICS obtains the amount of storage for TCTUAs in user key. This allows a user program executing in any key to modify the TCTUA.

CICS

If you specify CICS, CICS obtains the amount of storage in CICS key. This means that only programs executing in CICS key can modify the TCTUA, and user-key programs have read-only access.

If CICS is running without storage protection, the TCTUAKEY parameter only designates which DSA (User or CICS) the storage comes from. The TCTUAs are accessed in CICS-key whether they are in the UDSA or CDSA.

See [CICS System Definition Guide](#) for more information about TCTUAs.

TCTUALOC={BELOW|ANY}

Specifies where terminal user areas (TCTUA) are to be stored.

BELOW

Specifies that the TCTUAs are stored below the 16MB line.

ANY

Specifies that the TCTUAs are stored anywhere in virtual storage. CICS stores TCTUAs above the 16MB line if possible.

For more information about TCTUAs, see the [CICS System Definition Guide](#).

For details about defining terminals using RDO, see the [CICS Resource Definition Guide](#).

TD=({3|decimal-value-1},{3|decimal-value-2})

Specifies the number of VSAM buffers and strings to be used for intrapartition transient data (TD).

decimal-value-1

Specifies that the number of buffers to be allocated for the use of intrapartition transient data. The value must be in the range 1 through 32 767. The default value is 3.

CICS obtains, above the 16MB line, storage for the TD buffers in units of the page size (4KB). Because CICS optimizes the use of the storage obtained, TD may allocate more buffers than you specify, depending on the control interval (CI) size you have defined for the intrapartition data set.

For example, if the CI size is 1536, and you specify 3 buffers (the default number), CICS actually allocates 5 buffers. This is because 2 pages (8192 bytes) are required to obtain sufficient storage for three 1536-byte buffers, a total of only 4608 bytes, which would leave 3584 bytes of spare storage in the second page. In this case, CICS allocates another 2 buffers (3072 bytes) to minimize the amount of unused storage. In this way CICS makes use of storage that would otherwise be unavailable for any other purpose.

decimal-value-2

Specifies that the number of VSAM strings to be allocated for the use of intrapartition transient data. The value must be in the range 1 through 255, and must not exceed the value specified in decimal-value-1. The default value is 3.

For example, TD=(8,5) specifies 8 buffers and 5 strings.

The operands of the TD parameter are positional. You must code commas to indicate missing operands if others follow. For example, TD=(,2) specifies the number of strings and allows the number of buffers to default.

TRAP={OFF|ON}

Specifies whether the FE global trap exit is to be activated at system initialization. This exit is for diagnostic use under the guidance of service personnel. For background information about this exit, see the [CICS Problem Determination Guide](#).

TRDUMAX={999|number}

Specifies the limit on the number of transaction dumps that may be taken per dump table entry. If this number is exceeded, subsequent transaction dumps for that particular entry will be suppressed.

number

Specifies a number in the range 0 through 999. The default, 999, enables an unlimited number of dumps to be taken.

TRTABSZ={16|number-of-kilobytes}

Specifies the size in kilobytes of the internal trace table. (1KB = 1024 bytes.) The CICS trace table is allocated in virtual storage above the 16MB line, and it is allocated *before* the extended CICS-key DSA (ECDSA) and the extended user-key DSA (EUDSA). Ensure that there is sufficient virtual storage for the trace table, the ECDSA, and the EUDSA by defining a large enough VSE partition for your CICS job.

16

16KB is the default size of the trace table, and also the minimum size.

number

The number of kilobytes of storage to be allocated for the internal trace table, in the range 16KB through 1048576KB. Subpool 1 is used for the trace table storage, which exists for the duration of the CICS execution. The table is page aligned and occupies a whole number of pages. If the value specified is not a multiple of the page size (4KB), it is rounded up to the next multiple of 4KB.

Trace entries are of variable lengths, but the average length is approximately 100 bytes.

Note: To switch on internal tracing, use the INTTR parameter; for a description of INTTR, see “[INTTR={ON|OFF}](#)” on page 338.

TRTRANSZ={40|number-of-kilobytes}

specifies the size in kilobytes of the transaction dump trace table. (1KB = 1024 bytes.)

When a transaction dump is taken, CICS performs a VSE GETMAIN for storage above the 16MB line for the transaction dump trace table.

40

40KB is the default size of the transaction dump trace table. The minimum size is 16KB.

number

The number of kilobytes of storage to be allocated for the transaction dump trace table, in the range 16KB through 1048576KB.

TRTRANTY={TRAN|ALL}

specifies which trace entries should be copied from the internal trace table to the transaction dump trace table.

TRAN

Only the trace entries associated with the transaction that is abending will be copied to the transaction dump trace table.

ALL

All of the trace entries from the internal trace table will be copied to the transaction dump trace table. If the internal trace table size is larger than the transaction dump trace table size, the transaction dump trace table could wrap. This results in only the most recent trace entries being written to the transaction dump trace table.

TS=([COLD][,{0|3|decimal-value-1}][,{3|decimal-value-2}])

Specifies:

- Whether or not you want to cold start temporary storage
- The number of VSAM buffers to be used for auxiliary temporary storage
- The number of VSAM strings to be used for auxiliary temporary storage.

COLD

Specifies that the type of start for the temporary storage program. If you do not want a cold start, code a comma before the second operand.

Note: IF ICP is warm-started (that is no ICP=COLD) and TS is cold-started, any ICEs and AIDs with data attached are lost.

0

No buffers are required; that is, only MAIN temporary storage is required.

decimal-value-1

Specifies that the number of buffers to be allocated for the use of auxiliary temporary storage. The value must be in the range 3 through 32 767.

CICS obtains, above the 16MB line, storage for the auxiliary temporary storage buffers in units of the page size (4KB). Because CICS optimizes the use of the storage obtained, TS may allocate more buffers than you specify, depending on the control interval (CI) size you have defined for the auxiliary temporary storage data set.

For example, if the CI size is 2048, and you specify 3 buffers (the default number), CICS actually allocates 4 buffers. This is because 2 pages (8192 bytes) are required to obtain sufficient storage for three 2048-byte buffers, a total of 6144 bytes, which would leave 2048 bytes of spare storage in the second page. In this case, CICS allocates another 2048-byte buffer to use all of the 8192 bytes obtained. In this way CICS makes use of storage that would otherwise be unavailable for any other purpose.

decimal-value-2

Specifies that the number of VSAM strings to be allocated for the use of auxiliary temporary storage. The value must be in the range 1 through 255, and must not exceed the value specified in decimal-value-1. The default value is 3.

For example, TS=(COLD,8,5) specifies 8 buffers and 5 strings.

The operands of the TS parameter are positional. You must code commas to indicate missing operands if others follow. For example, TS=(,8) specifies the number of buffers and allows the other operands to default.

TSMGSET={4|number}

Specifies that the number of entries for which dynamic storage is allocated for storing pointers to records put to a temporary storage message set. When the entries are used, space is acquired for the same number of entries as many times as required to accommodate the total number of records in the queue. The range is 4 through 100.

TST={NO|YES|xx}

Specifies the temporary storage table suffix. For more information, refer to [“Notes on CICS resource table and module keywords”](#) on page 307.

For information about coding the macros for this table, see the [CICS Resource Definition Guide](#)

UDSASZE={0K|number}

Specifies the size of the UDSA. The default size is 0 indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

number

Specify number as an amount of storage in the range 0 to 16,777,215 bytes in multiples of 262,144 bytes (256KB). If the size specified is not a multiple of 256KB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4,194,304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

Restrictions: You can code the UDSASZE parameter in PARM, SYSIPT and CONSOLE only.

USERTR={ON|OFF}

To set the master user trace flag on or off. If the user trace flag is off, the user trace facility is disabled, and EXEC CICS ENTER TRACENUM commands receive an INVREQ condition if EXCEPTION is not specified. If the program does not handle this condition the transaction will abend with an abend code of AEIP.

For programming information about the user trace facility using EXEC CICS ENTER TRACENUM commands, see the [CICS Application Programming Reference](#).

USRDELAY={30|number}

Specify the maximum time, in the range 0 through 10080 minutes (up to 7 days), that an eligible userid and its associated attributes are to be retained in the user table if the userid is unused. An entry in the user table for a userid that is retained during the delay period can be reused.

The userids eligible for reuse within the USRDELAY period are any that are:

- Received from remote systems
- Specified on SECURITYNAME in RDO CONNECTION definitions
- Specified on USERID in RDO SESSIONS definitions
- Specified on USERID on DFHDCT TYPE=INTRA definitions
- Specified to be used for non-terminal STARTed transactions.

Within the USRDELAY period, a userid in any one of these categories can be reused in one of the other categories, provided the request for reuse is qualified with the same qualifiers. If a userid is qualified by different group id, APPLID, or terminal id, a retained entry is not reused.

If a userid is unused for more than the USRDELAY limit, it is removed from the system, and the message DFHUS0200 is issued. You can suppress this message in an XMEOUT global user exit program. If you specify USRDELAY=0, all eligible userids are deleted immediately after use, and the message DFHUS0200 is not issued. Do not code USRDELAY=0 if this CICS region communicates with other CICS regions and:

- ATTACHSEC=IDENTIFY is specified on the CONNECTION definitions for the connections used
- The connections used carry high volumes of transaction routing or function shipping activity.

You should specify a value that gives the optimum level of performance for your CICS environment.

Note: If a value, other than 0, is specified for USRDELAY, the ability to change the user's attributes or revoke the userid becomes more difficult because the userid and its attributes are retained in the region until the USRDELAY value has expired. For example, if you have specified USRDELAY=30 for a userid, but that userid continues to run transactions every 25 minutes, the USRDELAY value will never expire and any changes made to the userid will never come into effect.

When running a remote transaction, a userid remains signed-on to the remote CICS region (after the conversation associated with the first attach request is complete) until the delay specified by USRDELAY has elapsed since the last transaction associated with the attach request for the userid has completed. When this event occurs, the userid is removed from the remote CICS region.

VTAM={YES|NO}

Code VTAM=NO only if you do not use the VTAM access method. The default is VTAM=YES.

VTPREFIX={\|character}

Specifies that the first character to be used for the terminal identifiers (termids) of autoinstalled virtual terminals. Virtual terminals are used by the External Presentation Interface (EPI) and terminal emulator functions of the CICS Client products.

Termids generated by CICS for autoinstalled Client terminals consist of a 1-character prefix and a 3-character suffix. The default prefix is '\'. The suffix can have the values 'AAA' through '999'. That is, each character in the suffix can have the value 'A' through 'Z' or 'O' through '9'. The first suffix generated by CICS has the value 'AAA'. This is followed by 'AAB', 'AAC', ... 'AAZ', 'AA0', 'AA1', and so on, up to '999'.

Each time a Client virtual terminal is autoinstalled, CICS generates a 3-character suffix that it has not recorded as being in use.

By specifying a prefix, you can ensure that the termids of Client terminals autoinstalled on this system are unique in your transaction routing network. This prevents the conflicts that could occur if two or more terminal-owning regions (TORs) ship definitions of Client virtual terminals to the same application-owning region (AOR).

If such a naming conflict does occur—that is, if a Client virtual terminal is shipped to an AOR on which a remote terminal of the same name is already installed—the autoinstall user program is invoked in the AOR. Your user program can resolve the conflict by allocating an alias terminal identifier to the shipped definition. (For details of writing an autoinstall user program to install shipped definitions, see the [CICS Customization Guide](#)). However, you can avoid potential naming conflicts by specifying a different prefix, reserved for virtual terminals, on each TOR on which Client virtual terminals are to be installed.

You must not use the characters + - * < > = { } or blank.

Note:

1. The autoinstall user program is not called at install of Client terminals, so cannot be used to specify termids.
2. When specifying a prefix, ensure that termids generated by CICS for Client terminals do not conflict with those generated by your autoinstall user program for user terminals, or with the names of any other terminals or connections.
3. Client terminal definitions are not recovered after a restart. Immediately after a restart, no Client terminals are in use, so when CICS generates suffixes it begins again with 'AAA'. This means that CICS does **not** always generate the same termid for any given Client terminal. This in turn means that server applications should not assume that a particular CICS-generated termid always equates to a particular Client terminal.
4. Clients can override CICS-generated termids.

For further information about Client virtual terminals, see the [CICS Family: Communicating from CICS on System/390](#) publication.

WEBDELAY={5|time_out,60|keep_time}

Specifies two Web delay periods:

1. A time-out period. The maximum time, in minutes, in the range 1-60, that a transaction started through the Web 3270 bridge interface, is allowed to remain in terminal wait state before it is automatically purged by CICS.
2. The terminal keep time. The time, in minutes, in the range 1-6000, during which state data is kept for a CICS Web 3270 bridge transaction, before CICS performs clean-up.

WRKAREA={512|number}

Specifies the number of bytes to be allocated to the common work area (CWA). This area, for use by your installation, is initially set to binary zeros, and is available to all programs. It is not used by CICS. The maximum size for the CWA is 3584 bytes.

XAPPC={NO|YES}

Specifies whether ESM security can be used when establishing APPC sessions.

NO

ESM session security cannot be used. Only the BINDPASSWORD (defined to CICS for an APPC connection) is checked.

YES

ESM session security can be used.

If you specify BINDSECURITY=YES for a particular APPC connection, a request to the ESM is issued to extract the security profile. If the profile exists, it is used to bind the session. If it does not exist, only the BINDPASSWORD (defined to CICS for the connection) is checked.

Note: If you specify XAPPC=YES, the external security manager that you use must support the APPCLU general resource class, otherwise CICS fails to initialize.

Restrictions: You can code the XAPPC parameter in the SIT, PARM, or SYSIPT only.

XCMD={NO|name|YES}

specifies whether you want CICS to perform command security checking, and, optionally, the ESM resource class name in which you have defined the command security profiles. If you specify YES, or an ESM resource class name, CICS calls the ESM to verify that the userid associated with a transaction is authorized to use a CICS command for the specified resource. Such checking is performed every time a transaction tries to use a COLLECT, CREATE, DISABLE, DISCARD, ENABLE, EXTRACT, INQUIRE, PERFORM, RESYNC, or SET command, or any of the FEPI commands, for a resource.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the CMDSEC(YES) option on the RDO TRANSACTION resource definition.

For information about preparing for and using security with CICS, see the [CICS Security Guide](#).

NO

CICS does not perform any command security checks, allowing any user to use commands that would be subject to those checks.

name

CICS calls the ESM using the specified resource class name, prefixed by C or V, to verify that the userid associated with a transaction is authorized to use a CICS command for the specified resource. The resource class name is *Cname* and the grouping class name is *Vname*.

The resource class name specified must be 1 through 7 characters.

YES

CICS calls the ESM using the default class name of CICSCMD, prefixed by C or V, to check whether the userid associated with a transaction is authorized to use a CICS command for the specified resource. The resource class name is CCICSCMD and the grouping class name is VCICSCMD.

Restrictions: You can code the XCMD parameter in the SIT, PARM, or SYSIPT only.

XDCT={NO|name|YES}

Specifies whether you want CICS to perform transient data resource security checking. If you specify YES or an ESM resource class name, CICS calls the ESM to verify that the userid associated with a transaction is authorized to access the transient data destination. Such checking is performed every time a transaction tries to access a transient data destination.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the RDO TRANSACTION resource definition.

For information about preparing for and using security with CICS, see the [CICS Security Guide](#).

NO

CICS does perform any transient data security checks, allowing any user to access any transient data destination.

name

CICS calls the ESM using the specified resource class name to check whether the userid associated with the transaction is authorized to access the specified destination. The resource class name is *Dname* and the grouping class name is *Ename*.

The resource class name specified must be 1 through 7 characters.

YES

CICS calls the ESM with the default CICS resource class name of CICSXDCT, prefixed by D or E, to verify whether the userid associated with the transaction is authorized to access the specified destination.

The resource class name is DCICXDCT and the grouping class name is ECICXDCT.

Restrictions: You can code the XDCT parameter in the SIT, PARM, or SYSIPT only.

XFACT={NO|name|YES}

Specifies whether you want CICS to perform file resource security checking, and optionally specifies the ESM resource class name in which you have defined the file resource security profiles. If you specify YES, or an ESM resource class name, CICS calls the ESM to verify that the userid associated with a transaction is authorized to access file control-managed files. Such checking is performed every time a transaction tries to access a file managed by CICS file control.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the RDO TRANSACTION resource definitions.

For information about preparing for and using security with CICS, see the [CICS Security Guide](#).

NO

CICS does not perform any file resource security checks, allowing any user to access any file.

name

CICS calls the ESM, using the specified resource class name, to verify that the userid associated with a transaction is authorized to access files referenced by the transaction. The resource class name is *Fname* and the grouping class name is *Hname*.

The resource class name specified must be 1 through 7 characters.

YES

CICS calls the ESM, using the default CICS resource class name of CICSFCT prefixed by F or H, to verify that the userid associated with a transaction is authorized to access files reference by the transaction. The resource class name is FCICSFCT and the grouping class name is HCICSFCT.

Restrictions: You can code the XFACT parameter in the SIT, PARM, or SYSIPT only.

XJCT={NO|name|YES}

Specifies whether you want CICS to perform journal resource security checking. If you specify YES, or an ESM resource class name, CICS calls the ESM to verify that the userid associated with a transaction is authorized to access the referenced journal. Such checking is performed every time a transaction tries to access a CICS journal.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the RDO TRANSACTION resource definitions.

For information about preparing for and using security with CICS, see the [CICS Security Guide](#).

NO

CICS does not perform any journal resource security checks, allowing any user to access any CICS journal.

name

CICS calls the ESM, using the specified resource class name prefixed by J or K, to verify that the userid associated with a transaction is authorized to access CICS journals.

The resource class name specified must be 1 through 7 characters.

YES

CICS calls the ESM, using the default CICS resource class name of CICSJCT prefixed by a J or K, to check whether the userid associated with a transaction is authorized to access CICS journals referenced by the transaction. The resource class name is JCICSJCT and the grouping class name is KCICSJCT.

Restrictions: You can code the XJCT parameter in the SIT, PARM, or SYSIPT only.

XLT={NO|xx|YES}

Specifies a suffix for the transaction list table. For more information, refer to [“Notes on CICS resource table and module keywords” on page 307](#).

The XLT contains a list of transactions that can be attached during the first quiesce stage of system termination.

NO

Specifies that a transaction list table is not used.

xx

Specifies that the transaction list table DFHXLtx is used.

YES

Specifies that the default transaction list table, DFHXL, is used.

For guidance information about coding the macros for this table, see the [CICS Resource Definition Guide](#)

XPCT={NO|name|YES}

Specifies whether you want CICS to perform started transaction resource security checking, and optionally specifies the name of the ESM resource class name in which you have defined the started task security profiles. If you specify YES, or an ESM resource class name, CICS calls the ESM to verify that the userid associated with a transaction is authorized to use started transactions and related EXEC CICS commands. Such checking is performed every time a transaction tries to use a started transaction or one of the EXEC CICS commands: COLLECT STATISTICS TRANSACTION, DISCARD TRANSACTION, INQUIRE TRANSACTION, or SET TRANSACTION.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the RDO TRANSACTION resource definitions.

For information about preparing for and using security with CICS, see the [CICS Security Guide](#).

NO

CICS does not perform any started task resource security checks, allowing any user to use started transactions or related EXEC CICS commands.

name

CICS calls the ESM using the specified resource class name, to verify that the userid associated with a transaction is authorized to use started transactions or related EXEC CICS commands. The resource class name is ACICSPCT and the grouping class name is BCICSPCT.

The resource class name specified must be 1 through 7 characters.

YES

CICS calls the ESM, using the default CICS resource class name CICSPCT prefixed with A or B, to verify that the userid associated with a transaction is authorized to use started transactions or related EXEC CICS commands.

The resource class name is ACICSPCT and the grouping class name is BCICSPCT.

Restriction: You can code the XPCT parameter in the SIT, PARM, or SYSIPT only.

XPPT={NO|name|YES}

Specifies that CICS is to perform application program resource security checks, and optionally specifies the ESM resource class name in which you have defined the program resource security profiles. Such checking is performed every time a transaction tries to invoke another program by using one of the CICS commands: LINK, LOAD, or XCTL.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the RDO TRANSACTION resource definitions.

For information about preparing for and using security with CICS, see the [CICS Security Guide](#).

NO

CICS does not perform any application program authority checks, allowing any user to use LINK, LOAD, or XCTL commands to invoke other programs.

name

CICS calls the ESM, with the specified resource class name prefixed by M or N, to verify that the userid associated with a transaction is authorized to use LINK, LOAD, or XCTL commands to invoke other programs. The resource class name is Mname and the grouping class name is Nname.

The resource class name specified must be 1 through 7 characters.

YES

CICS calls the ESM, using the default resource class name prefixed by M or N, to verify that the userid associated with a transaction is authorized to use LINK, LOAD, or XCTL commands to invoke other programs. The resource class name is MCICSPPT and the grouping class name is NCICSPPT.

Restriction: You can code the XPPT parameter in the SIT, PARM, or SYSIPT only.

XPSB={NO|name|YES}

Specifies whether you want CICS to perform program specification block (PSB) security checking, and optionally specifies the ESM resource class name in which you have defined the PSB security profiles. If you specify YES, or an ESM resource class name, CICS calls the ESM to check that the userid associated with a transaction is authorized to access PSBs (which describe databases and logical message destinations used by application programs). Such checking is performed every time a transaction tries to access a PSB.

The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the RDO TRANSACTION resource definitions.

For information about preparing for and using security with CICS, see the [CICS Security Guide](#).

NO

CICS does not perform any PSB resource security checks, allowing any user to access any PSB.

name

CICS calls the ESM using the specified resource class name prefixed by P or Q, to verify that the userid associated with a transaction is authorized to access PSBs. The resource class name is *Pname* and the grouping class name is *Qname*.

The resource class name specified must be 1 through 7 characters.

YES

CICS calls the ESM, using the default resource class name CICSPSB prefixed by P or Q, to verify that the userid associated with a transaction is authorized to access PSBs. The resource class name is PCICSPSB and the grouping class name is QCICSPSB.

Restriction: You can code the XPSB parameter in the SIT, PARM, or SYSIPT only.

XRF={NO|YES}

You must specify YES if you want XRF support to be included in the CICS region. If the CICS region is started with the START=STANDBY system initialization parameter specified, the CICS region is the **alternate CICS region**. If the CICS region is started with the START=AUTO or START=COLD system initialization parameter specified, the CICS region is the **active CICS region**. The active CICS region signs on as such to the CICS availability manager. For background information about XRF, see the [CICS XRF Guide](#).

Note: You must code JCT=xx|YES, if you are using XRF.

XRFSOFF={NOFORCE|FORCE}

Specifies whether all users signed-on to the active CICS region are to remain signed-on following a takeover. This parameter is only applicable if you also code XRF=YES as a system initialization parameter.

NOFORCE

Code NOFORCE to allow CICS to determine sign-off according to the option set in either of:

- The CICS segment of the ESM database
- The RDO TYPETERM resource definition for the user's terminal.

Note: For a terminal to remain signed-on after an XRF takeover, NOFORCE must be specified in the SIT, ESM database, and the terminal's RDO TYPETERM resource definition.

FORCE

Code FORCE if you want **all** terminal users to be signed off in the event of a takeover by an alternate CICS region, regardless of individual options set in the ESM database or in the terminals' RDO TYPETERM resource definition.

For information about the XRFSSOFF option of the RDO TYPETERM resource definition, see the [CICS Resource Definition Guide](#).

XRFSTME={5|decimal-value}

Specifies, in minutes, a time-out delay interval for users who are still signed-on when an XRF takeover occurs.

If you have specified NOFORCE in the ESM database, and in the terminals' RDO TYPETERM resource definition, and the takeover takes longer than the time specified in the XRFSTME, all users who are still signed-on after takeover are signed off.

5

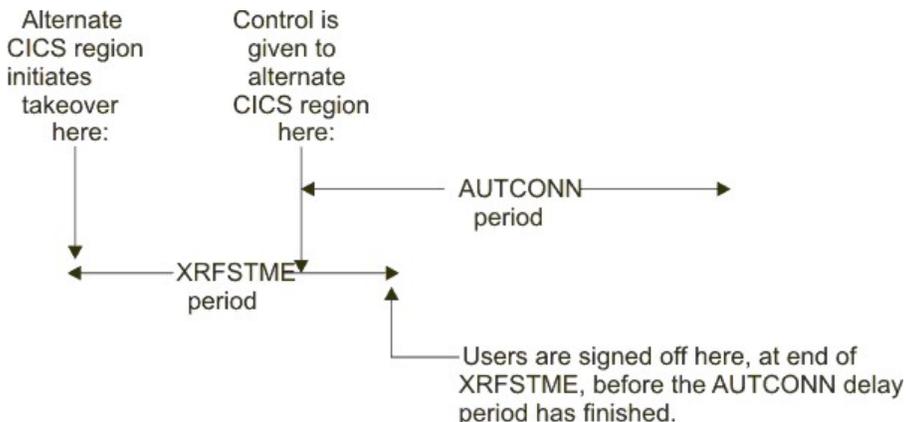
The default value is five minutes.

decimal-value

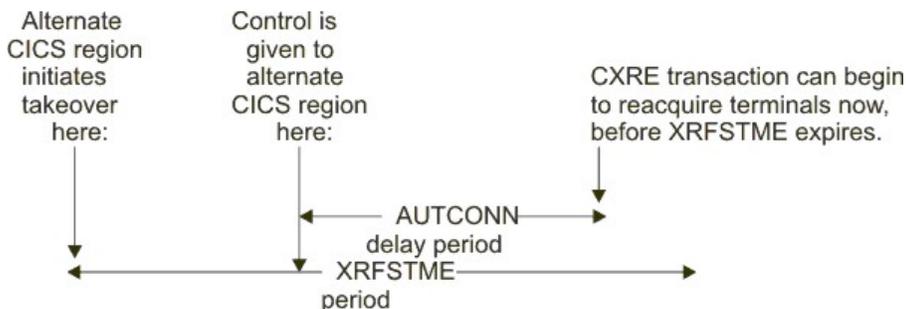
Code a value in the range 0 through 60 for the number of minutes CICS permits users to remain signed on during the takeover period. The takeover period is the time from when the takeover is initiated to the time at which CICS is ready to process user transactions. If the takeover takes longer than the specified period, all users signed-on at the time the takeover was initiated are signed-off.

A value of 0 specifies that there is no time-out delay, and terminals are signed off as soon as takeover commences, which means that XRFSTME=0 has the same effect as coding XRFSSOFF=FORCE.

For non-XRF-capable terminals, take into account any AUTCONN delay period when setting the value for XRFSTME. (See the description of the AUTCONN parameter "APPLID=(generic_applid,specific_applid)" on page 318.) You may need to increase the XRFSTME value to allow for the delay to the start of the CXRE transaction imposed by the AUTCONN parameter; otherwise, terminals may be signed-off too early. For example:



You can avoid this situation by extending the XRFSTME period to exceed the AUTCONN period. For example:



XRFTODI={30|number}

Use the XRFTODI parameter, in the SIT for an alternate XRF system, to specify, in seconds, the takeover delay interval. The minimum time value is 5 seconds. The alternate system has to ensure that the active system has been canceled before it can take over the resources owned by the active.

Because the cancelation process is not fully automatic (for example, the primary CPC fails), the XRFTODI value specifies the interval before the operator becomes involved.

XSWITCH=({@|1-254|NO}][, {???????|programe}][, {A|B})

Specifies that the XSWITCH parameter defines a programmable terminal switching unit, which may be used with mid-range 2-CPC XRF systems instead of a communication controller. The program defined on the XSWITCH parameter instructs the unit to switch terminal lines to the active system's CPC at start up, and to the alternate system's CPC at take over.

@|1-254

Specifies that the logical unit to which the switch is assigned. 0 is the default.

???????|programe

Specifies that the user-written program that issues commands to the switching unit. The program is device-dependent, but has a standard interface that follows normal subroutine conventions.

A|B

Specifies that the CEC to which the terminal lines are to be directed. A is the active CPC and B is the alternate CPC,

XTRAN={YES|name|NO}

Specifies whether you want CICS to perform transaction-attach security checking, and optionally specifies the ESM resource class name in which you have defined the transaction security profiles. If you specify YES, or an ESM resource class name, CICS calls the ESM to verify that the userid associated with the transaction is permitted to run the transaction.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the RDO TRANSACTION resource definitions.

YES

CICS calls the ESM, using the default CICS resource class name of CICSTRN prefixed by T or G, to verify that the userid associated with the transaction is authorized to run the transaction. The resource class name is TCICSTRN and the grouping class name is GCICSTRN.

name

CICS calls the ESM, using the specified resource class name prefixed by T or G, to verify that the userid associated with the transaction is authorized to run the transaction. The resource class name is *Tname* and the corresponding grouping class name is *Gname*.

The name specified must be 1 through 7 characters.

NO

CICS does not perform any transaction-attach security checks, allowing any user to run any transaction.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter.

Restrictions: You can code the XTRAN parameter in the SIT, PARM, or SYSIPT only.

XTST={NO|name|YES}

Specifies whether you want CICS to perform temporary storage security checking, and optionally specifies the ESM resource class name in which you have defined the temporary storage security profiles. If you specify YES or an ESM resource class name, CICS calls the ESM to verify that the userid associated with a temporary storage request is authorized to access the referenced temporary storage queue.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter, specified the RESSEC option on the RDO TRANSACTION resource definitions, and specified a DFHTYPE=SECURITY macro for the queue in the temporary storage table (TST).

NO

CICS does not perform any temporary storage security checks, allowing any user to access any temporary storage queue.

SIT Parameters

name

CICS calls the ESM, using the specified resource class name prefixed by S or U, to verify that the userid associated with a transaction is authorized to access temporary storage queues.

The name specified must be 1 through 7 characters.

YES

CICS calls the ESM, using the default CICS resource class name of CICSTST prefixed by S or U, to verify that the userid associated with the transaction is authorized to access temporary storage queues referenced by the transaction. The resource class name is SCICSTST and the corresponding grouping class name is UCICSTST.

Restrictions: You can code the XTST parameter in the SIT, PARM, or SYSIPT only.

XUSER={NO|YES}

Specifies whether or not CICS is to perform surrogate user checks.

NO

Specifies that CICS is not to perform any surrogate user checking.

YES

specifies that CICS is to perform surrogate user checking in all those situations that permit such checks to be made (for example, on EXEC CICS START commands without an associated terminal). For information about the various circumstances in which CICS performs surrogate user checks, see the [CICS Security Guide](#).

Restrictions: You can code the XUSER parameter in the SIT, PARM, or SYSIPT only.

Part 12. Trace entries

Chapter 51. Trace entries

IRC Commands AP trace points

Table 37. IRC Commands

Point ID	Module	Lvl	Type	Data
AP DD3C	DFHCRR	Exc	DFHTC WRITE LAST failure	1 Address of session TCTTE

Dispatcher domain trace points

Table 38. Dispatcher domain trace points

Point ID	Module	Lvl	Type	Data
DS 0022	DFHSDS3	DS 1	Before VSE wait	1 Internal data 2 Internal data
DS 0023	DFHSDS3	DS 1	After VSE wait	1 Internal data 2 Internal data

File control trace points

Table 39. File control trace points

Point ID	Module	Lvl	Type	Data
AP 0492	DFHFCVR	FC 2	After VSAM call	1 RPL 2 Address of RPL 3 RIDFLD
AP 0493	DFHFCVR	FC 2	After VSAM call	1 RPL 2 Address of RPL 3 RIDFLD

Document handler AP trace points

Table 40. Document handler

Point ID	Module	Lvl	Type	Data
AP F920	DFHEIDH	AP 2	Entry	1 EIEI parameter list
AP F921	DFHEIDH	AP 2	Exit	1 EIEI parameter list
AP F922	DFHEIDH	Exc	Invalid format	1 EIEI parameter list
AP F923	DFHEIDH	Exc	Invalid function	1 EIEI parameter list
AP F924	DFHEIDH	Exc	Invalid API function	1 EIEI parameter list

Socket domain

Table 41. Socket domain trace points

Point ID	Module	Lvl	Type	Data
SO 0101	DFHSODM	SO 1	Entry	1 DMDM parameter list
SO 0102	DFHSODM	SO 1	Exit	1 DMDM parameter list
SO 0103	DFHSODM	Exc	Recovery	1 DMDM parameter list 2 Kernel error data
SO 0104	DFHSODM	Exc	Invalid format	1 DMDM parameter list
SO 0105	DFHSODM	Exc	Invalid function	1 DMDM parameter list
SO 0106	DFHSODM	Exc	Exclusive lock error	1 DMDM parameter list 2 LMLM parameter list
SO 0107	DFHSODM	Exc	Exclusive unlock error	1 DMDM parameter list 2 LMLM parameter list
SO 0108	DFHSODM	Exc	Storage error	1 DMDM parameter list

Table 41. Socket domain trace points (continued)

Point ID	Module	Lvl	Type	Data
SO 0109	DFHSODM	Exc	Shared lock error	1 DMDM parameter list
				2 LMLM parameter list
SO 0110	DFHSODM	Exc	Shared unlock error	1 DMDM parameter list
				2 LMLM parameter list
SO 0111	DFHSODM	Exc	Recovery release lock error	1 DMDM parameter list
				2 LMLM parameter list
SO 0112	DFHSODM	Exc	Dispatcher error	1 DMDM parameter list
SO 0113	DFHSODM	Exc	Directory manager error	1 DMDM parameter list
SO 0201	DFH SOCK	SO 1	Entry	1 SOCK parameter list
				2 Send : sent data
SO 0201	DFH SOCK	SO 1	Entry	1 SOCK parameter list
SO 0201	DFH SOCK	SO 2	Entry	1 SOCK parameter list
				2 Send : sent data
SO 0202	DFH SOCK	SO 1	Exit	1 SOCK parameter list
SO 0202	DFH SOCK	SO 2	Exit	1 SOCK parameter list
				2 Receive : received data
SO 0203	DFH SOCK	Exc	Recovery	1 SOCK parameter list
				2 Kernel error data
SO 0204	DFH SOCK	Exc	Invalid format	1 SOCK parameter list
SO 0205	DFH SOCK	Exc	Invalid function	1 SOCK parameter list

Trace Entries

Table 41. Socket domain trace points (continued)

Point ID	Module	Lvl	Type	Data
SO 0206	DFH SOCK	Exc	Exclusive lock error	1 SOCK parameter list 2 LMLM parameter list
SO 0207	DFH SOCK	Exc	Exclusive unlock error	1 SOCK parameter list 2 LMLM parameter list
SO 0208	DFH SOCK	Exc	Getmain error	1 SMGF parameter list
SO 0209	DFH SOCK	SO 2	Before asyncio	1 bpx_interface 2 LTE 3 STE 4 AioCb
SO 0210	DFH SOCK	SO 2	After asyncio	1 bpx_interface 2 LTE 3 STE 4 AioCb
SO 0211	DFH SOCK	Exc	Asyncio error	1 bpx_interface 2 LTE 3 STE 4 AioCb

Table 41. Socket domain trace points (continued)

Point ID	Module	Lvl	Type	Data
SO 0212	DFH SOCK	SO 2	Before select	1 bpx_interface 2 LTE 3 Read_List 4 Write_List 5 Exception_List 6 ECB
SO 0213	DFH SOCK	SO 2	After select	1 bpx_interface 2 LTE 3 Read_List 4 Write_List 5 Exception_List 6 ECB
SO 0214	DFH SOCK	Exc	Select error	1 bpx_interface 2 LTE 3 Read_List 4 Write_List 5 Exception_List 6 ECB
SO 0215	DFH SOCK	SO 2	Before socket	1 bpx_interface 2 LTE
SO 0216	DFH SOCK	SO 2	After socket	1 bpx_interface 2 LTE

Trace Entries

Table 41. Socket domain trace points (continued)

Point ID	Module	Lvl	Type	Data
SO 0217	DFH SOCK	Exc	Socket error	1 bpx_interface 2 LTE
SO 0218	DFH SOCK	SO 2	Before bind	1 bpx_interface 2 LTE 3 Sockaddr
SO 0219	DFH SOCK	SO 2	After bind	1 bpx_interface 2 LTE 3 Sockaddr
SO 0220	DFH SOCK	Exc	Bind error	1 bpx_interface 2 LTE 3 Sockaddr
SO 0221	DFH SOCK	SO 2	Before listen	1 bpx_interface 2 LTE
SO 0222	DFH SOCK	SO 2	After listen	1 bpx_interface 2 LTE
SO 0223	DFH SOCK	Exc	Listen error	1 bpx_interface 2 LTE
SO 0224	DFH SOCK	SO 2	Before accept	1 bpx_interface 2 LTE 3 STE 4 Sockaddr

Table 41. Socket domain trace points (continued)

Point ID	Module	Lvl	Type	Data
SO 0225	DFH SOCK	SO 2	After accept	1 bpx_interface 2 LTE 3 STE 4 Sockaddr
SO 0226	DFH SOCK	Exc	Accept error	1 bpx_interface 2 LTE 3 STE 4 Sockaddr
SO 0227	DFH SOCK	SO 2	Before getclientid	1 bpx_interface 2 LTE 3 Clientid
SO 0228	DFH SOCK	SO 2	After getclientid	1 bpx_interface 2 LTE 3 Clientid
SO 0229	DFH SOCK	Exc	Getclientid error	1 bpx_interface 2 LTE 3 Clientid
SO 0231	DFH SOCK	SO 2	Before takesocket	1 bpx_interface 2 LTE 3 STE 4 Clientid

Trace Entries

Table 41. Socket domain trace points (continued)

Point ID	Module	Lvl	Type	Data
SO 0232	DFH SOCK	SO 2	After takesocket	1 bpx_interface 2 LTE 3 STE 4 Clientid
SO 0233	DFH SOCK	Exc	Takesocket error	1 bpx_interface 2 LTE 3 STE 4 Clientid
SO 0234	DFH SOCK	SO 2	Before givesocket	1 bpx_interface 2 LTE 3 STE 4 Clientid
SO 0235	DFH SOCK	SO 2	After givesocket	1 bpx_interface 2 LTE 3 STE 4 Clientid
SO 0236	DFH SOCK	Exc	Givesocket error	1 bpx_interface 2 LTE 3 STE 4 Clientid

Table 41. Socket domain trace points (continued)

Point ID	Module	Lvl	Type	Data
SO 0237	DFH SOCK	SO 2	Before close	1 bpx_interface 2 LTE 3 STE
SO 0238	DFH SOCK	SO 2	After close	1 bpx_interface 2 LTE 3 STE
SO 0239	DFH SOCK	Exc	Close error	1 bpx_interface 2 LTE 3 STE
SO 0240	DFH SOCK	SO 2	Before setsockopt	1 bpx_interface 2 LTE 3 option_data
SO 0241	DFH SOCK	SO 2	After setsockopt	1 bpx_interface 2 LTE 3 option_data
SO 0242	DFH SOCK	Exc	Setsockopt error	1 bpx_interface 2 LTE 3 option_data
SO 0243	DFH SOCK	Exc	Socket in use	1 SOCK parameter list
SO 0244	DFH SOCK	Exc	Attach error	1 STE 2 LTE

Trace Entries

Table 41. Socket domain trace points (continued)

Point ID	Module	Lvl	Type	Data
SO 0245	DFH SOCK	SO 2	IO wait posted for termination	1 STE
SO 0246	DFH SOCK	Exc	Unknown session token	1 SOCK parameter list
SO 0247	DFH SOCK	Exc	IO error	1 SOCK parameter list
SO 0248	DFH SOCK	Exc	DSSR wait error	1 SOCK parameter list
SO 0249	DFH SOCK	SO 2	Immclose requested	
SO 0250	DFH SOCK	SO 2	Register requested	
SO 0251	DFH SOCK	SO 2	Deregister requested	
SO 0252	DFH SOCK	SO 2	Quiesce requested	
SO 0253	DFH SOCK	SO 2	Terminate requested	
SO 0254	DFH SOCK	Exc	Unknown post code	1 bpx_interface 2 LTE 3 ECB
SO 0255	DFH SOCK	SO 2	Before gethostname	1 bpx_interface 2 Name
SO 0256	DFH SOCK	SO 2	After gethostname	1 bpx_interface 2 Name
SO 0257	DFH SOCK	Exc	Gethostname error	1 bpx_interface 2 Name
SO 0258	DFH SOCK	SO 2	TCP/IP inactive	
SO 0260	DFH SOCK	SO 2	Listen subroutine entry	1 SOCK parameter list
SO 0261	DFH SOCK	SO 2	Listen subroutine exit	1 SOCK parameter list

Table 41. Socket domain trace points (continued)

Point ID	Module	Lvl	Type	Data
SO 0262	DFH SOCK	Exc	EIO received	1 bpx_interface 2 LTE 3 Read_List 4 Write_List 5 Exception List 6 ECB
SO 0263	DFH SOCK	Exc	Accept EIO received	1 bpx_interface 2 LTE 3 STE 4 Sockaddr
SO 0270	DFH SOCK	Exc	Shared lock error	1 SOCK parameter list 2 LMLM parameter list
SO 0271	DFH SOCK	Exc	Shared unlock error	1 SOCK parameter list 2 LMLM parameter list
SO 0272	DFH SOCK	Exc	Recovery release lock error	1 SOCK parameter list 2 LMLM parameter list
SO 0273	DFH SOCK	Exc	Unknown KE error code	1 SOCK parameter list
SO 0274	DFH SOCK	Exc	Asyncio function error	1 bpx_interface 2 LTE 3 STE 4 AioCb
SO 0275	DFH SOCK	Exc	Getmain for STE failure	

Trace Entries

Table 41. Socket domain trace points (continued)

Point ID	Module	Lvl	Type	Data
SO 0280	DFH SOCK	SO 2	Before iwmsrsg	1 iwm_parameters
SO 0281	DFH SOCK	SO 2	After iwmsrsg	1 iwm_parameters
SO 0282	DFH SOCK	Exc	Iwmsrsg error	1 iwm_parameters
SO 0283	DFH SOCK	SO 2	Before iwmsrdrs	1 iwm_parameters
SO 0284	DFH SOCK	SO 2	After iwmsrdrs	1 iwm_parameters
SO 0285	DFH SOCK	Exc	Iwmsrdrs error	1 iwm_parameters
SO 0290	DFH SOCK	SO 1	Initialize timer entry	
SO 0291	DFH SOCK	SO 1	Initialize timer exit	
SO 0292	DFH SOCK	SO 1	Timer event	
SO 0293	DFH SOCK	SO 1	Monitor data put entry	
SO 0294	DFH SOCK	SO 1	Monitor data put exit	
SO 0295	DFH SOCK	SO 1	Cancel timer entry	
SO 0296	DFH SOCK	SO 1	Cancel timer exit	
SO 0297	DFH SOCK	SO 2	Connection count increment	1 Tcpiptime name 2 New connection count
SO 0298	DFH SOCK	SO 2	Connection count decrement	1 Tcpiptime name 2 New connection count
SO 0299	DFH SOCK	SO 2	Asynco wakeup	1 LTE 2 STE 3 AioRv 4 AioRc 5 AioRsn

Table 41. Socket domain trace points (continued)

Point ID	Module	Lvl	Type	Data
SO 029A	DFH SOCK	SO 2	Before sigprocmask	1 bpx_interface 2 How 3 New_signal_mask 4 Old_signal_mask
SO 029B	DFH SOCK	SO 2	After sigprocmask	1 bpx_interface 2 How 3 New_signal_mask 4 Old_signal_mask
SO 029C	DFH SOCK	Exc	Sigprocmask error	1 bpx_interface 2 How 3 New_signal_mask 4 Old_signal_mask
SO 0301	DFH SORD	SO 1	Entry	1 SORD parameter list
SO 0302	DFH SORD	SO 1	Exit	1 SORD parameter list
SO 0303	DFH SORD	Exc	Recovery	1 SORD parameter list 2 Kernel error data
SO 0304	DFH SORD	Exc	Invalid format	1 SORD parameter list
SO 0305	DFH SORD	Exc	Invalid function	1 SORD parameter list
SO 0306	DFH SORD	Exc	Lock error	1 SORD parameter list 2 LMLM parameter list

Trace Entries

Table 41. Socket domain trace points (continued)

Point ID	Module	Lvl	Type	Data
SO 0307	DFHSORD	Exc	Unlock error	1 SORD parameter list
				2 LMLM parameter list
SO 0308	DFHSORD	Exc	Storage error	1 SMGF parameter list
SO 0401	DFHSOIS	SO 1	Entry	1 SOIS parameter list
SO 0402	DFHSOIS	SO 1	Exit	1 SOIS parameter list
SO 0403	DFHSOIS	Exc	Recovery	1 SOIS parameter list
				2 Kernel error data
SO 0404	DFHSOIS	Exc	Invalid format	1 SOIS parameter list
SO 0405	DFHSOIS	Exc	Invalid function	1 SOIS parameter list
SO 0406	DFHSOIS	Exc	Exclusive lock error	1 SOIS parameter list
				2 LMLM parameter list
SO 0407	DFHSOIS	Exc	Exclusive unlock error	1 SOIS parameter list
				2 LMLM parameter list
SO 0408	DFHSOIS	Exc	Shared lock error	1 SOIS parameter list
				2 LMLM parameter list
SO 0409	DFHSOIS	Exc	Shared unlock error	1 SOIS parameter list
				2 LMLM parameter list
SO 0410	DFHSOIS	Exc	Recovery unlock error	1 SOIS parameter list
				2 LMLM parameter list
SO 0411	DFHSOIS	Exc	Storage error	1 SMGF parameter list
SO 0412	DFHSOIS	Exc	Unknown KE error code	

Table 41. Socket domain trace points (continued)

Point ID	Module	Lvl	Type	Data
SO 0413	DFHSOIS	Exc	CEEPIPI error	1 c_interface
SO 0414	DFHSOIS	Exc	Gethostbyaddr error	1 c_interface
SO 0415	DFHSOIS	Exc	Gethostid error	1 c_interface
SO 0416	DFHSOIS	SO 2	Before CEEPIPI	1 c_interface 2 pipi_table 3 init_sub_dp : runopts
SO 0417	DFHSOIS	SO 2	After CEEPIPI	1 c_interface 2 pipi_table 3 init_sub_dp : runopts
SO 0418	DFHSOIS	Exc	Dispatcher error	1 SOIS parameter list
SO 0419	DFHSOIS	Exc	Dub error	1 dub_results
SO 0420	DFHSOIS	SO 2	Before sigprocmask	1 bpx_interface 2 How 3 New_signal_mask 3 Old_signal_mask
SO 0421	DFHSOIS	SO 2	After sigprocmask	1 bpx_interface 2 How 3 New_signal_mask 3 Old_signal_mask

Trace Entries

Table 41. Socket domain trace points (continued)

Point ID	Module	Lvl	Type	Data
SO 0422	DFHSOIS	Exc	Sigprocmask error	1 bpx_interface 2 How 3 New_signal_mask 3 Old_signal_mask
SO 0501	DFHSOUE	SO 1	Entry	1 SOUE parameter list
SO 0502	DFHSOUE	SO 1	Exit	1 SOUE parameter list
SO 0503	DFHSOUE	Exc	Recovery	1 SOUE parameter list 2 Kernel error data
SO 0504	DFHSOUE	Exc	Invalid function	1 SOUE parameter list
SO 0505	DFHSOUE	Exc	Invalid format	1 SOUE parameter list
SO 0506	DFHSOUE	Exc	Lock error	1 SOUE parameter list 2 LMLM parameter list
SO 0507	DFHSOUE	Exc	Unlock error recovery	1 SOUE parameter list 2 LMLM parameter list
SO 0508	DFHSOUE	Exc	Release lock error	1 SOUE parameter list 2 LMLM parameter list
SO 0509	DFHSOUE	Exc	Recovery release lock error	1 SOUE parameter list 2 LMLM parameter list
SO 050A	DFHSOUE	Exc	Unknown KE error code	
SO 0601	DFHSOAD	SO 1	Entry	1 SOAD parameter list
SO 0602	DFHSOAD	SO 1	Exit	1 SOAD parameter list

Table 41. Socket domain trace points (continued)

Point ID	Module	Lvl	Type	Data
SO 0603	DFHSOAD	Exc	Recovery	1 SOAD parameter list
				2 Kernel error data
SO 0604	DFHSOAD	Exc	Invalid format	1 SOAD parameter list
SO 0605	DFHSOAD	Exc	Invalid function	1 SOAD parameter list
SO 0606	DFHSOAD	Exc	Unlock error recovery	1 SOAD parameter list
				2 LMLM parameter list
SO 0607	DFHSOAD	Exc	Storage error	1 SMGF parameter list
SO 0701	DFHSOTB	SO 1	Entry	1 SOTB parameter list
SO 0702	DFHSOTB	SO 1	Exit	1 SOTB parameter list
SO 0703	DFHSOTB	Exc	Recovery	1 SOTB parameter list
				2 Kernel error data
SO 0704	DFHSOTB	Exc	Invalid format	1 SOTB parameter list
SO 0705	DFHSOTB	Exc	Invalid function	1 SOTB parameter list
SO 0706	DFHSOTB	Exc	Unlock error recovery	1 SOTB parameter list
				2 LMLM parameter list
SO 0707	DFHSOTB	Exc	Storage error	1 SMGF parameter list
SO 0801	DFHSOSE	SO 1	Entry	1 SOSE parameter list
SO 0802	DFHSOSE	SO 1	Exit	1 SOSE parameter list
SO 0803	DFHSOSE	Exc	Recovery	1 SOSE parameter list
				2 Kernel error data

Trace Entries

Table 41. Socket domain trace points (continued)

Point ID	Module	Lvl	Type	Data
SO 0804	DFHSOSE	Exc	Invalid format	1 SOSE parameter list
SO 0805	DFHSOSE	Exc	Invalid function	1 SOSE parameter list
SO 0806	DFHSOSE	Exc	Lock error	1 SOSE parameter list 2 LMLM parameter list
SO 0807	DFHSOSE	Exc	Unlock error	1 SOSE parameter list 2 LMLM parameter list
SO 0808	DFHSOSE	Exc	Storage error	1 SMGF parameter list
SO 0809	DFHSOSE	SO 2	Before CEEPIPI	1 gsk_interface
SO 080A	DFHSOSE	SO 2	After CEEPIPI	1 gsk_interface
SO 080B	DFHSOSE	Exc	CEEPIPI error	1 gsk_interface
SO 080C	DFHSOSE	Exc	GSK error	1 gsk_interface
SO 080D	DFHSOSE	Exc	Getmain error	
SO 080E	DFHSOSE	Exc	Repository error	
SO 080F	DFHSOSE	Exc	Directory error	1 SOSE parameter list
SO 0901	DFHSOXM	SO 1	Entry	1 MXM parameter list
SO 0902	DFHSOXM	SO 1	Exit	1 MXM parameter list
SO 0903	DFHSOXM	Exc	Invalid function	1 MXM Parameter list
SO 0904	DFHSOXM	Exc	Invalid format	1 MXM parameter list
SO 0905	DFHSOXM	Exc	Recovery	1 MXM parameter list 2 Kernel error data

Table 41. Socket domain trace points (continued)

Point ID	Module	Lvl	Type	Data
SO 0A01	DFHSOST	SO 1	Entry	1 STST parameter list
SO 0A02	DFHSOST	SO 1	Exit	1 STST parameter list
SO 0A03	DFHSOST	Exc	Recovery	1 STST parameter list 2 Kernel error data
SO 0A04	DFHSOST	Exc	Invalid format	1 STST parameter list
SO 0A05	DFHSOST	Exc	Invalid function	1 STST parameter list
SO 0A06	DFHSOST	Exc	Exclusive lock error	1 STST parameter list 2 LMLM parameter list
SO 0A07	DFHSOST	Exc	Exclusive unlock error	1 STST parameter list 2 LMLM parameter list
SO 0A08	DFHSOST	Exc	Shared lock error	1 STST parameter list 2 LMLM parameter list
SO 0A09	DFHSOST	Exc	Shared unlock error	1 STST parameter list 2 LMLM parameter list
SO 0A10	DFHSOST	Exc	Recovery release lock error	1 STST parameter list 2 LMLM parameter list
SO 0A11	DFHSOST	Exc	Getmain error	1 SMGF parameter list
SO 0A12	DFHSOST	Exc	Unknown kernel error code	1 STST parameter list 2 Kernel error data
SO 0A13	DFHSOST	Exc	Invalid parameters	1 STST parameter list

Trace Entries

Table 41. Socket domain trace points (continued)

Point ID	Module	Lvl	Type	Data
SO 0B01	DFHSOTI	SO 1	Entry	1 SO anchor block
SO 0B02	DFHSOTI	SO 1	Exit	1 SO anchor block
SO 0B03	DFHSOTI	SO 1	Before posting listener	1 1 TISR parameter list
SO 0B04	DFHSOTI	SO 1	After posting listener	1 1 TISR parameter list
SO 0B05	DFHSOTI	Exc	Recovery entry	1 SO anchor block 2 Kernel error data
SO 0B06	DFHSOTI	Exc	Recovery exit	1 SO anchor block 2 Kernel error data
SO 0B07	DFHSOTI	Exc	Invalid formatn	1 TISR parameter list 2 SO anchor block
SO 0B08	DFHSOTI	Exc	Invalid function	1 TISR parameter list 2 SO anchor block
SO 0B09	DFHSOTI	SO 1	Timeout receive entry	1 1 TISR parameter list
SO 0B0A	DFHSOTI	SO 1	Timeout receive exit	1 1 TISR parameter list
SO 0B0B	DFHSOTI	Exc	Receive cancel exc	1 1 TISR parameter list

Web domain trace points

Table 42. Web domain trace points

Point ID	Module	Lvl	Type	Data
WB 0100	DFHWBDM	WB 1	Entry	1 WBDM parameter list
WB 0101	DFHWBDM	WB 1	Exit	1 WBDM parameter list

Table 42. Web domain trace points (continued)

Point ID	Module	Lvl	Type	Data
WB 0102	DFHWBDM	Exc	Recovery	1 WBDM parameter list
WB 0103	DFHWBDM	Exc	Invalid format	1 DMDM parameter list
WB 0104	DFHWBDM	Exc	Invalid function	1 DMDM parameter list
WB 0105	DFHWBDM	Exc	Release lock error	1 DMDM parameter list 2 LMLM parameter list
WB 0107	DFHWBDM	Exc	No storage for anchor block	1 DMDM parameter list
WB 0108	DFHWBDM	Exc	Get parameters failed	1 DMDM parameter list
WB 0200	DFHWBGP	WB 1	Entry	1 WBGP parameter list
WB 0201	DFHWBGP	WB 1	Exit	1 WBGP parameter list
WB 0202	DFHWBGP	Exc	Recovery	1 WBGP parameter list
WB 0203	DFHWBGP	Exc	Invalid format	1 WBGP parameter list
WB 0204	DFHWBGP	Exc	Invalid function	1 WBGP parameter list
WB 0300	DFHWBAP	WB 1	Entry	1 WBAP parameter list 2 Client codepage
WB 0301	DFHWBAP	WB 1	Exit	1 WBAP parameter list 2 Client codepage
WB 0302	DFHWBAP	Exc	Recovery	1 WBAP parameter list
WB 0303	DFHWBAP	Exc	Invalid format	1 WBAP parameter list
WB 0304	DFHWBAP	Exc	Invalid function	1 WBAP parameter list

Trace Entries

Table 42. Web domain trace points (continued)

Point ID	Module	Lvl	Type	Data
WB 0305	DFHWBAP	Exc	Unlock error recovery	1 WBAP parameter list
WB 0410	DFHWBRQ	Event	Inbound HTTP request	1 HTTP request headers
WB 0411	DFHWBSR	Exc	TSQ put header failed	1 WRB
WB 0412	DFHWBRQ	Exc	TS put request body failed	1 WRB
WB 0413	DFHWBSR	Exc	No repository token returned	1 WRB
WB 0414	DFHWBSR	Event	Before call to Analyzer	1 Analyzer parameter list
WB 0415	DFHWBSR	Event	After call to Analyzer	1 Analyzer parameter list
WB 0416	DFHWBSR	Data	HTTP request body	1 HTTP request body
WB 0500	DFHWBSR	WB 1	Entry	1 WBSR parameter list
WB 0501	DFHWBSR	WB 1	Exit	1 WBSR parameter list
WB 0502	DFHWBSR	Exc	Invalid format	1 WBSR parameter list
WB 0503	DFHWBSR	Exc	Invalid function	1 WBSR parameter list
WB 0600	DFHWBXM	WB 1	Entry	1 XMAC parameter list
WB 0601	DFHWBXM	WB 1	Exit	1 XMAC parameter list
WB 0602	DFHWBXM	Exc	Invalid format	1 XMAC parameter list
WB 0603	DFHWBXM	Exc	Invalid function	1 XMAC parameter list
WB 0604	DFHWBXM	Exc	Recovery	1 XMAC parameter list
WB 4109	DFHWBLT	Exc	Trigger partner failed	1 WBST parameter list

Table 42. Web domain trace points (continued)

Point ID	Module	Lvl	Type	Data
WB 4654	DFHWBST	Exc	Directory browse error	1 WBST parameter list
				2 DDBR parameter list
WB FF60	DFHQBQM	WB 1	Entry	1 WBQM parameter list
WB FF61	DFHQBQM	WB 1	Exit	1 WBQM parameter list
WB FF62	DFHQBQM	Exc	Recovery	1 WBQM parameter list
WB FF63	DFHQBQM	Exc	Invalid format	1 WBAP parameter list
WB FF64	DFHQBQM	Exc	Invalid function	1 WBQM parameter list
WB FF65	DFHQBQM	Exc	Invalid parameter	1 WBQM parameter list
WB FF66	DFHQBQM	Exc	TS queue error	1 WBQM parameter list

Bridge facility management

Table 43. Bridge facility management trace points

Point ID	Module	Lvl	Type	Data
AP 2140	DFHBRFM	BR 1	Entry	1 BRFM parameter list
AP 2141	DFHBRFM	BR 1	Exit	1 BRFM parameter list
AP 2142	DFHBRFM	Exc	Invalid format	1 BRFM parameter list
AP 2143	DFHBRFM	Exc	Invalid function	1 BRFM parameter list
AP 2144	DFHBRFM	Exc	Invalid token	1 BRFM parameter list
AP 2145	DFHBRFM	Exc	Invalid template	1 BRFM parameter list
AP 2146	DFHBRFM	Exc	Token not current	1 BRFM parameter list
AP 2147	DFHBRFM	Exc	GETMAIN failed	1 BRFM parameter list

Trace Entries

Table 43. Bridge facility management trace points (continued)

Point ID	Module	Lvl	Type	Data
AP 2148	DFHBRFM	Exc	No free name	1 BRFM parameter list
AP 2149	DFHBRFM	Exc	Name create failed	1 BRFM parameter list
AP 214A	DFHBRFM	Exc	Recovery entered	1 BRFM parameter list 2 Kernel error data
AP 214B	DFHBRFM	BR 1	Kept BFB expired	1 BFB name 2 BFB index 3 BFB expiry time
AP 214C	DFHBRFM	BR 1	Kept BFB used	1 BFB name 2 BFB index 3 BFB expiry time
AP 214D	DFHBRFM	BR 1	BFB added to keep chain	1 BFB name 2 BFB index 3 BFB expiry time
AP 214E	DFHBRFM	Exc	SNUS signon failed	1 BRFM parameter list
AP 2160	DFHBRMS	BR 1	Entry	1 Exec parameter list
AP 2161	DFHBRMS	BR 1	Exit	1 Exec parameter list
AP 2162	DFHBRMS	Exc	Recovery	1 Exec parameter list 2 Kernel error data
AP 2163	DFHBRTC	BR 1	Entry	1 Exec parameter list
AP 2164	DFHBRTC	BR 1	Exit	1 Exec parameter list

Table 43. Bridge facility management trace points (continued)

Point ID	Module	Lvl	Type	Data
AP 2165	DFHBRTC	Exc	Recovery	1 Exec parameter list 2 Kernel error data
AP 2166	DFHBRIC	BR 1	Entry	1 Exec parameter list
AP 2167	DFHBRIC	BR 1	Exit	1 Exec parameter list
AP 2168	DFHBRIC	Exc	Recovery	1 Exec parameter list 2 Kernel error data
AP 2169	DFHBRxx	Exc	Invalid function	1 Exec parameter list
AP 216A	DFHBRxx	Exc	Invalid group	1 Exec parameter list
AP 216B	DFHBRxx	Exc	Invalid EXEC function	1 Exec parameter list
AP 216C	DFHBRSP	BR 1	Entry	1 Exec parameter list
AP 216D	DFHBRSP	BR 1	Exit	1 Exec parameter list
AP 216E	DFHBRSP	Exc	Recovery	1 Exec parameter list 2 Kernel error data
AP 216F	DFHBRMS	BR 2	Call user exit	1 BRXA transaction area 2 BRXA command area
AP 2170	DFHBRMS	BR 2	Return from user exit	1 BRXA transaction area 2 BRXA command area
AP 2171	DFHBRTC	BR 2	Call user exit	1 BRXA transaction area 2 BRXA command area

Trace Entries

Table 43. Bridge facility management trace points (continued)

Point ID	Module	Lvl	Type	Data
AP 2172	DFHBRTC	BR 2	Return from user exit	1 BRXA transaction area 2 BRXA command area
AP 2173	DFHBRIC	BR 2	Call user exit	1 BRXA transaction area 2 BRXA command area
AP 2174	DFHBRIC	BR 2	Return from user exit	1 BRXA transaction area 2 BRXA command area
AP 2175	DFHBRSP	BR 2	Call user exit	1 BRXA transaction area 2 BRXA command area
AP 2176	DFHBRSP	BR 2	Return from user exit	1 BRXA transaction area 2 BRXA command area
AP 2177	DFHBRSP	Exc	Syncpoint change	1 BRSP parameter list 2 BRXA transaction area 3 BRXA command area
AP 2178	DFHBRMS	BR 2	Call formatter	1 BRXA transaction area 2 BRXA command area
AP 2179	DFHBRMS	BR 2	Formatter return	1 BRXA transaction area 2 BRXA command area
AP 217A	DFHBRTC	BR 2	Call formatter	1 BRXA transaction area 2 BRXA command area
AP 217B	DFHBRTC	BR 2	Formatter return	1 BRXA transaction area 2 BRXA command area

Table 43. Bridge facility management trace points (continued)

Point ID	Module	Lvl	Type	Data
AP 217C	DFHBRIC	BR 2	Call formatter	1 BRXA transaction area 2; BRXA command area
AP 217D	DFHBRIC	BR 2	Call formatter	1 BRXA transaction area 2 BRXA command area

Bridge facility management 2

Table 44. Bridge facility management trace points

Point ID	Module	Lvl	Type	Data
AP 2800	DFHBRAT	BR 1	Entry	1 BRAT parameter list
AP 2801	DFHBRAT	BR 1	Exit	1 BRAT parameter list
AP 2802	DFHBRAT	Exc	Invalid format	1 BRAT parameter list
AP 2803	DFHBRAT	Exc	Invalid function	1 BRAT parameter list
AP 2804	DFHBRAT	Exc	Recovery	1 BRAT parameter list 2 Kernel error data
AP 2805	DFHBRAT	Exc	Xmat sno	1 BRAT parameter list
AP 2806	DFHBRAT	Exc	Xmxd sno	1 BRAT parameter list
AP 2807	DFHBRAT	Exc	Xsrc sno	1 BRAT parameter list
AP 2808	DFHBRAT	Exc	Usad sno	1 BRAT parameter list
AP 2809	DFHBRAT	Exc	Brfm allocate failed	1 BRAT parameter list
AP 280A	DFHBRAT	Exc	Tm locate failed	1 BRAT parameter list
AP 2820	DFHBRIQ	BR 1	Entry	1 BRIQ parameter list

Trace Entries

Table 44. Bridge facility management trace points (continued)

Point ID	Module	Lvl	Type	Data
AP 2821	DFHBRIQ	BR 1	Exit	1 BRIQ parameter list
AP 2822	DFHBRIQ	Exc	Invalid format	1 BRIQ parameter list
AP 2823	DFHBRIQ	Exc	Invalid function	1 BRIQ parameter list
AP 2824	DFHBRIQ	Exc	Recovery	1 BRIQ parameter list 2 Kernel error data
AP 2840	DFHBRRM	BR 1	Entry	1 RMRO parameter list
AP 2841	DFHBRRM	BR 1	Exit	1 RMRO parameter list
AP 2842	DFHBRRM	Exc	Recovery	1 RMRO parameter list 2 Kernel error data
AP 2847	DFHBRRM	BR 2	Call user exit	1 BRXA transaction area 2 BRXA command area
AP 2848	DFHBRRM	BR 2	Return from user exit	1 BRXA transaction area 2 BRXA command area
AP 2860	DFHBRXM	BR 1	Entry	1 XMAC parameter list
AP 2861	DFHBRXM	BR 1	Exit	1 XMAC parameter list
AP 2862	DFHBRXM	Exc	Invalid format	1 XMAC parameter list
AP 2863	DFHBRXM	Exc	Invalid function	1 XMAC parameter list
AP 2864	DFHBRXM	Exc	Recovery	1 XMAC parameter list 2 Kernel error data
AP 2865	DFHBRXM	Exc	Smgf sno	1 XMAC parameter list

Table 44. Bridge facility management trace points (continued)

Point ID	Module	Lvl	Type	Data
AP 2866	DFHBRXM	Exc	Xmxd sno	1 XMAC parameter list
AP 2867	DFHBRXM	BR 2	Call user exit	1 BRXA transaction area 2 BRXA command area
AP 2868	DFHBRXM	BR 2	Return from user exit	1 BRXA transaction area 2 BRXA command area

AP domain recovery trace points

Table 45. AP domain recovery trace points

Point ID	Module	Lvl	Type	Data
AP 0785	DFHSRP	Exc	Abend AEYF	1 Application program name 2 Parameter address 3 ARGO

AP domain transaction initiation trace points

Table 46. AP domain transaction initiation trace points

Point ID	Module	Lvl	Type	Data
AP 1751	DFHZSUP	Exc	START TRANS W/ TERM DFHXMATM FUNCTION ATTACH FACILITY_TYPE(TERMINAL)	1 TCTTE

AP domain bridge facility trace points

Table 47. AP domain bridge facility trace points

Point ID	Module	Lvl	Type	Data
AP 286C	DFHBRXM	Exc	Security violation	1 Signed-on userid 2 Current userid

AP domain conversion trace points

Table 48. AP domain conversion trace points

Point ID	Module	Lvl	Type	Data
AP 4805	DFHCCNVG	Event	About to call z/VSE	1 z/VSE parameter list
AP 4806	DFHCCNVG	Event	Return from z/VSE	1 z/VSE parameter list
AP 4807	DFHCCNVG	Exc	Getmain failed	1 CCNVG parameter list
AP 4809	DFHCCNVG	Exc	Add lock failed	none
AP 480A	DFHCCNVG	Exc	Release lock error	1 CCNVG parameter list
AP 480B	DFHCCNVG	Exc	Get lock error	1 CCNVG parameter list
AP 480C	DFHCCNVG	Event	Converted data	1 Source buffer 2 Target buffer

AP domain container data transformation trace points

Table 49. AP domain container data transformation trace points

Point ID	Module	Lvl	Type	Data
AP 4E00	DFHAPCR	AP 1	Entry	1 APCR parameter list
AP 4E01	DFHAPCR	AP 1	Exit	1 APCR parameter list
AP 4E02	DFHAPCR	AP Exc	Invalid format	1 APCR parameter list
AP 4E03	DFHAPCR	AP Exc	Invalid function	1 APCR parameter list
AP 4E04	DFHAPCR	AP Exc	Recovery	1 APCR parameter list 2 Kernel error data
AP 4E05	DFHAPCR	AP Exc	Delete container failed	1 APCR parameter list 2 PGCR parameter list

Table 49. AP domain container data transformation trace points (continued)

Point ID	Module	Lvl	Type	Data
AP 4E06	DFHAPCR	AP Exc	Put container failed	1
				APCR parameter list
				2
				PGCR parameter list
AP 4E07	DFHAPCR	AP 2	Receive tioa	1
				tioa
				2
				03307400
AP 4E08	DFHAPCR	AP Exc	Create channel failed	1
				APCR parameter list
				2
				PGCR parameter list
				3
				03307900
AP 4E09	DFHAPCR	AP Exc	No room for channel header	1
				APCR parameter list
				2
				03308400
AP 4E0A	DFHAPCR	AP Exc	Getmain failed	1
				APCR parameter list
				2
				SMGF parameter list
				3
				03308900
AP 4E0B	DFHAPCR	AP Exc	DFHtc error	1
				Response
				2
				Abend code
				3
				Sense code
				4
				03309400
AP 4E0C	DFHAPCR	AP Exc	Extract total length	1
				APCR parameter list
				2
				Buffer left
				3
				03309900

Trace Entries

Table 49. AP domain container data transformation trace points (continued)

Point ID	Module	Lvl	Type	Data
AP 4E0D	DFHAPCR	AP Exc	Extract channel header	1
				APCR parameter list
				2
				Buffer left
				3
				03310400
AP 4E0E	DFHAPCR	AP Exc	Extract container header	1
				APCR parameter list
				2
				Buffer left
				3
				03310900
AP 4E0F	DFHAPCR	AP Exc	Premature end of data	1
				APCR parameter list
				2
				Buffer left
				3
				03311400
AP 4E10	DFHAPCR	AP Exc	More data expected	1
				APCR parameter list
				2
				Buffer left
				3
				03311900
AP 4E11	DFHAPCR	AP Exc	Extract container length	1
				APCR parameter list
				2
				Buffer left
				3
				03312100
AP 4E12	DFHAPCR	AP Exc	Bad channel eye-catcher	1
				APCR parameter list
				2
				Channel header
				3
				03312200
AP 4E13	DFHAPCR	AP Exc	Bad container eye-catcher	1
				APCR parameter list
				2
				Container header
				3
				03312300

Table 49. AP domain container data transformation trace points (continued)

Point ID	Module	Lvl	Type	Data
AP 4E14	DFHAPCR	AP 2	Extract channel length	1
				2
AP 4E15	DFHAPCR	AP 2	Extract channel header	1
				2
AP 4E16	DFHAPCR	AP 2	Extract container header	1
				2
AP 4E17	DFHAPCR	AP 2	Extract container length	1
				2
AP 4E18	DFHAPCR	AP 2	Extract container data	1
				2
AP 4E19	DFHAPCR	AP Exc	Bad response to domain call	1
				2
				3
AP 4E20	DFHAPCR	AP 2	Container added	1
				2
				3
AP 4E21	DFHAPCR	AP 2	Container changed	1
				2
				3
				4
				5

Trace Entries

Table 49. AP domain container data transformation trace points (continued)

Point ID	Module	Lvl	Type	Data
AP 4E22	DFHAPCR	AP 2	Container deleted	1 Container name
				2 Owner
				3 03313600

Program manager domain trace points

Table 50. Program manager domain trace entries

Point ID	Module	Lvl	Type	Data
PG 1701	DFHPGCH	PG 1	EXIT	1 PGCH parameter list
PG 1702	DFHPGCH	PG Exc	Invalid format	1 PGCH parameter list
PG 1703	DFHPGCH	PG Exc	Invalid function	1 PGCH parameter list
PG 1704	DFHPGCH	PG Exc	Recovery	1 PGCH parameter list
				2 Kernel error data
PG 1705	DFHPGCH	PG Exc	Smgf failure	1 PGCH parameter list
				2 SMGF parameter list
PG 1706	DFHPGCH	PG Exc	Create container pool error	1 PGCH parameter list
				2 PGCP parameter list
PG 1707	DFHPGCH	PG Exc	Copy container pool error	1 PGCH parameter list
				2 PGCP parameter list
PG 1708	DFHPGCH	PG 2	Add channel to plcb	1 Current chcb
				2 chcb chain
PG 1709	DFHPGCH	PG Exc	Invalid link level	1 PGCH parameter list
PG 1800	DFHPGCP	PG 1	Entry	1 PGCP parameter list

Table 50. Program manager domain trace entries (continued)

Point ID	Module	Lvl	Type	Data
PG 1801	DFHPGCP	PG 1	Exit	1 PGCP parameter list
PG 1802	DFHPGCP	PG Exc	Invalid format	1 PGCH parameter list
PG 1803	DFHPGCP	PG Exc	Invalid function	1 PGCP parameter list
PG 1804	DFHPGCP	PG Exc	Recovery	1 PGCP parameter list 2 Kernel error data
PG 1805	DFHPGCP	PG 2	Copy container	1 crcb
PG 1900	DFHPGCR	PG 1	Entry	1 PGCR parameter list
PG 1901	DFHPGCR	PG 1	Exit	1 PGCR parameter list
PG 1902	DFHPGCR	PG Exc	Invalid format	1 PGCR parameter list
PG 1903	DFHPGCR	PG Exc	Invalid function	1 PGCR parameter list
PG 1904	DFHPGCR	PG Exc	Recovery	1 PGCR parameter list 2 Kernel error data
PG 1905	DFHPGCR	PG Exc	Getmain crcb error	1 PGCR parameter list 2 SMGF parameter list
PG 1906	DFHPGCR	PG Exc	Getmain cscb error	1 PGCR parameter list 2 SMGF parameter list
PG 1907	DFHPGCR	PG Exc	Freemain crcb error	1 PGCR parameter list 2 SMGF parameter list
PG 1908	DFHPGCR	PG Exc	Freemain cscb error	1 PGCR parameter list 2 SMGF parameter list

Trace Entries

Table 50. Program manager domain trace entries (continued)

Point ID	Module	Lvl	Type	Data
PG 1909	DFHPGCR	PG Exc	Getmain set stor error	1 PGCR parameter list 2 SMGF parameter list
PG 190A	DFHPGCR	PG Exc	Freemain set stor error	1 PGCR parameter list 2 SMGF parameter list
PG 190B	DFHPGCR	PG Exc	Getmain crbb error	1 PGCR parameter list 2 SMGF parameter list
PG 190C	DFHPGCR	PG Exc	Freemain crbb error	1 PGCR parameter list 2 SMGF parameter list
PG 190D	DFHPGCR	PG 2	Locate container	1 cpcb 2 Container name 3 crcb ptr
PG 190E	DFHPGCR	PG Exc	Ccnv token error	1 CCNV parameter list
PG 190F	DFHPGCR	PG Exc	Ccnv convert error	1 CCNV parameter list
PG 1910	DFHPGCR	PG 2	Put container data	1 crcb 2 Container data
PG 1911	DFHPGCR	PG 2	Get container into data	1 crcb 2 Container data
PG 1912	DFHPGCR	PG 2	Get container set data	1 crcb 2 Container data

AP domain event manager trace points

Table 51. Event manager trace points

Point ID	Module	Lvl	Type	Data
AP F801	DFHEIBAM	AP 2	Entry	1 EIEI parameter list
AP F802	DFHEIBAM	AP 2	Exit	1 EIEI parameter list
AP F804	DFHEIBAM	Exc	Invalid format	1 EIEI parameter list
AP F805	DFHEIBAM	Exc	Invalid function	1 EIEI parameter list

CMCI domain trace points

Table 52. CMCI domain trace points

Point ID	Module	Lvl	Type	Data
AP A200	DFHWUIPG	WU 1	Entry	
AP A201	DFHWUIPG	WU 2	Entry	1 HTTP method 2 URI path
AP A202	DFHWUIPG	Exc	WEB EXTRACT Failed	1 RESP1 2 RESP2
AP A203	DFHWUIPG	Exc	DOCUMENT CREATE Failed	1 RESP1 2 RESP2
AP A204	DFHWUIPG	Exc	DOCUMENT SEND Failed	1 RESP1 2 RESP2
AP A205	DFHWUIPG	Exc	WEB EXTRACT Length Error	1 RESP1 2 RESP2
AP A206	DFHWUIPG	Exc	WEB RECEIVE Failed	1 RESP1 2 RESP2
AP A208	DFHWUIPG	Exc	Attribute GETMAIN failure	

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A209	DFHWUIPG	Exc	ParmAddr GETMAIN failure	1
				RESP1
				2
				RESP2
AP A20A	DFHWUIPG	WU 1	Exit	
AP A20B	DFHWUIPG	WU 2	HTTP body empty	
AP A20C	DFHWUIPG	Exc	DOCUMENT INSERT failure insert1	1
				RESP1
				2
				RESP2
AP A20D	DFHWUIPG	Exc	DOCUMENT INSERT failure insert2	1
				RESP1
				2
				RESP2
AP A20E	DFHWUIPG	Exc	DOCUMENT INSERT failure insert3	1
				RESP1
				2
				RESP2
AP A20F	DFHWUIPG	Exc	DOCUMENT INSERT failure insert4	1
				RESP1
				2
				RESP2
AP A210	DFHWUIPG	Exc	DOCUMENT SET failure Version	1
				RESP1
				2
				RESP2
AP A211	DFHWUIPG	Exc	DOCUMENT SET failure ErrorCode	1
				RESP1
				2
				RESP2
AP A212	DFHWUIPG	Exc	DOCUMENT INSERT failure Title	1
				RESP1
				2
				RESP2
AP A213	DFHWUIPG	Exc	DOCUMENT INSERT failure Message	1
				RESP1
				2
				RESP2
AP A214	DFHWUIPG	Exc	DOCUMENT INSERT failure Full	1
				RESP1
				2
				RESP2

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A215	DFHWUIPG	Exc	DOCUMENT INSERT failure Short	1
				RESP1
				2
				RESP2
AP A216	DFHWUIPG	Exc	DOCUMENT INSERT failure Info	1
				RESP1
				2
				RESP2
AP A217	DFHWUIPG	Exc	DFHWUPFM call failed	1 Return Code
AP A218	DFHWUIPG	Exc	WEB WRITE failure addHeader	1
				RESP1
				2
				RESP2
AP A219	DFHWUIPG	WU 2	WEB READ HTTPHEADER failure notfound	1
				RESP1
				2
				RESP2
AP A21A	DFHWUIPG	Exc	WEB READ HTTPHEADER failure InvalidRequest	1
				RESP1
				2
				RESP2
AP A21B	DFHWUIPG	Exc	WEB READ HTTPHEADER failure	1
				RESP1
				2
				RESP2
AP A21C	DFHWUIPG	Exc	WEB WRITE HTTPHEADER failure	1
				RESP1
				2
				RESP2
AP A21D	DFHWUIPG	Exc	DOCUMENT INSERT failure insert5	1
				RESP1
				2
				RESP2
AP A21E	DFHWUIPG	Exc	DOCUMENT INSERT failure insert6	1
				RESP1
				2
				RESP2
AP A21F	DFHWUIPG	Exc	CICS LOAD PROGRAM failure DFHEITBS	1
				RESP1
				2
				RESP2

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A220	DFHWUIPG	Exc	CICS LOAD PROGRAM failure DFHEITAB	1 RESP1 2 RESP2
AP A221	DFHWUIPG	Exc	CICS LOAD PROGRAM failure DFHEITSZ	1 RESP1 2 RESP2
AP A240	DFHWUPFM	WU 1	Entry	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block
AP A241	DFHWUPFM	WU 2	Entry	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 IPG_REQUEST_PATH
AP A242	DFHWUPFM	WU 2	URI Incorrectly specified	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 IPG_REQUEST_PATH
AP A243	DFHWUPFM	WU 2	ASSOCIATION failure	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 Association Value 5 Resource Name 6 Association Model Name

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A244	DFHWUPFM	WU 2	No Equal found in ASSOCIATION string	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 Association Value
AP A245	DFHWUPFM	WU 2	ASSOCIATION string incorrectly specified	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 Association Value
AP A247	DFHWUPFM	WU 2	CRITERIA element unknown	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 Query Address 5 Query Parameter 6 Query Parameter Value
AP A248	DFHWUPFM	WU 2	CONTEXT omitted	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block
AP A24C	DFHWUPFM	WU 2	Before DFHWURQP call	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A24D	DFHWUPFM	WU 2	After DFHWURQP call	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 Return Code
AP A24E	DFHWUPFM	Exc	DFHWURQP failed	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 Return Code
AP A24F	DFHWUPFM	WU 2	RESINGRP Version Value invalid	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 Association Value
AP A250	DFHWUPFM	WU 2	Association RESINGRP value incorrect	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 Association Value
AP A251	DFHWUPFM	WU 2	Association RESOUREC value incorrect	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 Association Value

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A252	DFHWUPFM	Exc	FREEMAIN failure CRITERIA	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 RESP1 5 RESP2 6 CRITERIA pointer
AP A253	DFHWUPFM	Exc	FREEMAIN failure PARAMETER	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 RESP1 5 RESP2 6 PARAMETER pointer
AP A254	DFHWUPFM	Exc	FREEMAIN failure ASSOCIATION	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 RESP1 5 RESP2 6 ASSOCIATION pointer
AP A256	DFHWUPFM	WU 2	DFHWUXML failure	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A257	DFHWUPFM	WU 2	XMLREQUEST type incorrectly specified	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block
AP A258	DFHWUPFM	WU 2	URI has too much data	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 IPG_REQUEST_PATH
AP A259	DFHWUPFM	WU 1	Exit	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block
AP A260	DFHWUPFM	WU 2	Invalid HTTP Method for CACHE request	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block
AP A261	DFHWUPFM	WU 2	Too many slashes in URI	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block
AP A262	DFHWUPFM	WU 2	Invalid HTTP Method	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A263	DFHWUPFM	WU 2	Invalid Record Count	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 IPG_REQUEST_PATH
AP A264	DFHWUPFM	WU 2	Invalid Record Index	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 IPG_REQUEST_PATH
AP A265	DFHWUPFM	WU 2	Invalid HTTP Method for Schema request	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block
AP A266	DFHWUPFM	WU 2	SCHEMA file omitted	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 IPG_REQUEST_PATH
AP A267	DFHWUPFM	WU 2	Rendering a SCHEMA failed	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block
AP A268	DFHWUPFM	WU 2	CACHE token exceeded length	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block 4 IPG_REQUEST_PATH

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A269	DFHWUPFM	WU 2	XML REQUEST ACTION invalid	1 IPG_REQUEST block 2 XML_REQUEST block 3 IPG_ERROR block
AP A270	DFHWURQP	WU 1	Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER
AP A271	DFHWURQP	WU 2	Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Model Name address 4 Output Format Address
AP A272	DFHWURQP	WU 2	Attempting to acquire CACHED ResultSet	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER
AP A273	DFHWURQP	WU 2	Finished attempted to acquire CACHED ResultSet	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 RSM_RESULTSET_HEADER above bar 4 Return Code
AP A274	DFHWURQP	WU 2	Building CPSM ResultSet	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Build Method

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A275	DFHWURQP	WU 2	CPSM ResultSet Built	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Object Token 4 Input ResultSet Token 5 Input Attribute token
AP A276	DFHWURQP	WU 2	DISCARD CPSM ResultSets started	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input Object Token 4 Input ResultSet Token 5 Input Attribute token
AP A277	DFHWURQP	WU 2	DISCARD CPSM ResultSets finished	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Return Code
AP A278	DFHWURQP	WU 2	Format Result Summary started	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input ResultSet Token 4 Input render records
AP A279	DFHWURQP	WU 2	Format Result Summary finished	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Return Code

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A27A	DFHWURQP	WU 2	Format ResultSet started	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Output Buffer 4 Input ResultSet Header
AP A27B	DFHWURQP	WU 2	Format ResultSet finished	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Return Code
AP A27C	DFHWURQP	WU 2	Before DFHWURSF call	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Output Buffer 4 Input ResultSet Header
AP A27D	DFHWURQP	WU 2	After DFHWURSF call	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Return Code
AP A27E	DFHWURQP	WU 2	CACHE ResultSet Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input ResultSet Token
AP A27F	DFHWURQP	WU 2	CACHE ResultSet Exit	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Output ResultSet Above Bar 4 Return Code

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A280	DFHWURQP	WU 2	Release ResultSet Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Output ResultSet Above Bar
AP A281	DFHWURQP	WU 2	Release ResultSet Exit	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Return Code
AP A282	DFHWURQP	WU 2	Unsupported HTTP Method	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER
AP A283	DFHWURQP	WU 2	Cached ResultSet retrieval failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Return Code
AP A284	DFHWURQP	WU 2	DISCARD CPSM ResultSet failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input ResultSet Token
AP A285	DFHWURQP	WU 2	CPSM DISCONNECT failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Return Code
AP A286	DFHWURQP	WU 2	CPSM FETCH ResultSet Above Bar failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input ResultSet Token 4 Return Code

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A287	DFHWURQP	WU 2	CPSM GET ResultSet failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 ResultSet Data Length 4 ResultSet Data Count 5 Input ResultSet Token 6 Api Function
AP A287	DFHWURQP	WU 2	CPSM GET ResultSet failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Api Response1 4 Return Code
AP A288	DFHWURQP	WU 2	CPSM GET Attributes failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 ResultSet Data Length 4 ResultSet Data Count 5 Input Attribute token 6 Return Code
AP A28A	DFHWURQP	Exc	CACHE ResultSet above bar failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Cache token 4 Cache area pointer 5 Storage length 6 Return Code

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A28B	DFHWURQP	Exc	Release ResultSet failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Return Code
AP A28C	DFHWURQP	WU 2	Before GETMAIN ResultSet Header	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Storage length
AP A28D	DFHWURQP	WU 2	After GETMAIN ResultSet Header	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Storage length 4 RESP1 5 RESP2
AP A28E	DFHWURQP	WU 2	GETMIAN ResultSet Header failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Storage length 4 RESP1 5 RESP2
AP A28F	DFHWURQP	WU 2	CPSM DISCARD attribute data failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input Attribute token 4 Return Code

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A290	DFHWURQP	WU 2	CPSM FETCH attribute to buffer failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input Attribute token 4 Input ResultSet Header 5 Attribute data offset 6 Attribute data length
AP A291	DFHWURQP	WU 2	CPSM CONNECT failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Return Code
AP A292	DFHWURQP	Exc	Copy Above bar storage failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Cache area pointer 4 Input WURSM below bar pointer 5 Object Record Length 6 Return Code
AP A294	DFHWURQP	WU 2	CPSM DISCARD object failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input Object Token 4 Return Code

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A295	DFHWURQP	WU 2	CPSM FETCH object failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Return Code 4 Build Method
AP A296	DFHWURQP	Exc	Failed to acquire cached ResultSet Header	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Return Code
AP A297	DFHWURQP	Exc	Failed to build CPSM ResultSet	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Below bar ResultSet Header 4 CPSM Object token 5 CPSM ResultSet token 6 CPSM Attribute token
AP A298	DFHWURQP	Exc	Failed to cache ResultSet Header	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Below bar ResultSet Header 4 CPSM Object token 5 CPSM ResultSet token 6 CPSM Attribute token

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A299	DFHWURQP	Exc	FREEMAIN failure below_bar_resultset_header_ptr	1
				IPG_REQUEST block
				2
				RSM_RESULTSET_HEADER
				3
Below bar ResultSet Header				
AP A29A	DFHWURQP	WU 2	CPSM UPDATE records Entry	4
				RESP1
				5
				RESP2
				1
AP A29A	DFHWURQP	WU 2	CPSM UPDATE records Entry	IPG_REQUEST block
				2
				RSM_RESULTSET_HEADER
AP A29B	DFHWURQP	WU 2	CPSMN UPDATE records Exit	3
				Input ResultSet Token
				1
AP A29B	DFHWURQP	WU 2	CPSMN UPDATE records Exit	IPG_REQUEST block
				2
				RSM_RESULTSET_HEADER
AP A29C	DFHWURQP	WU 2	CPSM UPDATE records failed	3
				Input ResultSet Token
				1
AP A29C	DFHWURQP	WU 2	CPSM UPDATE records failed	IPG_REQUEST block
				2
				RSM_RESULTSET_HEADER
AP A29D	DFHWURQP	Exc	CPSM UPDATE parameters invalid	3
				Input ResultSet Token
				1
AP A29D	DFHWURQP	Exc	CPSM UPDATE parameters invalid	IPG_REQUEST block
				2
				RSM_RESULTSET_HEADER
AP A29E	DFHWURQP	WU 2	CPSM PERFORM ACTION entry	3
				Input ResultSet Token
				1
AP A29E	DFHWURQP	WU 2	CPSM PERFORM ACTION entry	IPG_REQUEST block
				2
				RSM_RESULTSET_HEADER
AP A29E	DFHWURQP	WU 2	CPSM PERFORM ACTION entry	3
				Input ResultSet Token
				1
AP A29E	DFHWURQP	WU 2	CPSM PERFORM ACTION entry	IPG_REQUEST block
				2
				RSM_RESULTSET_HEADER

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A29F	DFHWURQP	WU 2	CPSM PERFORM ACTION exit	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input ResultSet Token
AP A2A0	DFHWURQP	WU 2	CPSM PERFORM ACTION failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input ResultSet Token
AP A2A1	DFHWURQP	Exc	CPSM PERFORM ACTION input failed validation	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input ResultSet Token
AP A2A2	DFHWURQP	WU 2	Exit	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Return Code
AP A2A3	DFHWURQP	WU 2	CPSM REMOVE ResultSets Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input ResultSet Token
AP A2A4	DFHWURQP	WU 2	CPSM REMOVE ResultSets Exit	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input ResultSet Token
AP A2A5	DFHWURQP	Exc	CPSM REMOVE ResultSets failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input ResultSet Token

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A2A6	DFHWURQP	Exc	CPSM REMOVE ResultSets input failed validation	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input ResultSet Token
AP A2A7	DFHWURQP	WU 2	CPSM REFRESH ResultSet Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER
AP A2A8	DFHWURQP	WU 2	CPSM REFRESH ResultSet Exit	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER
AP A2A9	DFHWURQP	WU 2	CPSM REFRESH ResultSet failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER
AP A2AA	DFHWURQP	WU 2	Record range start index out of bounds	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER
AP A2AB	DFHWURQP	WU 2	Record range start index specification not allowed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER
AP A2AC	DFHWURQP	WU 2	CPSM FETCH ResultSet failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input ResultSet Token
AP A2AE	DFHWURQP	WU 2	CPSM FETCH ResultSet Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Output Buffer 4 Input ResultSet Header

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A2AF	DFHWURQP	WU 2	CPSM FETCH ResultSet Exit	1
				IPG_REQUEST block
				2
				RSM_RESULTSET_HEADER
				3
				Output Buffer
AP A2B3	DFHWURQP	WU 2	Open CPSM CONNECTION Entry	1
				IPG_REQUEST block
				2
				RSM_RESULTSET_HEADER
AP A2B4	DFHWURQP	WU 2	Open CPSM CONNECTION Exit	1
				IPG_REQUEST block
				2
				RSM_RESULTSET_HEADER
				3
				Return Code
AP A2B5	DFHWURQP	WU 2	Format feedback records Entry	1
				IPG_REQUEST block
				2
				RSM_RESULTSET_HEADER
				3
				Output Buffer
AP A2B6	DFHWURQP	WU 2	Format feedback records Exit	1
				IPG_REQUEST block
				2
				RSM_RESULTSET_HEADER
				3
				Return Code
AP A2B7	DFHWURQP	WU 2	Create CPSM record Entry	1
				IPG_REQUEST block
				2
				RSM_RESULTSET_HEADER
				3
				Resource Record
				4
				Input Resource Record

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A2B8	DFHWURQP	WU 2	Create CPSM record Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Resource Record 4 Criteria 5 Return Code
AP A2B9	DFHWURQP	WU 2	Create CPSM record Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Resource Record 4 Criteria 5 Return Code
AP A2BA	DFHWURQP	WU 2	External to Internal conversion failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Attributes 4 Resource Record 5 Input Resource Record 6 Return Code
AP A2BB	DFHWURQP	WU 2	Invalid user specified Attribute	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Resource Record 4 WURQP_ATTRIBUTE

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A2BC	DFHWURQP	WU 2	Before External to Internal conversion	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Attributes 4 Resource Record 5 Input Resource Record 6 WURQP_ATTRIBUTE in value
AP A2BD	DFHWURQP	WU 2	After External to Internal conversion	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Attributes 4 Resource Record 5 Input Resource Record 6 WURQP_ATTRIBUTE in value
AP A2BE	DFHWURQP	WU 2	Rebuild CPSM Resultset Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input ResultSet Token
AP A2BF	DFHWURQP	WU 2	Rebuild CPSM Resultset Exit	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER
AP A2C0	DFHWURQP	WU 2	Create not valid for RESOURCE	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER
AP A2C1	DFHWURQP	WU 2	DEFVER specified as zero	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 WURQP_ATTRIBUTE

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A2C2	DFHWURQP	WU 2	Update Index none cached Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER
AP A2C3	DFHWURQP	WU 2	Update Index none cached Exit	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Return Code
AP A2C4	DFHWURQP	WU 2	Update Index cached Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input WURSM pointer
AP A2C5	DFHWURQP	WU 2	Update Index cached Exit	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Return Code
AP A2C7	DFHWURQP	WU 2	Invalid SMSS Object	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Local Object Pointer 4 Return Code
AP A2C8	DFHWURQP	Exc	CPSM TRANSLATE failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Response 4 Reason 5 Input for translation

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A2C9	DFHWURQP	Exc	CPSM REFRESH ResultSet limit reached	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 ResultSet Data Length 4 ResultSet Data Count 5 ResultSet Token
AP A2CA	DFHWURQP	Exc	Bad Environment	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 WUI Activity flag 4 Return Code
AP A2CD	DFHWURQP	WU 1	Entry Sort Index	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Sort operation type
AP A2CE	DFHWURQP	WU 1	Exit Sort Index	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Return code
AP A2CF	DFHWURQP	WU 1	Exit	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Return code
AP A2D0	DFHWURSM	WU 1	Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Return code

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A2D1	DFHWURS M	WU 2	Entry	1 Operation Type 2 Cache Token 3 Storage Class 4 Data Pointer 5 Copy Pointer
AP A2DA	DFHWURS M	Exc	Invalid Operation	1 Operation Type
AP A2DE	DFHWURS M	Exc	Insufficient storage to register	1 Global storage area
AP A2E1	DFHWURS M	Exc	Invalid Directory	1 Global storage area
AP A2E4	DFHWURS M	Exc	Bad UserID	1 Owners UserID 2 CICS UserID
AP A2E5	DFHWURS M	WU 1	Exit	1 Return Code
AP A2E6	DFHWURS M	WU 2	Lock Cache Entry	1 Cache Token
AP A2E7	DFHWURS M	WU 2	Lock Cache Exit	1 Cache Token 2 Output Storage Pointer 3 Output Storage Length 4 Output Storage Class 5 Return Code
AP A2E8	DFHWURS M	WU 2	Unlock Cache Entry	1 Cache Token 2 Unlock type

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A2E9	DFHWURS M	WU 2	Unlock Cache Exit	1 Cache Token 2 Local metadata userID 3 Return Code
AP A2EA	DFHWURS M	WU 2	Create Cache Entry	1 Storage Class 2 Storage Length 3 Cache token 4 Copy Pointer
AP A2EB	DFHWURS M	WU 2	Create Cache Exit	1 Storage Pointer 2 Local Cache Token 3 Local Metadata 4 Above the bar pointer 5 Return Code
AP A2EE	DFHWURS M	Exc	Lock not found	1 Global lock token
AP A2EF	DFHWURS M	Exc	ACQUIRE lock failed	1 Trace point ID 2 Global lock token 3 Response 4 Reason
AP A2F0	DFHWURS M	WU 2	Add to LinkedList Entry	1 Cache area
AP A2F1	DFHWURS M	Exc	Cache token already registered	1 Global cache area
AP A2F2	DFHWURS M	Exc	Add cache token failed	1 Trace point ID 2 Global cache area

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A2F3	DFHWURS M	Exc	Release lock failed	1 Trace point ID
				2 Global lock token
				3 Response
				4 Reason
AP A2F4	DFHWURS M	WU 2	Add to LinkedList Exit	1 Local Metadata
				2 Local LinkedList
				3 Return Code
AP A2F5	DFHWURS M	Exc	Cache token not found	1 Global cache token
AP A2F6	DFHWURS M	Exc	Cache token lookup action failed	1 Trace point ID
				2 Global cache token
				3 Response
				4 Reason
AP A2F7	DFHWURS M	WU 2	Remove from LinkedList Entry	1 Cache area
AP A2F8	DFHWURS M	WU 2	Remove from LinkedList Exit	1 Local Metadata
				2 Local LinkedList
				3 Return Code
AP A2F9	DFHWURS M	WU 2	Garbage Collection Entry	1 Cache Token
AP A2FA	DFHWURS M	WU 2	Garbage Collection Exit	1 Return Code

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A2FB	DFHWURS M	Exc	Cache token delete failed	1 Trace point ID 2 Global cache token 3 Cache token 4 Response 5 Reason
AP A2FC	DFHWURS M	Exc	Unlock Cache bad metadata	1 Metadata
AP A2FD	DFHWURS M	WU 2	Destroy region Entry	1 Metadata 2 LinkedList operation
AP A2FE	DFHWURS M	WU 2	Destroy region Exit	1 Return Code
AP A2FF	DFHWURS M	WU 2	Entry Delete Sort Index	
AP A300	DFHWURS M	Exc	Sort Program Link failed	
AP A301	DFHWURS M	Exc	Delete Sort Index failed	
AP A302	DFHWURS M	WU 2	Exit Delete Sort Index	1 Return Code
AP A303	DFHWURS M	WU 1	Exit	1 Return Code
AP A310	DFHWUWE B	WU 1	Entry	
AP A311	DFHWUWE B	WU 2	Entry	1 Output buffer 2 Output buffer length 3 Media type
AP A312	DFHWUWE B	Exc	Invalid WEB SEND	1 RESP1 2 RESP2
AP A313	DFHWUWE B	WU 1	Exit	

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A320	DFHWUEYU	WU 1	Entry	
AP A321	DFHWUEYU	WU 2	Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Input buffer 4 Input buffer above the bar 5 Input buffer length 6 Input buffer length
AP A322	DFHWUEYU	Exc	Invalid Operation type	1 IPG_REQUEST block 2 Operation type
AP A323	DFHWUEYU	WU 2	Open CPSM CONNECTION Entry	1 IPG_REQUEST block
AP A324	DFHWUEYU	WU 2	Open CPSM CONNECTION Exit	1 IPG_REQUEST block 2 Response 3 Reason 4 Return Code
AP A325	DFHWUEYU	WU 2	Close CPSM CONNECTION Entry	1 IPG_REQUEST block 2 Connection Thread
AP A326	DFHWUEYU	WU 2	Close CPSM CONNECTION Exit	1 IPG_REQUEST block 2 Connection Thread 3 Response 4 Reason 5 Return Code

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A327	DFHWUEYU	WU 2	GET ResultSet Entry	1 IPG_REQUEST block 2 ResultSet Token
AP A328	DFHWUEYU	WU 2	GET ResultSet Exit	1 IPG_REQUEST block 2 Count 3 Length 4 Response 5 Reason 6 Return Code
AP A329	DFHWUEYU	WU 2	Get Attributes Entry	1 IPG_REQUEST block 2 ResultSet Token 3 Count 4 Length
AP A32A	DFHWUEYU	WU 2	Get Attributes Exit	1 IPG_REQUEST block 2 Count 3 Length 4 Response 5 Reason 6 Return Code

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A32B	DFHWUEYU	WU 2	FETCH Attributes Entry	1 IPG_REQUEST block 2 ResultSet Token 3 Buffer Pointer 4 Buffer Length 5 Record Position 6 Fetch Style
AP A32C	DFHWUEYU	WU 2	FETCH Attributes Exit	1 IPG_REQUEST block 2 Response 3 Reason 4 Return Code
AP A32D	DFHWUEYU	WU 2	DISCARD CPSM ResultSet Entry	1 IPG_REQUEST block 2 ResultSet Token
AP A32E	DFHWUEYU	WU 2	DISCARD CPSM ResultSet Exit	1 IPG_REQUEST block 2 Response 3 Reason 4 Return Code
AP A32F	DFHWUEYU	WU 2	UPDATE CPSM ResultSet Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 ResultSet Token

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A330	DFHWUEYU	WU 2	UPDATE CPSM ResultSet Exit	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Response 4 Reason 5 Return Code
AP A331	DFHWUEYU	WU 2	FETCH Resource table OBJECT	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 ResultSet Token 4 Buffer Pointer 5 Buffer Length
AP A332	DFHWUEYU	WU 2	FETCH Resource table OBJECT	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Get Def count 4 Object data Count 5 Response 6 Reason
AP A333	DFHWUEYU	WU 2	DISCARD CPSM ResultSet Failed	1 IPG_REQUEST block 2 Connection Thread 3 Response 4 Reason

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A334	DFHWUEYU	WU 2	GET ResultSet failed	1 IPG_REQUEST block 2 Criteria 3 Parameter 4 Response 5 Reason
AP A335	DFHWUEYU	WU 2	GETDEF Attributes failed	1 IPG_REQUEST block 2 ResultSet Token 3 Response 4 Reason 5 Return Code
AP A336	DFHWUEYU	WU 2	QUERY Attribute failed	1 IPG_REQUEST block 2 ResultSet Token 3 Response 4 Reason 5 Return Code
AP A337	DFHWUEYU	WU 2	QUERY ResultSet Failed	1 IPG_REQUEST block 2 Connection Thread 3 ResultSet Token 4 Response 5 Reason

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A338	DFHWUEYU	WU 2	FETCH ResultSet failed	1 IPG_REQUEST block 2 Connection Thread 3 Response 4 Reason 5 Return Code
AP A339	DFHWUEYU	WU 2	UPDATE Definitions Failed	1 IPG_REQUEST block 2 Resource name 3 Modification address 4 Modification Length 5 Response 6 Reason
AP A33A	DFHWUEYU	WU 2	UPDATE Resource Failed	1 IPG_REQUEST block 2 Modification address 3 Modification Length 4 Response 5 Reason
AP A33B	DFHWUEYU	WU 2	CPSM GETDEF Object Failed	1 IPG_REQUEST block 2 Resource name 3 ResultSet Token 4 Connection Thread 5 Response 6 Reason

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A33C	DFHWUEYU	WU 2	CPSM FETCH object failed	1 IPG_REQUEST block 2 Buffer Pointer 3 Buffer Length 4 ResultSet Token 5 Connection Thread 6 Response
AP A33D	DFHWUEYU	WU 2	PERFORM action Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 ResultSet Token
AP A33E	DFHWUEYU	WU 2	PERFORM action Exit	1 IPG_REQUEST block 2 Response 3 Reason 4 Return Code
AP A33F	DFHWUEYU	WU 2	PERFORM action Failed	1 IPG_REQUEST block 2 ResultSet Token 3 XML Parameter address pointer 4 XML Parameter address 5 Response 6 Reason

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A340	DFHWUEYU	WU 1	Exit	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 I/O Token 4 Output 5 Length 6 Return Code
AP A341	DFHWUEYU	WU 2	REMOVE Definitions Failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 ResultSet Token 4 Object 5 Response 6 Reason
AP A342	DFHWUEYU	WU 2	REMOVE Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 ResultSet Token
AP A343	DFHWUEYU	WU 2	REMOVE Exit	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 ResultSet Token 4 Object Structure pointer 5 Response 6 Reason

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A344	DFHWUEYU	WU 2	REFRESH ResultSet Entry	1 IPG_REQUEST block 2 ResultSet Token
AP A345	DFHWUEYU	WU 2	REFRESH ResultSet Exit	1 IPG_REQUEST block 2 Response 3 Reason 4 Return Code
AP A346	DFHWUEYU	WU 2	REFRESH ResultSet Failed	1 IPG_REQUEST block 2 Connection Thread 3 Response 4 Reason 5 Return Code
AP A348	DFHWUEYU	WU 2	Obtain Feedback Entry	1 IPG_REQUEST block 2 ResultSet Token
AP A349	DFHWUEYU	WU 2	Obtain Feedback Exit	1 IPG_REQUEST block
AP A34A	DFHWUEYU	WU 2	Before Feedback Storage	1 IPG_REQUEST block 2 Old Count 3 New Count 4 Requested Count 5 Space required

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A34B	DFHWUEYU	WU 2	After Feedback Storage	1 IPG_REQUEST block 2 New Pointer 3 Space Required 4 RESP1 5 RESP2
AP A34F	DFHWUEYU	WU 2	CPSM FEEDBACK first failed	1 IPG_REQUEST block 2 Response 3 Reason
AP A350	DFHWUEYU	WU 2	CPSM FEEDBACK next failed	1 IPG_REQUEST block 2 Response 3 Reason
AP A351	DFHWUEYU	WU 2	Process feedback Entry	1 IPG_REQUEST block 2 Feedback pointer 3 WUEYU feedback pointer
AP A352	DFHWUEYU	WU 2	Process feedback Exit	1 IPG_REQUEST block 2 Feedback pointer
AP A353	DFHWUEYU	WU 2	Obtain Feedback before fetch	1 IPG_REQUEST block 2 Objstat 3 Feedback result record id 4 Fetch count 5 ResultSet Token

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A354	DFHWUEYU	WU 2	Obtain Feedback after fetch	1 IPG_REQUEST block 2 Objstat record 3 Objstat length 4 Fetch count 5 Response 6 Reason
AP A357	DFHWUEYU	WU 2	Process BINSTERR Entry	1 IPG_REQUEST block 2 BINSTERR record 3 WUEYU BINSTERR
AP A358	DFHWUEYU	WU 2	Process BINSTERR Exit	1 IPG_REQUEST block 2 WUEYU BINSTERR
AP A359	DFHWUEYU	WU 2	Process BINSONSC Entry	1 IPG_REQUEST block 2 BINCONSC record 3 WUEYU BINCONSC
AP A35A	DFHWUEYU	WU 2	Process BINSONSC Exit	1 IPG_REQUEST block 2 WUEYU BINCONSC
AP A35B	DFHWUEYU	WU 2	Process BINCONRS Entry	1 IPG_REQUEST block 2 BINCONRS record 3 WUEYU BINCONRS
AP A35C	DFHWUEYU	WU 2	Process BINCONRS Exit	1 IPG_REQUEST block 2 WUEYU BINCONRS

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A360	DFHWUEYU	WU 2	Obtain Feedback before fetch 2	1 IPG_REQUEST block 2 BAS data length 3 BAS index 4 BAS data count 5 WUEYU feedback error resul
AP A361	DFHWUEYU	WU 2	Obtain Feedback After fetch 2	1 IPG_REQUEST block 2 BAS data length 3 BAS index 4 BAS data count 5 Response 6 Reason
AP A363	DFHWUEYU	WU 2	Grow error info Entry	1 IPG_REQUEST block 2 Space allocated
AP A364	DFHWUEYU	WU 2	Grow error info Exit	1 IPG_REQUEST block 2 Return Code
AP A365	DFHWUEYU	WU 2	CPSM CREATE resource routine Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Resource Record
AP A366	DFHWUEYU	WU 2	CPSM CREATE resource routine Exit	1 IPG_REQUEST block 2 Response 3 Reason 4 Return Code

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A367	DFHWUEYU	WU 2	CPSM CREATE resource failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Resource Record 4 Response 5 Reason 6 Return Code
AP A368	DFHWUEYU	WU 2	FETCH ResultSet above Entry	1 IPG_REQUEST block 2 ResultSet Token 3 Buffer Above the bar 4 Buffer Length 5 Record Position 6 Count
AP A369	DFHWUEYU	WU 2	FETCH ResultSet above Exit	1 IPG_REQUEST block 2 Response 3 Reason 4 Return Code
AP A36A	DFHWUEYU	WU 2	FETCH Object above Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 ResultSet Token 4 Buffer Above the bar 5 Buffer Length

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A36B	DFHWUEYU	WU 2	FETCH Object above Exit	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 CPSM GET DEF count 4 Object data Count 5 Response 6 Reason
AP A36C	DFHWUEYU	WU 2	After Feedback FREEMAIN	1 IPG_REQUEST block 2 Error info Records address 3 RESP1 4 RESP2
AP A36D	DFHWUEYU	Exc	Start CPSMAPI bad ENQ	1 IPG_REQUEST block 2 Operation type 3 ENQ name 4 RESPS
AP A36E	DFHWUEYU	Exc	Start CPSMAPI bad LINK	1 IPG_REQUEST block 2 Operation type 3 RESPS
AP A36F	DFHWUEYU	Exc	Start CPSMAPI bad DE'Queue	1 IPG_REQUEST block 2 Operation type 3 ENQ name 4 RESPS

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A370	DFHWUEYU	Exc	Start CPSMAPI bad Assignment	1 IPG_REQUEST block 2 Operation type 3 RESPS
AP A371	DFHWUEYU	Exc	Start CPSMAPI bad Connection	1 IPG_REQUEST block 2 Operation type 3 Response 4 Reason 5 Default RMAS Applid 6 CPSM Connection Version
AP A372	DFHWUEYU	Exc	Start CPSMAPI ENQ busy	1 IPG_REQUEST block 2 Operation type 3 ENQ name
AP A373	DFHWUEYU	WU 2	Query record count entry	1 IPG_REQUEST block
AP A374	DFHWUEYU	WU 2	Query record count exit	1 IPG_REQUEST block 2 Response 3 Reason

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A375	DFHWUEYU	Exc	Query record count failed	1 IPG_REQUEST block 2 Context name 3 Scope name 4 Criteria 5 Parameters 6 Response 7 Reason
AP A376	DFHWUEYU	Exc	Query record count exceeded	1 IPG_REQUEST block 2 Default warning count 3 Current count
AP A377	DFHWUEYU	WU 2	Order result set entry	1 IPG_REQUEST block 2 Result token
AP A378	DFHWUEYU	WU 2	Order result set exit	1 IPG_REQUEST block 2 Response 3 Reason 4 Return code
AP A379	DFHWUEYU	Exc	Order result set exit failed	1 IPG_REQUEST block 2 Connection thread 3 Response 4 Reason 5 Return code

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A37A	DFHWUEYU	WU 1	Exit	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 I/O Token 4 Count 5 Length 6 Return code
AP A380	DFHWUIN1	WU 1	Entry	1 DFHROINM parameter list
AP A381	DFHWUIN1	Exc	Invaild ROIN Function	1 DFHROINM parameter list
AP A382	DFHWUIN1	Exc	System task attach failed	1 System Task
AP A383	DFHWUIN1	Exc	Recovery routine entered	1 DFHROINM parameter list 2 Kernel error data
AP A385	DFHWUCPA	WU 1	Entry	
AP A386	DFHWUCPA	WU 2	Entry	1 Association Name 2 Association Value 3 Association Version 4 Resource Name 5 Connection Thread
AP A387	DFHWUCPA	WU 2	Before GET	1 Association Name 2 Criteria Value 3 Connection Thread

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A388	DFHWUCPA	WU 2	After GET	1 Record Count 2 API Result 3 API Response 4 API Reason
AP A389	DFHWUCPA	Exc	Bad GET	1 Association Name 2 Criteria Value 3 Connection Thread 4 Record Count 5 API Result 6 API Response
AP A38A	DFHWUCPA	WU 2	No Data from GET	1 Association Name 2 Criteria Value
AP A38B	DFHWUCPA	WU 2	Before GET OBJECT	1 Association Name 2 Connection Thread
AP A38C	DFHWUCPA	WU 2	After GET OBJECT	1 Object data result 2 Response 3 Reason
AP A38D	DFHWUCPA	Exc	Bad GET OBJECT	1 Association Name 2 Connection Thread 3 Object data result 4 Response 5 Reason

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A38E	DFHWUCPA	WU 2	Before FETCH OBJECT	1 Object data result 2 Connection Thread
AP A38F	DFHWUCPA	WU 2	After FETCH OBJECT	1 Object storage 2 Response 3 Reason
AP A390	DFHWUCPA	Exc	Bad FETCH OBJECT	1 Object data result 2 Connection Thread 3 Object Storage 4 Response 5 Reason
AP A391	DFHWUCPA	WU 2	Before FETCH	1 API Result 2 Connection Thread
AP A392	DFHWUCPA	WU 2	After FETCH	1 Record Count 2 Fetch association value 3 API Response 4 API Reason
AP A393	DFHWUCPA	Exc	Bad FETCH	1 API Result 2 Connection Thread 3 Record Count 4 Fetch association value 5 API Response 6 API Reason

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A394	DFHWUCPA	WU 2	Before GET Attributes	1 Association Name 2 Link record attribute 3 Connection Thread
AP A395	DFHWUCPA	WU 2	After GET Attributes	1 Attribute data result 2 API Response 3 API Reason
AP A396	DFHWUCPA	Exc	Bad GET Attributes	1 Association Name 2 Link record attribute 3 Connection Thread 4 Attribute data result 5 API Response 6 API Reason
AP A397	DFHWUCPA	WU 2	Before FETCH Attributes	1 Attribute data result 2 Connection Thread
AP A398	DFHWUCPA	WU 2	After FETCH Attributes	1 Attribute storage 2 API Response 3 API Reason
AP A399	DFHWUCPA	Exc	Bad FETCH Attributes	1 Attribute data result 2 Connection Thread 3 Attribute storage 4 API Response 5 API Reason

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A39A	DFHWUCPA	WU 2	Before FILTER	1 Resource Name 2 Connection Thread 3 Filter buffer
AP A39B	DFHWUCPA	WU 2	After FILTER	1 Filter Token 2 API Response 3 API Reason
AP A39C	DFHWUCPA	Exc	Bad FILTER	1 Resource Name 2 Connection Thread 3 Filter buffer 4 Filter Token 5 API Response 6 API Reason
AP A39D	DFHWUCPA	Exc	Bad copy	1 IO ResultSet Token 2 New ResultSet Token 3 Connection Thread 4 API Response 5 API Reason
AP A39E	DFHWUCPA	Exc	Bad FILTER discard	1 Filter Token 2 Connection Thread 3 API Response 4 API Reason

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A39F	DFHWUCPA	Exc	Bad ResultSet discard	1 Filter Token 2 Connection Thread 3 API Response 4 API Reason
AP A3A0	DFHWUCPA	WU 2	Before FILTER 2	1 Resource Name 2 Connection Thread 3 Filter buffer
AP A3A1	DFHWUCPA	WU 2	After FILTER 2	1 Filter Token 2 API Response 3 API Reason
AP A3A2	DFHWUCPA	Exc	Bad Filter 2	1 Resource Name 2 Connection Thread 3 Filter buffer 4 Filter Token 5 API Response 6 API Reason
AP A3A3	DFHWUCPA	WU 1	Exit	1 IO ResultSet Token 2 Return Code
AP A3B0	DFHWURSF	WU 1	Entry	

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A3B1	DFHWURSF	WU 2	Entry	1
				Operation Type
				2
				Restful request
				3
				IPG Request
AP A3B2	DFHWURSF	WU 2	Entry Format	4
				Send type
				5
				Output buffer
				6
				ResultSet Header
AP A3B3	DFHWURSF	WU 2	Entry Format 2(a)	1
				Send type
				2
				Output Buffer
				3
				IPG Request
AP A3B4	DFHWURSF	WU 2	Exit Format	4
				ResultSet Header
				5
				ResultSet Range start index
				6
				ResultSet range count
AP A3B5	DFHWURSF	WU 2	Entry Format 2(b)	1
				ResultSet
				2
				Attributes
				3
				Object
AP A3B6	DFHWURSF	WU 2	Entry Format 2(c)	4
				Model name
				5
				ResultSet Record Length
				6
				ResultSet Record Count
AP A3B7	DFHWURSF	WU 2	Exit Format	1
				Attribute record length
AP A3B8	DFHWURSF	WU 2	Exit Format	2
				Attribute record count
AP A3B9	DFHWURSF	WU 2	Exit Format	1
				Output buffer

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A3B7	DFHWURSF	WU 2	Before internal to external attribute conversion	1
				Connection Thread
				2
				Record
AP A3B8	DFHWURSF	WU 2	After internal to external attribute conversion	3
				Attribute
				4
				Attribute Name
AP A3B8	DFHWURSF	WU 2	After internal to external attribute conversion	1
				Record
				2
				Attribute
AP A3B9	DFHWURSF	WU 2	Entry Summary(a)	3
				Buffer data
				4
				External Value length
AP A3B9	DFHWURSF	WU 2	Entry Summary(a)	1
				Output Buffer
				2
				Output Format
				3
				API Function
AP A3B9	DFHWURSF	WU 2	Entry Summary(b)	4
				API Response1
				5
				API Response2
				6
				Record Count
AP A3B9	DFHWURSF	WU 2	Entry Summary(b)	1
				ResultSet Cache Token
AP A3B9	DFHWURSF	WU 2	Entry Summary(b)	2
				Success Count
AP A3BA	DFHWURSF	Exc	Bad Summary Input	1
				Output Buffer
				2
				Output Format
				3
				API Function
AP A3BA	DFHWURSF	Exc	Bad Summary Input	4
				API Response1
				5
				API Response2
				6
				Record Count

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A3BB	DFHWURSF	Exc	Invalid Range values	1 ResultSet Range start index 2 ResultSet range count
AP A3BC	DFHWURSF	Exc	Invalid Operation	1 Operation Type
AP A3C2	DFHWURSF	Exc	Internal to External Attribute conversion ABEND	1 Connection Thread 2 Record 3 Attribute 4 Object
AP A3C3	DFHWURSF	WU 2	Entry Format Feedback	1 Output buffer 2 IPG request 3 Send Type
AP A3C4	DFHWURSF	WU 2	Exit Format Feedback	1 Return Code
AP A3C5	DFHWURSF	WU 2	Entry Format Keydata	1 Output Buffer 2 IPG request 3 Input Name pointer 4 Input Name Length 5 Keydata pointer 6 Keydata Length
AP A3C6	DFHWURSF	WU 2	Exit Format Keydata	1 Output buffer 2 Return Code

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A3C7	DFHWURSF	Exc	Invalid Buffer	1 Output buffer 2 Output data 3 Output data Length
AP A3C8	DFHWURSF	WU 2	Entry Format BINSTERR	1 Output buffer 2 IPG request 3 EYU BINSTERR
AP A3C9	DFHWURSF	WU 2	Exit Format BINSTERR	1 Return Code
AP A3CA	DFHWURSF	WU 2	Entry Format BINCONSC	1 Output buffer 2 IPG request 3 EYU BINCONSC
AP A3CB	DFHWURSF	WU 2	Exit Format BINCONSC	1 Return Code
AP A3CC	DFHWURSF	WU 2	Entry Format BINCONRS	1 Output buffer 2 IPG request 3 EYU BINCONRS
AP A3CD	DFHWURSF	WU 2	Exit Format BINCONRS	1 Return Code
AP A3CE	DFHWURSF	WU 1	Exit	
AP A3CF	DFHWURSF	WU 2	Exit	1 Resultful Request 2 IPG request 3 Return Code
AP A3D0	DFHWURSF	WU 2	Bad Feedback Record	1 EYU Error information 2 IDX Record

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A3D1	DFHWURSF	Exc	Bad Write Header	1 Response 2 Response2
AP A3D2	DFHWURSF	Exc	Bad Web Extract	1 Response 2 Response2
AP A3D3	DFHWURSF	Exc	Bad Handle	1 Return Code
AP A3D4	DFHWURSF	Exc	Bad Stack	1 Stack
AP A3D7	DFHWURSF	Exc	Internal to External Abend Data	1 Output Buffer Data 2 Length of buffer data
AP A400	DFHWURP	WU 1	Entry	
AP A401	DFHWURP	Exc	Create Subpool failed	1 Parameter list
AP A402	DFHWURP	Exc	Create Directory failed	1 DDDI Type request parameter list
AP A404	DFHWURP	WU 1	Exit	
AP A405	DFHWURP	Exc	Create Cache lock failed	1 LMLM Type request parameter list
AP A406	DFHWURP	Exc	Add Lock token to Directory failed	1 DDDI Type request parameter list
AP A407	DFHWURP	Exc	Allocate LinkedList meta failed	1 S2GF Type request parameter list
AP A408	DFHWURP	Exc	Add meta to Directory failed	1 DDDI Type request parameter list
AP A409	DFHWURP	Exc	Unlock cache failed	1 LMLM Type request parameter list
AP A40A	DFHWURQP	Exc	Copy LinkedList meta failed	1 S2SR Type request parameter list
AP A40B	DFHWURP	Exc	Add Enqueue To Directory File	1 DDDI Type request parameter list
AP A40C	DFHWURP	Exc	Create Sort Enqueue	1 NQNQ Type request parameter list
AP A420	DFHWUXML	WU 1	Entry	

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A421	DFHWUXML	Exc	Request tag failed parsing	
AP A422	DFHWUXML	Exc	Request tag name invalid	1 Name 2 Attribute count
AP A423	DFHWUXML	Exc	Method tag failed parsing	
AP A424	DFHWUXML	Exc	Method tag name invalid	
AP A425	DFHWUXML	Exc	Create tag failed parsing	
AP A426	DFHWUXML	Exc	Update tag failed parsing	
AP A427	DFHWUXML	Exc	Action tag failed parsing	
AP A428	DFHWUXML	Exc	Attributes/Parameters specified incorrectly	1 Attributes found 2 Parameters found
AP A429	DFHWUXML	Exc	Invalid End tag	1 Name 2 Type
AP A42A	DFHWUXML	Exc	XML within Create failed parsing	
AP A42B	DFHWUXML	Exc	XML within Create failed parsing	
AP A42C	DFHWUXML	Exc	XML within Create invalid End tag	
AP A42D	DFHWUXML	Exc	XML within Update failed parsing	
AP A42E	DFHWUXML	Exc	XML within Update failed parsing	
AP A430	DFHWUXML	Exc	XML within Action failed parsing	
AP A431	DFHWUXML	Exc	Action name was not specified	1 Tag attribute name
AP A432	DFHWUXML	Exc	Action parameters failed parsing	
AP A433	DFHWUXML	Exc	Action tag is empty	
AP A434	DFHWUXML	Exc	Action tag invalid End tag	
AP A435	DFHWUXML	Exc	Parameter failed parsing	
AP A436	DFHWUXML	Exc	Invalid compound parameters	
AP A437	DFHWUXML	Exc	Parameter value failed parsing	
AP A438	DFHWUXML	Exc	Parameter name was not specified correctly	1 Parameter name pointer 2 Parameter value pointer

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A439	DFHWUXML	Exc	Parameter value was not specified correctly	1 Parameter name pointer 2 Parameter value pointer
AP A43A	DFHWUXML	Exc	Building CREATE Attributes list failed	
AP A43B	DFHWUXML	Exc	Building UPDATE Attributes list failed	1 Return Code
AP A43D	DFHWUXML	Exc	Invalid REQUEST End tag	
AP A43E	DFHWUXML	WU 2	Parse tag Entry	
AP A43F	DFHWUXML	WU 2	Parse tag Exit	
AP A440	DFHWUXML	Exc	Tags must start with a <	
AP A441	DFHWUXML	Exc	Tag name failed verification	
AP A442	DFHWUXML	Exc	Tag end > was not found	
AP A443	DFHWUXML	Exc	Could not find > for a Start/Empty tag	
AP A444	DFHWUXML	WU 2	Compare Characters Entry	1 Input buffer 2 Compared character string
AP A445	DFHWUXML	WU 2	Compare Characters Exit	1 Match status
AP A446	DFHWUXML	WU 2	Normalise attributes Entry	1 Attributes 2 Attribute count
AP A447	DFHWUXML	WU 2	Normalise attributes Exit	
AP A44C	DFHWUXML	WU 2	Build Attribute list Entry	
AP A44D	DFHWUXML	WU 2	Build Attribute list Exit	
AP A44E	DFHWUXML	Exc	Exceed max attribute count	
AP A44F	DFHWUXML	Exc	Attribute name too long	
AP A450	DFHWUXML	WU 2	Parse Name value Entry	1 Input buffer 2 Index into Input buffer
AP A451	DFHWUXML	WU 2	Parse name value Exit	1 Attribute name 2 Attribute value
AP A452	DFHWUXML	Exc	Pair attribute name too long	

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A453	DFHWUXML	Exc	None Equate found between pair	
AP A454	DFHWUXML	Exc	Value is missing quotes	
AP A455	DFHWUXML	Exc	Value missing end quote	1 Index into Input buffer 2 Block length
AP A456	DFHWUXML	Exc	Unescape XML Entry	1 Attribute data
AP A457	DFHWUXML	Exc	Unescape XML Exit	1 Attribute data
AP A458	DFHWUXML	WU 2	Remove white space Entry	1 Input buffer 2 Index into Input buffer
AP A459	DFHWUXML	WU 2	Remove white space Exit	1 Input buffer 2 Index into Input buffer
AP A45A	DFHWUXML	Exc	Remove white space Error	
AP A45B	DFHWUXML	WU 2	Verify Data Entry	1 Start of the data to be verified 2 Maximum size of data expected
AP A45C	DFHWUXML	WU 2	Verify Data Exit	1 Calculated length of verified data 2 Maximum size of data expected
AP A45D	DFHWUXML	WU 2	Copy to attribute buffer Entry	1 Data to copy
AP A45E	DFHWUXML	WU 2	Copy to attribute buffer Exit	1 Data to copy
AP A45F	DFHWUXML	WU 2	Copy to parameter buffer Entry	1 Data to copy
AP A460	DFHWUXML	WU 2	Copy to parameter buffer Exit	1 Data to copy
AP A463	DFHWUXML	Exc	Attribute buffer not big enough	
AP A464	DFHWUXML	Exc	Parameter buffer not big enough	
AP A465	DFHWUXML	Exc	No parameter given to copy	
AP A466	DFHWUXML	Exc	No attribute given to copy	

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A467	DFHWUXML	Exc	White space expected between attributes	
AP A468	DFHWUXML	Exc	No attribute tag found in CREATE request	1 Attribute tags found 2 Parameter tags found
AP A469	DFHWUXML	Exc	No attribute tag found in UPDATE request	1 Attribute tags found 2 Parameter tags found
AP A46A	DFHWUXML	Exc	ACTION request should have no attribute tags	1 Attribute tags found 2 Parameter tags found
AP A46B	DFHWUXML	Exc	Invalid ACTION inner tags	
AP A46C	DFHWUXML	Exc	Extra data exists in the buffer after parsing	
AP A46D	DFHWUXML	WU 1	Exit	
AP A470	DFHWUSTG	WU 1	Entry	
AP A471	DFHWUSTG	WU 2	Entry	1 Operation Type 2 Storage pointer 3 Storage Length 4 Storage above the bar
AP A472	DFHWUSTG	Exc	Invalid Operation type	1 Operation Type 2 Storage pointer 3 Storage Length 4 Storage above the bar
AP A473	DFHWUSTG	WU 2	Release above the bar storage Entry	1 Storage above the bar 2 Storage Length

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A474	DFHWUSTG	WU 2	Release above the bar storage Exit	1 Storage Length 2 Return Code
AP A475	DFHWUSTG	WU 2	Copy storage above the bar Entry	1 Storage Pointer 2 Storage Length
AP A476	DFHWUSTG	WU 2	Copy storage above the bar Exit	1 Storage pointer 2 Storage Length 3 Storage above the bar 4 Return Code
AP A477	DFHWUSTG	WU 2	GETMAIN above the bar storage Entry	1 Storage Length
AP A478	DFHWUSTG	WU 2	GETMAIN above the bar storage Exit	1 Storage Length 2 Storage above the bar 3 Return Code
AP A479	DFHWUSTG	Exc	GETMAIN storage failed	1 Storage Length 2 Storage above the bar
AP A47A	DFHWUSTG	Exc	GETMAIN storage failed	1 Storage Length 2 Storage above the bar 3 Return Code 4 S2GF Response 5 S2GF Reason

Trace Entries

Table 52. CICI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A47B	DFHWUSTG	Exc	GETMAIN failed due to insufficient storage	1 Storage Length 2 Storage above the bar 3 Return Code 4 S2GF Response 5 S2GF Reason
AP A47C	DFHWUSTG	Exc	Above the bar storage RELEASE failed	1 Storage above the bar 2 Storage Length 3 Actual Storage length to be released 4 S2GF Response 5 S2GF Reason
AP A47D	DFHWUSTG	Exc	Copy above the bar failed	1 Storage pointer 2 Storage Length 3 Storage above the bar 4 S2SR Response 5 S2SR Reason
AP A47E	DFHWUSTG	Exc	Bad address for GETMAIN	1 Storage Length 2 Storage above the bar 3 Return Code
AP A47F	DFHWUSTG	WU 1	Exit	
AP A490	DFHWUGC1	WU 1	Entry	
AP A491	DFHWUGC1	WU 1	Exit	1 Return Code
AP A492	DFHWUGC1	WU 2	Entry main processing logic	

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A493	DFHWUGC1	WU 2	Exit main processing logic	1 Linked List 2 Linked list metadata block pointer 3 Number of entries deleted
AP A494	DFHWUGC1	Exc	Cache token not found	1 Cache Token
AP A495	DFHWUGC1	Exc	Failed to find Cache area	1 Trace point ID 2 Cache Token 3 DDLO Response 4 DDLO Reason
AP A496	DFHWUGC1	Exc	Invalid directory	1 Cache Token
AP A497	DFHWUGC1	Exc	Lock token not found	1 Linked List Lock Token
AP A498	DFHWUGC1	Exc	Acquire lock failed	1 Trace point ID 2 Linked List Token 3 LMLM Response 4 LMLM Reason
AP A499	DFHWUGC1	Exc	Release lock failed	1 Trace point ID 2 Linked List Token 3 LMLM Response 4 LMLM Reason
AP A4A0	DFHWUI2E	WU 1	Entry	
AP A4A1	DFHWUI2E	WU 2	Entry	
AP A4A2	DFHWUI2E	WU 2	Entry	

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A4A3	DFHWUI2E	WU 2	Before Image	1 Record attribute pointer 2 Attribute Value Length 3 Data Type
AP A4A4	DFHWUI2E	WU 2	After Image	1 Output buffer data 2 Length of buffer data
AP A4A5	DFHWUI2E	Exc	Bad Format	
AP A4A6	DFHWUI2E	WU 1	Exit	1 Return Code
AP A4A7	DFHWUI2E	Exc	Unknown CICS release	1 CICS Name 2 Release level data 3 Release level max
AP A4B0	DFHWUSCH	WU 1	Entry	
AP A4B1	DFHWUSCH	WU 2	Entry	1 IPG_REQUEST block 2 Model Name
AP A4B2	DFHWUSCH	WU 2	CPSM CONNECTION failed	1 IPG_REQUEST block 2 Return Code
AP A4B3	DFHWUSCH	WU 2	CPSM GET attribute failed	1 IPG_REQUEST block 2 Attribute data length 3 Attribute data count 4 ResultSet Token

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A4B3	DFHWUSCH	WU 2	CPSM GET attribute failed	1 API Function 2 API Response1 3 API Response2 4 Return Code
AP A4B4	DFHWUSCH	Exc	AVA GETDEF failed	1 Resource Name 2 API Thread 3 Result Token 4 API Response 5 API Reason
AP A4B5	DFHWUSCH	Exc	AVA FETCH failed	1 Attrava record 2 Index count 3 Result Token 4 API Thread 5 API Response 6 API Reason
AP A4B6	DFHWUSCH	Exc	AVA Discard failed	1 API Thread 2 Result Token 3 API Response 4 API Reason
AP A4B7	DFHWUSCH	WU 2	Render Resource Exit	1 IPG_REQUEST block 2 Return Code
AP A4B8	DFHWUSCH	WU 2	Render Main Schema Exit	1 Return Code

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A4B9	DFHWUSCH	WU 2	Render Error Schema Exit	1 Resource Name 2 Return Code
AP A4BA	DFHWUSCH	WU 2	Render Schema Attributes Install Error Exit	1 Return Code
AP A4BB	DFHWUSCH	WU 2	Render Feedback Scheme Exit	1 Return Code
AP A4BC	DFHWUSCH	WU 2	Render InconsistentSet Schema attribute Exit	1 Return Code
AP A4BD	DFHWUSCH	WU 2	Render InconsistentScope Schema attribute Exit	1 Return Code
AP A4BE	DFHWUSCH	WU 2	CPSM GET OBJECT failed	1 IPG_REQUEST block 2 CPSM Object ResultSet Token
AP A4BF	DFHWUSCH	WU 2	CPSM FETCH OBJECT failed	1 IPG_REQUEST block 2 CPSM Object ResultSet Token
AP A4C0	DFHWUSCH	WU 2	CPSM FETCH ATTRIBUTE failed	1 IPG_REQUEST block 2 CPSM Object ResultSet Token
AP A4C1	DFHWUSCH	Exc	CICS WEB EXTRACT failed	1 IPG_REQUEST block 2 RESP1 3 RESP2
AP A4C2	DFHWUSCH	WU 1	Exit	
AP A4D0	DFHWUSRT	WU 1	Entry	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER
AP A4D1	DFHWUSRT	WU 2	Entry 2	

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A4D2	DFHWUSRT	Exc	Acquire Enqueue failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 NQNQ request response code 4 NQNQ request reason code
AP A4D3	DFHWUSRT	Exc	Release Enqueue failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 NQNQ request response code 4 NQNQ request reason code
AP A4D4	DFHWUSRT	Exc	Lookup Enqueue Pool failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 DDLO request response code 4 DDLO request reason code
AP A4D5	DFHWUSRT	Exc	Invalid operation	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Operation type
AP A4D6	DFHWUSRT	Exc	Get Main Index failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Request buffer size
AP A4D7	DFHWUSRT	Exc	Free Main Index failed	1 IPG_REQUEST block 2 RSM_RESULTSET_HEADER 3 Buffer address 4 Buffer size

Trace Entries

Table 52. CMCI domain trace points (continued)

Point ID	Module	Lvl	Type	Data
AP A4DF	DFHWUSRT	WU 1	Exit	1 IPG_REQUEST block
				2 RSM_RESULTSET_HEADER
AP A4E0	DFHWUSRQ	WU 1	Entry	1 SRQ parameter block
				2 Low index
				3 High index
				4 Recursion depth
AP A4EF	DFHWUSRQ	WU 1	Exit	1 SRQ parameter block

Timer domain trace points

Table 53. Timer domain trace points

Point ID	Module	Lvl	Type	Data
TI 0200	DFHTIMF	TI 1	Entry	1 TIMF parameter list
TI 0201	DFHTIMF	TI 1	Exit	1 TIMF parameter list
TI 0250	DFHTIMF	Exc	Invalid domain call	1 TIMF parameter list
TI 0251	DFHTIMF	Exc	Invalid format number	1 TIMF parameter list
TI 0260	DFHTIMF	Exc	Recovery	1 TIMF parameter list
				2 Kernel error data

Miscellaneous Updates

With the exception of the *CICS TS Enhancements Guide*, no CICS TS publications are currently updated. This topic summarizes updates or clarifications to information that is contained in various CICS TS publications.

DFHCNV Macro

A conversion table can be defined with the DFHCNV resource definition macros. Although the DFHCNV TYPE=ENTRY macro allows to specify various resource types, RTYPE={FC|TS|TD|IC|PC}, RTYPE=PC is the only resource for which conversion takes place on CICS TS for z/VSE (and CICS TS for VSE/ESA).

Starting with CICS TS for z/VSE 2.1, the CLINTCP parameter, and SRVERCP parameter in the DFHCNV macro can be set to SYSDEF. The syntax of the DFHCNV macro is described in the publication [CICS Family: Communicating from CICS on System/390](#).

Resource and command check cross-reference

The "Resource command and check cross reference" section in the [CICS Security Guide](#) provides a complete API command and resource check cross reference. The behaviour of some commands did not match the specification.

The following commands were corrected with the CICS TS for z/VSE:

- All CREATE and DISCARD commands were changed and now require the specified ACCESS level, which is ALTER.
- ENABLE|DISABLE PROGRAM commands were changed and now require the specified ACCESS level, which is UPDATE.

These EXEC CICS commands are listed, but do not exist in CICS TS for z/VSE:

- WAIT JOURNALNAME
- DISCARD JOURNALNAME

CICS-supplied transactions

CEMT INQUIRE TASK **S**tartcode(*value*)

CEST INQUIRE TASK **S**tartcode(*value*)

Startcode(*value*)

displays how this task was started.

SZ

FEPI Start Command

REXX for CICS TS for z/VSE (REXX/CICS)

These are changes to the REXX/CICS product:

REXX/CICS SQL support

Back in 2007, the REXX/CICS SQL interface was withdrawn. Since then, REXX/CICS does not offer DB2® support. This was announced with the informational APAR PK33259. Therefore, the parts belonging to the CICS SQL package are no longer included in the CICS TS for z/VSE product. These are:

- CICS SQL.A
- CICS SQL.PHASE
- CICS SQL.OBJ
- CICS SQL.LK.OBJ

REXX/CICS documentation

Member CICR3270.Y that contained the REXX/CICS documentation is no longer shipped.

REXX/CICS samples

REXX/CICS samples that were shipped as type CIC*.Z are now shipped as type .PROC.

Notices

This information was developed for products and services offered in the US. 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 right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not 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
US*

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 provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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

This publication documents intended Programming Interfaces that allow the customer to write programs to obtain services of z/VSE.

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.

IPv6/VSE is a registered trademark of Barnard Software, Inc.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Terms and Conditions for Product Documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein. IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed. You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/VSE enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using Assistive Technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/VSE. Consult the assistive technology documentation for specific information when using such products to access z/VSE interfaces.

Documentation Format

The publications for this product are in Adobe Portable Document Format (PDF) and should be compliant with accessibility standards. If you experience difficulties when you use the PDF files and want to request a web-based format for a publication, you can either write an email to s390id@de.ibm.com, or use the Reader Comment Form in the back of this publication or direct your mail to the following address:

IBM Deutschland Research & Development GmbH
Department 3282
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

In the request, be sure to include the publication number and title.

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

Index

Special Characters

.END, PARM keyword [304](#)

A

abend codes

1590 [315](#)

AEIP [316](#)

AEYD [316](#)

ASRD [316](#)

accessibility [487](#)

ACTION option

WEB SEND command [95](#)

active delay interval for XRF [316](#)

activity keypoint frequency (AKPFREQ) [316](#)

ADI, system initialization parameter [316](#)

AIEXIT, system initialization parameter [316](#)

AILDELAY, system initialization parameter [316](#)

AIQMAX, system initialization parameter [316](#)

AIRDELAY, system initialization parameter [316](#)

AKPFREQ, system initialization parameter [316](#)

alias transaction CWBA [42](#)

ALL

CEMT INQUIRE TASK [174](#)

CEMT INQUIRE TRANSACTION [171](#)

CEMT INQUIRE TSQUEUE [171](#)

ANY

CEMT INQUIRE TRANSACTION [171](#)

AP trace points [376](#)

APPENDCRLF

CEMT INQUIRE DOCTEMPLATE [171](#)

AppendCrLf attribute

DOCTEMPLATE definition [151](#)

APPL statement, VTAM VBUILD application identifier [316](#)

APPLID, system initialization parameter [316](#)

ASSIGN (CHANNEL) command [191](#)

AT option

DOCUMENT INSERT command [60](#)

ATTACHSEC attribute

TCPIPSERVICE definition [161](#)

AUTCONN, system initialization parameter [316](#)

autoinstall

AIEXIT, system initialization parameter [316](#)

AILDELAY system initialization parameter [316](#)

AIQMAX, system initialization parameter [316](#)

PGAICTLG, system initialization parameter [316](#)

PGAIXIT, system initialization parameter [316](#)

PGAIPGM, system initialization parameter [316](#)

automatic start [312](#)

AUXILIARY

CEMT INQUIRE TSQUEUE [171](#)

AUXILIARY option

WRITEQ TS command [79](#)

auxiliary storage trace [316](#)

AUXTR, system initialization parameter [316](#)

AUXTRSW, system initialization parameter [316](#)

B

BACKLOG attribute

TCPIPSERVICE definition [162](#)

backout of resources at emergency restart [316](#)

batching requests [316](#)

BELOW

CEMT INQUIRE TRANSACTION [171](#)

bibliography xvii

big COMMAREAs [213](#), [216](#), [236](#)

BINARY option

DOCUMENT INSERT command [61](#)

BMS (basic mapping support)

BMS system initialization parameter [316](#)

DFHBMS, macro [316](#)

page-chaining command character string [316](#)

page-copying command character string [316](#)

page-purging command character string [316](#)

page-retrieval command character string [316](#)

PGCHAIN, BMS CHAIN command [316](#)

PGCOPY, BMS COPY command [316](#)

PGPURGE, BMS PURGE command [316](#)

PGRET, BMS RETRIEVAL command [316](#)

PRGDLAY, BMS PURGE DELAY command [316](#)

purge delay time interval [316](#)

selecting versions of BMS [309](#)

versions of BMS [316](#)

BMS, system initialization parameter [316](#)

BOOKMARK option

DOCUMENT INSERT command [61](#)

BRDATA option

START BREXIT command [72](#)

BRDATALENGTH option

START BREXIT command [72](#)

BREXIT

CEMT INQUIRE TRANSACTION [171](#)

BREXIT option

START BREXIT command [72](#)

BRIDGE

CEMT INQUIRE TASK [175](#)

bridge (3270)

start a task [71](#)

buffer size, DTB [316](#)

buffers and strings, VSAM [316](#)

C

CADDRLENGTH option

EXTRACT TCPIP command [67](#)

CDAKEY

CEMT INQUIRE TRANSACTION [171](#)

CDSASZE, system initialization parameter [316](#)

CEBR transaction [167](#)

CEDA DEFINE

DOCTEMPLATE [151](#)

PROFILE [155](#)

TCPIPSERVICE [161](#)

- CEDF transaction [169](#)
- CEMT INQUIRE DOCTEMPLATE
 - APPENDCRLF [171](#)
 - DOCTEMPLATE [171](#)
 - DSNAME [171](#)
 - NAME [171](#)
 - TEMPLATENAME [171](#)
 - TEMPLATETYPE [171](#)
 - TYPE [171](#)
- CEMT INQUIRE DOCTEMPLATE definition
 - LIBRARY [171](#)
- CEMT INQUIRE/SET TCPIP SERVICE [171](#)
- CEMT transaction
 - DOCTEMPLATE [171](#)
 - TASK [173](#)
 - TCPIP [171](#)
 - TCPIP SERVICE [171](#)
 - TRANSACTION [171](#)
 - TSQUEUE [171](#)
- CERTIFICATE attribute
 - TCPIP SERVICE definition [162](#)
- Certificate Authority (CA)
 - description [127](#)
- CERTIFICATE option
 - EXTRACT CERTIFICATE command [69](#)
- CHAINCONTROL attribute
 - PROFILE definition [155](#)
- channels
 - API commands for using [191](#)
 - as large COMMAREAs [213](#)
 - basic examples [214](#)
 - benefits of [236](#)
 - CICS translator [223](#)
 - constructing [230](#)
 - creating [218](#)
 - current [219](#), [223](#)
 - current, example, with LINK commands [220](#)
 - current, example, with XCTL commands [222](#)
 - data conversion [232](#)
 - designing [229](#)
 - discovering which containers were passed to a program [228](#)
 - discovering which containers were returned from a link [228](#)
 - dynamic transaction routing [231](#)
 - migrating
 - LINK command [237](#)
 - RETURN command [238](#)
 - XCTL command [237](#)
 - overview [189](#), [213](#)
 - passing a channel to another program [227](#)
 - processing containers in a sub-routine [227](#)
 - read only containers [228](#)
 - scope of [224](#), [226](#)
 - typical scenarios
 - multiple interactive components [218](#)
 - one channel—one program [216](#)
 - one channel—several programs [217](#)
 - several channels, one component [217](#)
- channels as large COMMAREAs [213](#), [216](#), [236](#)
- CHARACTERSET option
 - WEB RECEIVE command [91](#)
 - WEB SEND command [96](#)
- CHKSTRM, system initialization parameter [316](#)
- CHKSTSK, system initialization parameter [316](#)
- CHUNKING option
 - WEB SEND command [96](#)
- CI (control interval)
 - for auxiliary temporary storage data set [316](#)
 - for intrapartition data set [316](#)
- CICS
 - ECI over TCP/IP [137](#)
- CICS Explorer
 - configuring a connection to a CICS system [24](#)
 - configuring host [17](#)
 - configuring new connection to CICS system [23](#)
 - connecting to a CICS system [24](#)
 - defining connection credentials [23](#)
 - downloading client-part [20](#)
 - Definitions views that are supported by z/VSE [9](#)
 - example of using views [6](#)
 - HTTP GET request for information on resources [12](#)
 - HTTP PUT request to process a resource [13](#)
 - messages [35](#)
 - Operations views that are supported by z/VSE [7](#)
 - overview [3](#)
 - perspectives [3](#)
 - restrictions when connecting to CICS TS for z/VSE system [15](#)
 - Rich Client Platform (RCP) [3](#)
 - using existing connection to a CICS system [27](#)
 - using HTTP requests to process resource definitions [14](#)
 - Workbench [3](#)
- CICS Explorer client-part
 - downloading from CICS Explorer Web page [20](#)
 - installing [20](#)
- CICS Explorer traces
 - Server part [32](#)
- CICS Explorer user workspace
 - changing [28](#)
- CICS Explorer (diagnosing problems) [31](#)
- CICS Management Interface resources, supported by when connecting to a CICS TS for z/VSE system [11](#)
- CICS spooler interface
 - XPCC, cross-partition communication component [316](#)
- CICS system initialization parameters
 - TCPIP [39](#)
- CICS Web interface
 - WEBDELAY system initialization parameter [316](#)
- CICS Web interface (CWI) commands
 - WEB RECEIVE [91](#)
 - WEB RETRIEVE [94](#)
- CICS Web Interface (CWI) commands
 - DOCUMENT CREATE [57](#)
 - DOCUMENT INSERT [60](#)
 - DOCUMENT RETRIEVE [62](#)
 - DOCUMENT SET [63](#)
 - EXTRACT WEB [84](#)
 - WEB ENDBROWSE FORMFIELD [83](#)
 - WEB ENDBROWSE HTTPHEADER [83](#)
 - WEB EXTRACT [84](#)
 - WEB READ FORMFIELD [86](#)
 - WEB READ HTTPHEADER [88](#)
 - WEB READNEXT FORMFIELD [89](#)
 - WEB READNEXT HTTPHEADER [90](#)
 - WEB SEND [95](#)
 - WEB STARTBROWSE FORMFIELD [100](#)
 - WEB STARTBROWSE HTTPHEADER [101](#)

CICS Web Interface (CWI) commands *(continued)*
 WEB WRITE [102](#)
 CIEP, used by ECI via TCP/IP [164](#)
 CLASS
 CEMT INQUIRE TRANSACTION [171](#)
 class, monitoring
 monitoring [316](#)
 client authentication
 configuring a CICS client for [132](#)
 mapping client certificates to VSE User IDs [132](#)
 overview of use [127](#)
 client certificate
 overview of use [127](#)
 client keyring file, supplied by IBM [131](#)
 client requests
 extracting information [84](#)
 Client virtual terminals
 system initialization parameter, VTPREFIX [316](#)
 CLIENTADDR option
 EXTRACT TCPIP command [67](#)
 CLIENTADDRNU option
 EXTRACT TCPIP command [67](#)
 CLIENTNAME option
 EXTRACT TCPIP command [67](#)
 CLNTCODEPAGE option
 DOCUMENT RETRIEVE command [63](#)
 WEB READ FORMFIELD command [86](#)
 WEB RECEIVE command [92](#)
 WEB SEND command [96](#)
 WEB STARTBROWSE FORMFIELD command [101](#)
 CLOSESTATUS option
 WEB SEND command [97](#)
 CLSDSTP, system initialization parameter [316](#)
 CLT (command list table)
 CLT system initialization parameter [316](#)
 CMDPROT, system initialization parameter [316](#)
 CMDSEC, system initialization parameter [316](#)
 CNAMELENGTH option
 EXTRACT TCPIP command [67](#)
 COD0, CODB debugging transactions [31](#)
 code page conversion
 preparing [232](#)
 COLD option
 system initialization parameter BMS [316](#)
 system initialization parameter DCT [316](#)
 system initialization parameter DLI [316](#)
 system initialization parameter START [316](#)
 system initialization parameter TS [316](#)
 commands
 CEMT INQUIRE/SET
[171](#)
 COMMAREA
 migrating to channels and containers
 LINK command [237](#)
 RETURN command [238](#)
 XCTL command [237](#)
 COMMAREAs > 32K [213](#), [236](#)
 common work area storage key
 system initialization parameter [316](#)
 COMMONNAME option
 EXTRACT CERTIFICATE command [69](#)
 COMMONNAMLEN option
 EXTRACT CERTIFICATE command [69](#)
 components
 components *(continued)*
 multiple, interactive [218](#)
 one channel—several programs [217](#)
 several channels, one component [217](#)
 CONFDATA, system initialization parameter [316](#)
 CONFTEXT, system initialization parameter [316](#)
 CONSOLE (CN), PARM keyword [304](#)
 consoles
 entering system initialization parameters [307](#)
 constructing a channel [230](#)
 container commands
 ASSIGN (CHANNEL) [191](#)
 DELETE CONTAINER (CHANNEL) [192](#)
 ENDBROWSE CONTAINER [192](#)
 GET CONTAINER (CHANNEL) [193](#)
 GETNEXT CONTAINER [197](#)
 LINK [198](#)
 MOVE CONTAINER (CHANNEL) [199](#)
 PUT CONTAINER (CHANNEL) [201](#)
 RETURN [204](#)
 START CHANNEL [206](#)
 STARTBROWSE CONTAINER [209](#)
 XCTL [210](#)
 containers
 API commands for using [191](#)
 basic examples [214](#)
 designing a channel [229](#)
 discovering which containers were passed to a program
[228](#)
 discovering which were returned from a link [228](#)
 migrating
 LINK command [237](#)
 RETURN command [238](#)
 XCTL command [237](#)
 overview [189](#), [213](#)
 passing a channel to another program [227](#)
 processing containers in a sub-routine [227](#)
 read only [228](#)
 Control block
 internal [265](#)
 Control blocks [253](#)
 Control tables
 reassembling [259](#)
 CONVERTTIME command [283](#)
 COUNTRY option
 EXTRACT CERTIFICATE command [69](#)
 COUNTRYLEN option
 EXTRACT CERTIFICATE command [69](#)
 creating a channel [218](#)
 cross-partition communication component, XPCC [316](#)
 CSD [261](#)
 CSD (CICS system definition data set)
 CSDACC, system initialization parameter [316](#)
 CSDBUFND, system initialization parameter [316](#)
 CSDBUFNI, system initialization parameter [316](#)
 CSDFRLOG, system initialization parameter [316](#)
 CSDJID, system initialization parameter [316](#)
 CSDLSRNO, system initialization parameter [316](#)
 CSDRECOV, system initialization parameter [316](#)
 CSDSTRNO, system initialization parameter [316](#)
 CSDACC, system initialization parameter [316](#)
 CSDBUFND, system initialization parameter [316](#)
 CSDBUFNI, system initialization parameter [316](#)
 CSDFRLOG, system initialization parameter [316](#)

CSDJID, system initialization parameter [316](#)
 CSDLRNO, system initialization parameter [316](#)
 CSDRECOV, system initialization parameter [316](#)
 CSDSTRNO, system initialization parameter [316](#)
 CSECT operand of system initialization parameter TYPE [316](#)
 current channel
 example, with LINK commands [220](#)
 example, with XCTL commands [222](#)
 overview [219](#), [223](#)
 CWA (common work area)
 WRKAREA, system initialization parameter [316](#)
 CWAKEY
 system initialization parameter [316](#)
 CWBA alias transaction [42](#)
 CWXN Web attach transaction [42](#)

D

D
 CEMT INQUIRE TASK [177](#)
 data conversion
 and channels
 file example [235](#)
 why necessary [231](#)
 with channels [232](#)
 data sets
 dump [316](#)
 journal data sets [316](#)
 SYSIPT data set [306](#)
 data, deleting
 temporary storage queues [75](#)
 DATAONLY option
 DOCUMENT RETRIEVE command [63](#)
 date format [316](#)
 DATFORM, system initialization parameter [316](#)
 DBP (dynamic backout program)
 DBP, system initialization parameter [316](#)
 using suffixes [309](#)
 DBP, system initialization parameter [316](#)
 DBUFSZ, system initialization parameter [316](#)
 DCT (destination control table)
 DCT, system initialization parameter [316](#)
 specifying the DCT suffix [316](#)
 DCT, system initialization parameter [316](#)
 DDNAME
 CEMT INQUIRE DOCTEMPLATE [171](#)
 DDS option of system initialization parameter BMS [316](#)
 Debugging transactions
 COD0, CODB [31](#)
 delay intervals
 active delay for XRF [316](#)
 reconnection for XRF [316](#)
 delay, persistent verification [316](#)
 DELETE CONTAINER (CHANNEL) command [192](#)
 DELETEQ TS command [75](#)
 deleting data
 temporary storage queues [75](#)
 DESCRIPTION attribute
 DOCTEMPLATE definition [151](#)
 PROFILE definition [155](#)
 TCPIPSERVICE definition [162](#)
 designing a channel [229](#)
 DEST
 CEMT INQUIRE TASK [176](#)

DFH\$WB1A [46](#), [47](#)
 DFH\$WBSN [46](#)
 DFH\$WBSN RDO group [40](#)
 DFHOWBCA sample application [132](#)
 DFHCMT [274](#)
 DFHCNV [253](#), [269](#)
 DFHCNV table [43](#)
 DFHDHTXD [41](#)
 DFHDHTXH [41](#)
 DFHDHTXL [41](#)
 DFHDHTXO [41](#)
 DFHDU420 [279](#)
 DFHDU430 [279](#)
 DFHDYP, dynamic transaction routing program
 coding the DTRPGM system initialization parameter [316](#)
 DFHDYDPS [236](#)
 DFHEIENT macro [243](#)
 DFHEIRET macro [244](#)
 DFHTML DD name [41](#), [43](#)
 DFHMCT [274](#)
 DFHPD430 [279](#)
 DFHSIT [39](#)
 DFHSIT keywords and operands
 DFHSIT keywords and operands [294](#)
 DFHSIT, system initialization macro
 .END, PARM keyword [304](#)
 assembling the SIT [301](#)
 coding the PARM parameter over two lines [305](#)
 CONSOLE (CN), PARM keyword [304](#)
 creating a SIT [300](#)
 creating more than one SIT [300](#)
 defining your SIT to CICS at startup [302](#)
 DFHSIT TYPE=DSECT [316](#)
 parameters that cannot be coded [301](#)
 SITs supplied with z/VSE [301](#)
 supplied SITs [300](#)
 SYSIN (SI), PARM keyword [303](#)
 DFHTCTDY, dummy TCT [310](#)
 DFHTU420 [279](#)
 DFHTU430 [279](#)
 DFHUCNV [253](#), [269](#)
 DFHWBA alias program [42](#)
 DFHWBENV (environment variables program) [46](#)
 DFHWBHH conversion template [43](#)
 DFHWBHH conversion template name [43](#)
 DFHWBUD conversion template [43](#)
 DFHWBUD conversion template name [43](#)
 DFHWEB RDO group [40](#)
 DFLTUSER, system initialization parameter [316](#)
 DHFSIT [253](#), [257](#)
 Diagnosing problems in CICS Explorer [31](#)
 DIP, system initialization parameter [316](#)
 disability [487](#)
 DISABLED
 CEMT INQUIRE TRANSACTION [171](#)
 discovering which containers were passed to a program [228](#)
 discovering which containers were returned from a link [228](#)
 DISMACP, system initialization parameter [316](#)
 DISPATCHABLE
 CEMT INQUIRE TASK [176](#)
 DL/I
 DLI (DL1), system initialization parameter [316](#)
 DLIOER, system initialization parameter [316](#)
 XDBDERR, global user exit [316](#)

DLI (DL1), system initialization parameter [316](#)
 DLIOER, system initialization parameter [316](#)
 DNS1 [46](#)
 DOCCODEPAGE, system initialization parameter [316](#)
 DOCSIZE option
 DOCUMENT INSERT command [61](#)
 DOCTEMPLATE
 CEMT INQUIRE DOCTEMPLATE [171](#)
 CEMT transaction [171](#)
 DOCTEMPLATE attribute
 DOCTEMPLATE definition [151](#)
 DOCTEMPLATE definition
 AppendCrLf attribute [151](#)
 DESCRIPTION attribute [151](#)
 DOCTEMPLATE attribute [151](#)
 DOCTEMPLATE definition [151](#)
 EXITPGM attribute [151](#)
 FILENAME attribute [151](#)
 GROUP attribute [151](#)
 Library attribute [151](#)
 MEMBERNAME attribute [151](#)
 PROGRAMNAME attribute [151](#)
 RDO command (CEDA DEFINE) [151](#)
 TDQUEUE attribute [151](#)
 TEMPLATENAME attribute [151](#)
 TSQUEUE attribute [151](#)
 TYPE attribute [151](#)
 DOCTEMPLATE resource [142](#)
 DOCTOKEN option
 DOCUMENT RETRIEVE command [63](#)
 DOCUMENT SET command [64](#)
 WEB RETRIEVE command [94](#)
 WEB SEND command [97](#)
 document
 adding symbols to symbol table [63](#)
 creating [57](#)
 sending documents [95](#)
 DOCUMENT CREATE command [57](#)
 DOCUMENT INSERT command [60](#)
 DOCUMENT option
 DOCUMENT INSERT command [61](#)
 DOCUMENT RETRIEVE command [62](#)
 DOCUMENT SET command [63](#)
 documents
 creating [141](#)
 DOCTEMPLATE [142](#)
 HTML [139](#)
 domains
 document domain [139](#)
 double-byte character set (DBCS) [43](#)
 DS
 CEMT INQUIRE TASK [177](#)
 DSA (dynamic storage area)
 EDSALIM, system initialization parameter [316](#)
 RENTPGM, system initialization parameter [316](#)
 STGPROT, system initialization parameter [316](#)
 DSALIM, system initialization parameter [316](#)
 DSECT operand of system initialization parameter TYPE [316](#)
 DSHIPIDL, system initialization parameter [316](#)
 DSHIPINT, system initialization parameter [316](#)
 DSNAME
 CEMT INQUIRE DOCTEMPLATE [171](#)
 DTB (dynamic transaction backout)
 dynamic buffer size [316](#)

DTB (dynamic transaction backout) (*continued*)
 indicating the program version [316](#)
 DTRPGM, system initialization parameter [316](#)
 DTRTRAN, system initialization parameter [316](#)
 dump data sets [316](#)
 Dump exit routine DFHPD420 [279](#)
 dump facilities
 DUMP, system initialization parameter [316](#)
 DUMPDS, system initialization parameter [316](#)
 DUMPSW, system initialization parameter [316](#)
 effect of START= parameter [312](#)
 SYDUMAX, system initialization parameter [316](#)
 TRDUMAX, system initialization parameter [316](#)
 DUMP, system initialization parameter [316](#)
 DUMPDS, system initialization parameter [316](#)
 DUMPSW, system initialization parameter [316](#)
 DVSUPRT attribute
 PROFILE definition [156](#)
 dynamic buffer size [316](#)
 Dynamic routing programs [271](#)
 dynamic transaction routing
 DTRPGM, system initialization parameter [316](#)
 DTRTRAN, system initialization parameter [316](#)
 dynamic transaction routing program with channels [231](#)
 Dynamic transaction routing programs [271](#)
 Dynamic Transaction Routing programs [253](#)

E

ECDSASZE, system initialization parameter [316](#)
 ECI (external call interface)
 configuring CICS for [137](#)
 defining TCPIP SERVICE for [137](#)
 displaying ATTACHSEC attribute [138](#)
 EDF (execution diagnostic facility) [169](#)
 EDSALIM, system initialization parameter [316](#)
 emergency restart
 resource backout [316](#)
 START system initialization parameter [316](#)
 ENABLED
 CEMT INQUIRE TRANSACTION [171](#)
 ENCRYPTION, system initialization parameter [316](#)
 ENDBROWSE CONTAINER command [192](#)
 ENDFILE condition
 WEB READNEXT FORMFIELD command [89](#)
 WEB READNEXT HTTPHEADER command [90](#)
 environment variables program (DFHWBENV) [46](#)
 EODI, system initialization parameter [316](#)
 ERDSASZE, system initialization parameter [316](#)
 ESASASZE, system initialization parameter [316](#)
 ESMEXITS, system initialization parameter [316](#)
 EUDSASZE, system initialization parameter [316](#)
 examples
 channels, basic [214](#)
 CICS client program that constructs a channel [230](#)
 CICS server program that uses a channel [231](#)
 containers, basic [214](#)
 multiple interactive components [218](#)
 one channel—one program [216](#)
 one channel—several programs [217](#)
 several channels, one component [217](#)
 using the READQ TS command [78](#)
 using the START BREXIT command [72](#)
 exception class monitoring [316](#)

EXEC CICS application programming interface [283](#)
EXEC interface block [211](#)
execution diagnostic facility (EDF) [169](#)
exit interval, region [316](#)
EXITPGM attribute
 DOCTEMPLATE definition [151](#)
exits
 FE, global trap exit [316](#)
 XDBDERR, global user exit [316](#)
external security interface [316](#)
EXTRACT CERTIFICATE command [68](#), [132](#)
EXTRACT TCPIP command [67](#)
EXTRACT WEB command [84](#)
EYUPARM job
 for entering/changing debugging commands/
 parameters [29](#)

F

facilities
 auxiliary trace autoswitch facility [316](#)
FACILITY
 CEMT INQUIRE TASK [175](#)
FACILITYLIKE
 CEMT INQUIRE TRANSACTION [171](#)
FACILITYLIKE attribute
 PROFILE definition [156](#)
FCT (file control table)
 specifying the FCT suffix [316](#)
FCT, system initialization parameter [316](#)
FE global trap exit [316](#)
field
 extracting information [86](#)
field name start character [316](#)
field separator characters [316](#)
FLDSEP, system initialization parameter [316](#)
FLDSTRT, system initialization parameter [316](#)
FORCEPURGE
 CEMT INQUIRE TASK [176](#)
form field
 extracting information [86](#)
FORMATTIME command [285](#)
FORMFIELD option
 WEB READ FORMFIELD command [87](#)
 WEB READNEXT FORMFIELD command [89](#)
frequency, activity keypoint [316](#)
FROM option
 WEB SEND command [97](#)
 WRITEQ TS command [79](#)
FROMDOC option
 DOCUMENT INSERT command [61](#)
FROMLENGTH option
 WEB SEND command [97](#)
front end programming interface (FEPI)
 FEPI, system initialization parameter [316](#)
FSSTAFF, system initialization parameter [316](#)
FTYPE
 CEMT INQUIRE TASK [175](#)
FULL option of system initialization parameter BMS [316](#)
function shipping
 FSSTAFF, system initialization parameter [316](#)

G

generic applid for CICS XRF systems [316](#)
GET CONTAINER (CHANNEL) command [193](#)
GETNEXT CONTAINER command [197](#)
global catalog data set (GCD)
 role in system initialization process [310](#)
global trap exit, FE [316](#)
Global user exit programming interface [267](#)
Global user exit programs [253](#)
GMTEXT, system initialization parameter [316](#)
GMTRAN, system initialization parameter [316](#)
GNTRAN, system initialization parameter [316](#)
good morning message [316](#)
good morning transaction [316](#)
GOOD MORNING transaction [316](#)
GROUP attribute
 PROFILE definition [156](#)
 TCPIP SERVICE definition [162](#)
group list, RDO [316](#)
GRPLIST, system initialization parameter [316](#)

H

header
 browsing [83](#), [100](#), [101](#)
 extracting information [88](#)
 retrieve next [89](#), [90](#)
homepage
 for downloading CICS Explorer client-part [20](#)
HOST option
 WEB EXTRACT command [84](#)
HOSTCODEPAGE option
 WEB READ FORMFIELD command [87](#)
 WEB RECEIVE command [92](#)
 WEB SEND command [97](#)
 WEB STARTBROWSE FORMFIELD command [101](#)
HOSTLENGTH option
 WEB EXTRACT command [84](#)
HTIME
 CEMT INQUIRE TASK [176](#)
HTML template manager
 setting up a sublibrary [43](#)
HTTP GET request, for information on resources [12](#)
HTTP PUT request, to process a resource [13](#)
HTTP requests
 receiving [91](#), [94](#)
HTTP requests to process resource definitions [14](#)
HTTP/1.1
 CICS Web support behavior in compliance with
 HTTP/1.1 [50](#)
 compliance for CICS as an HTTP server [49](#)
httpheader
 extracting information [102](#)
HTTPHEADER option
 WEB READ HTTPHEADER command [88](#)
 WEB READNEXT HTTPHEADER command [90](#)
 WEB WRITE command [102](#)
HTTPMETHOD option
 WEB EXTRACT command [84](#)
HTTPS [163](#)
HTTPVERSION option
 WEB EXTRACT command [84](#)
HTYPE

HTYPE (*continued*)
CEMT INQUIRE TASK [176](#)
HVALUE
CEMT INQUIRE TASK [176](#)

I

ICP, system initialization parameter [316](#)
ICV, system initialization parameter [316](#)
ICVR, system initialization parameter [316](#)
ICVTSD, system initialization parameter [316](#)
IDENTIFIER
CEMT INQUIRE TASK [176](#)
ILLOGIC condition
WEB EXTRACT command [85](#)
INBFMH attribute
PROFILE definition [156](#)
INITPARM, system initialization parameter [316](#)
internal trace, main storage [316](#)
intervals, activity keypoint [316](#)
INTO option
DOCUMENT RETRIEVE command [63](#)
READQ TS command [76](#)
WEB RECEIVE command [92](#)
INTR, system initialization parameter [316](#)
INVREQ condition
DELETEQ TS command [75](#)
EXTRACT CERTIFICATE command [70](#)
EXTRACT TCPIP command [68](#)
READQ TS command [77](#)
START BREXIT command [72](#)
WEB ENDBROWSE FORMFIELD command [83](#)
WEB ENDBROWSE HTTPHEADER command [83](#)
WEB EXTRACT command [86](#)
WEB READ FORMFIELD command [87](#)
WEB READ HTTPHEADER command [88](#)
WEB READNEXT FORMFIELD command [89](#)
WEB READNEXT HTTPHEADER command [90](#)
WEB RECEIVE command [93](#)
WEB RETRIEVE command [94](#)
WEB SEND command [98](#)
WEB STARTBROWSE FORMFIELD command [101](#)
WEB STARTBROWSE HTTPHEADER command [101](#)
WEB WRITE command [102](#)
WRITEQ TS command [80](#)
IOERR condition
READQ TS command [77](#)
WRITEQ TS command [81](#)
IOERR condition WEB SEND command [100](#)
IPADDRESS attribute
TCPIPSERVICE definition [162](#)
IRC (interregion communication)
IRCSTRT, system initialization parameter [316](#)
IRCSTRT, system initialization parameter [316](#)
ISC (intersystem communication)
ISC, system initialization parameter [316](#)
ISC, system initialization parameter [316](#)
ISCINVREQ condition
DELETEQ TS command [75](#)
READQ TS command [77](#)
WRITEQ TS command [81](#)
ISO 8859-1 character set [43](#)
ISSUER option
EXTRACT CERTIFICATE command [70](#)

ITEM option
READQ TS command [76](#)
WRITEQ TS command [79](#)
ITEMERR condition
READQ TS command [78](#)
WRITEQ TS command [81](#)

J

JCT, system initialization parameter [316](#)
journal archiving, automatic
CSDJID, system initialization parameter [316](#)
JOURNAL attribute
PROFILE definition [156](#)
journal control table (JCT)
JCT, system initialization parameter [316](#)
specifying the JCT suffix [316](#)
journal data sets
JSTATUS, system initialization parameter [316](#)
journaling
JCT, system initialization parameter [316](#)
specifying security checking for journal entries [316](#)
XJCT, system initialization parameter [316](#)
JSTATUS, system initialization parameter [316](#)

K

key ring, used by CICS [162](#)
KEYFILE, system initialization parameter [316](#)
Keyman/VSE tool [131](#)
keypoint frequency [316](#)
keys for page-retrieval [316](#)

L

large COMMAREAs [213](#), [216](#), [236](#)
Latin-1 character set [43](#)
LENGERR condition
EXTRACT CERTIFICATE command [70](#)
EXTRACT TCPIP command [68](#)
READQ TS command [78](#)
START BREXIT command [72](#)
WEB EXTRACT command [86](#)
WEB READ FORMFIELD command [87](#)
WEB READ HTTPHEADER command [88](#)
WEB READNEXT command [90](#)
WEB READNEXT FORMFIELD command [89](#)
WEB RECEIVE command [94](#)
WEB WRITE command [103](#)
WRITEQ TS command [81](#)
LENGERR option
DOCUMENT RETRIEVE command [63](#)
LENGTH
CEMT INQUIRE TSQUEUE [171](#)
LENGTH option
DOCUMENT RETRIEVE command [63](#)
DOCUMENT SET command [64](#)
EXTRACT CERTIFICATE command [70](#)
READQ TS command [77](#)
WEB RECEIVE command [92](#)
WEB SEND command [97](#)
WRITEQ TS command [79](#)
LEVSE (LE support)

LEVSE (LE support) *(continued)*
 LEVSE, system initialization parameter [316](#)
 LEVSE, system initialization parameter [316](#)
 LGNMSG, system initialization parameter [316](#)
 Library attribute
 DOCTEMPLATE definition [151](#)
 LINK
 migrating to channels and containers [237](#)
 LINK command [198](#)
 load modules [41](#)
 local catalog data set (LCD)
 role in system initialization process [310](#)
 use in restart [311](#)
 LOCALITY option
 EXTRACT CERTIFICATE command [70](#)
 LOCALITYLEN option
 EXTRACT CERTIFICATE command [70](#)
 LOCATION
 CEMT INQUIRE TSQUEUE [171](#)
 LOCKED condition
 DELETEQ TS command [76](#)
 logon data, VTAM [316](#)
 LOGREC attribute
 PROFILE definition [157](#)

M

macros
 DFHBMS, BMS macro [316](#)
 MAIN
 CEMT INQUIRE TSQUEUE [171](#)
 MAIN option
 WRITEQ TS command [80](#)
 MAXITEMLEN
 CEMT INQUIRE TSQUEUE [171](#)
 MAXLENGTH option
 DOCUMENT RETRIEVE command [63](#)
 WEB RECEIVE command [92](#)
 MCT [273](#)
 MCT, system initialization parameter [316](#)
 MEDIATYPE option
 WEB SEND command [97](#)
 MEMBERNAME attribute
 DOCTEMPLATE definition [151](#)
 message case [316](#)
 message level [316](#)
 message set, temporary storage [316](#)
 message, good morning [316](#)
 messages
 generated when using CICS Explorer [35](#)
 MSGCASE, system initialization parameter [316](#)
 MSGLVL, system initialization parameter [316](#)
 METHODLENGTH option
 WEB EXTRACT command [85](#)
 migrate TCIPSERVICE definitions [263](#)
 migrating START data to use channels [238](#)
 migration
 of START data to use channels [238](#)
 Migration [253](#)
 MINIMUM option of system initialization parameter BMS [316](#)
 MINITEMLEN
 CEMT INQUIRE TSQUEUE [171](#)
 MN, system initialization parameter [316](#)
 MNCONV, system initialization parameter [316](#)

MNEXC, system initialization parameter [316](#)
 MNFREQ, system initialization parameter [316](#)
 MNPER, system initialization parameter [316](#)
 MNSYNC, system initialization parameter [316](#)
 MNTIME, system initialization parameter [316](#)
 MODENAME attribute
 PROFILE definition [157](#)
 modernising COMMAREAs [213](#)
 Monitoring [253](#), [273](#)
 Monitoring control table, MCT [273](#)
 monitoring facilities
 effect of START= parameter [312](#)
 exception class [316](#)
 MCT, system initialization parameter [316](#)
 MN, system initialization parameter [316](#)
 MNCONV, system initialization parameter [316](#)
 MNEXC, system initialization parameter [316](#)
 MNFREQ, system initialization parameter [316](#)
 MNPER, system initialization parameter [316](#)
 MNSYNC, system initialization parameter [316](#)
 MNTIME, system initialization parameter [316](#)
 performance class [316](#)
 Monitoring utility program [275](#)
 MOVE CONTAINER (CHANNEL) command [199](#)
 MRO (multiregion operation)
 batching [316](#)
 extend lifetime of long-running mirror [316](#)
 long-running mirror [316](#)
 MROBTCH, system initialization parameter [316](#)
 MROFSE, system initialization parameter [316](#)
 MROLRM, system initialization parameter [316](#)
 MROBTCH, system initialization parameter [316](#)
 MROFSE, system initialization parameter [316](#)
 MROLRM, system initialization parameter [316](#)
 MSGCASE, system initialization parameter [316](#)
 MSGINTEG attribute
 PROFILE definition [157](#)
 MSGJRNL attribute
 PROFILE definition [157](#)
 MSGLVL, system initialization parameter [316](#)
 MXT, system initialization parameter [316](#)

N

NAME
 CEMT INQUIRE DOCTEMPLATE [171](#)
 name server [46](#)
 NAMELENGTH option
 WEB READ FORMFIELD command [87](#)
 WEB READ HTTPHEADER command [88](#)
 WEB READNEXT FORMFIELD command [89](#)
 WEB READNEXT HTTPHEADER command [90](#)
 WEB WRITE command [102](#)
 NATLANG, system initialization parameter [316](#)
 NEPCLASS attribute
 PROFILE definition [157](#)
 NEWSIT, system initialization parameter
 effect on warm start [312](#)
 NEXT option
 READQ TS command [77](#)
 NODDS option of system initialization parameter BMS [316](#)
 NOSPACE condition
 WRITEQ TS command [81](#)
 NOSUSPEND option

NOSUSPEND option (*continued*)
 WRITEQ TS command [80](#)

NOTAUTH condition
 DELETEQ TS command [76](#)
 READQ TS command [78](#)
 START BREXIT command [72](#)
 WRITEQ TS command [81](#)

NOTFND condition
 WEB READ FORMFIELD command [88](#)
 WEB READ HTTPHEADER command [88](#)
 WEB RECEIVE command [94](#)

NOTFND condition WEB SEND command [100](#)

NOTFND option
 DOCUMENT RETRIEVE command [63](#)

NOTPURGEABLE
 CEMT INQUIRE TRANSACTION [171](#)

NOTTRUNCATE option
 WEB RECEIVE command [92](#)

NUMITEMS
 CEMT INQUIRE TSQUEUE [171](#)

NUMITEMS option
 READQ TS command [77](#)
 WRITEQ TS command [80](#)

O

OFF
 CEDX [169](#)

ON
 CEDX [169](#)

ONEWTE attribute
 PROFILE definition [158](#)

OpenSSL
 setting up z/VSE system for [131](#)

operator communication for initialization parameters [307](#)

OPERTIM, system initialization parameter [316](#)

ORGANIZATION option
 EXTRACT CERTIFICATE command [70](#)

ORGANIZATLEN option
 EXTRACT CERTIFICATE command [70](#)

ORGUNIT option
 EXTRACT CERTIFICATE command [70](#)

ORGUNITLEN option
 EXTRACT CERTIFICATE command [70](#)

overriding system initialization parameters
 from the console [307](#)
 from the SYSIPT data set [306](#)

overview
 dynamic transaction routing program with channels [231](#)

OWNER option
 EXTRACT CERTIFICATE command [70](#)

P

PA keys for page-retrieval [316](#)
 PA keys for screen copying [316](#)
 page-chaining command character string [316](#)
 page-copying command character string [316](#)
 page-purging command character string [316](#)
 page-retrieval command character string [316](#)
 page-retrieval keys [316](#)
 PARMERR, system initialization parameter [316](#)
 passing a channel to another program [227](#)

PATH option
 WEB EXTRACT command [85](#)

PATHLENGTH option
 WEB EXTRACT command [85](#)

PDI, system initialization parameter [316](#)

performance class monitoring [316](#)

Performance class monitoring [273](#)

persistent sessions support
 PVDELAY, system initialization parameter [316](#)

persistent verification delay [316](#)

perspectives, Eclipse [3](#)

PF keys for page-retrieval [316](#)

PGCHAIN, system initialization parameter [316](#)

PGCOPY, system initialization parameter [316](#)

PGMIDERR condition
 START BREXIT command [72](#)

PGPURGE, system initialization parameter [316](#)

PGRET, system initialization parameter [316](#)

PLT (program list table)
 PLTPI, system initialization parameter [316](#)
 PLTPISEC, system initialization parameter [316](#)
 PLTPIUSR, system initialization parameter [316](#)
 PLTSD, system initialization parameter [316](#)

PLTPI, system initialization parameter [316](#)

PLTPISEC, system initialization parameter [316](#)

PLTPIUSR, system initialization parameter [316](#)

PLTSD, system initialization parameter [316](#)

plus 32K COMMAREAs [213](#), [236](#)

port numbers [45](#)

PORTNUMBER attribute
 TCPIP SERVICE definition [162](#)

PORTNUMBER option
 EXTRACT TCPIP command [67](#)
 WEB EXTRACT command [85](#)

PORTNUMNU option
 EXTRACT TCPIP command [68](#)

PRFILE
 CEMT INQUIRE TRANSACTION [171](#)

PRGDLY, system initialization parameter [316](#)

PRINT, system initialization parameter [316](#)

PRINTERCOMP attribute
 PROFILE definition [158](#)

printing facilities
 PRINT, system initialization parameter [316](#)
 screen copying [316](#)

PRIORITY
 CEMT INQUIRE TASK [176](#)
 CEMT INQUIRE TRANSACTION [171](#)

private key
 description [126](#)

processing containers in a sub-routine [227](#)

PROFILE attribute
 PROFILE definition [158](#)

PROFILE definition
 CHAINCONTROL attribute [155](#)
 DESCRIPTION attribute [155](#)
 DVSUPRT attribute [156](#)
 FACILITYLIKE attribute [156](#)
 GROUP attribute [156](#)
 INBFMH attribute [156](#)
 JOURNAL attribute [156](#)
 LOGREC attribute [157](#)
 MODENAME attribute [157](#)
 MSGINTEG attribute [157](#)

PROFILE definition (*continued*)
 MSGJRN attribute [157](#)
 NEPCCLASS attribute [157](#)
 ONEWTE attribute [158](#)
 PRINTERCOMP attribute [158](#)
 PROFILE attribute [158](#)
 PROTECT attribute [158](#)
 RAQ attribute [158](#)
 RDO command (CEDA DEFINE) [155](#)
 RTIMOUT attribute [159](#)
 SCRNSIZE attribute [159](#)
 UCTRAN attribute [159](#)

PROGRAM
 CEMT INQUIRE TRANSACTION [171](#)

PROGRAM definitions [43](#)

PROGRAMNAME attribute
 DOCTEMPLATE definition [151](#)

PROTECT attribute
 PROFILE definition [158](#)

PRTYAGE, system initialization parameter [316](#)

PRVMOD, system initialization parameter [316](#)

PSDINT, system initialization parameter [294](#), [316](#)

public key
 description [126](#)

Public key cryptography standard (PKCS) [126](#)

PURGE
 CEMT INQUIRE TASK [176](#)

purge delay time interval, BMS [316](#)

PURGEABILITY
 CEMT INQUIRE TRANSACTION [171](#)

PURGEABLE
 CEMT INQUIRE TRANSACTION [171](#)

PURGETYPE
 CEMT INQUIRE TASK [176](#)

PUT CONTAINER (CHANNEL) command [201](#)

PVDELAY, system initialization parameter [316](#)

Q

QD
 CEMT INQUIRE TASK [177](#)

QIDERR condition
 DELETEQ TS command [76](#)
 READQ TS command [78](#)
 WRITEQ TS command [81](#)

QNAME option
 DELETEQ TS command [75](#)
 READQ TS command [77](#)
 WRITEQ TS command [80](#)

QUERYSTRING option
 WEB EXTRACT command [85](#)

QUERYSTRLEN option
 WEB EXTRACT command [85](#)

QUEUE option
 DELETEQ TS command [75](#)
 READQ TS command [77](#)
 WRITEQ TS command [80](#)

R

RAMAX, system initialization parameter [316](#)

RAPOOL, system initialization parameter [316](#)

RAQ attribute

RAQ attribute (*continued*)
 PROFILE definition [158](#)

RDO (resource definition online)
 group list (GRPLIST) [316](#)

RDO command
 DOCTEMPLATE [151](#)
 PROFILE [155](#)
 TCPIP SERVICE [161](#)

RDSASZE, system initialization parameter [316](#)

read only containers [228](#)

read-only storage
 system initialization parameter [316](#)

reading records
 from temporary storage queue [76](#)

READQ TS command [76](#)

RECEIVE ANY (RA) maximum [316](#)

RECEIVE ANY (RA) pool size [316](#)

reconnection delay interval (XRF) [316](#)

reconnection transaction for XRF [316](#)

RECUNITID
 CEMT INQUIRE TASK [176](#)

region exit interval (ICV) [316](#)

RENTPGM, system initialization parameter [316](#)

request parameter list (RPL) [316](#)

REQUESTTYPE option
 WEB EXTRACT command [85](#)

resource backout at emergency restart [316](#)

Resource definition macro [259](#)

RESP, system initialization parameter [316](#)

RESSEC, system initialization parameter [316](#)

RETURN
 migrating to channels and containers [238](#)

RETURN command [204](#)

REWRITE option
 WRITEQ TS command [80](#)

Rich Client Platform (RCP)
 CICS Explorer [3](#)

RMTRAN, system initialization parameter [316](#)

RPG applications [281](#)

RPL (request parameter list) [316](#)

RTIMOUT attribute
 PROFILE definition [159](#)

RUNNING
 CEMT INQUIRE TASK [176](#)

RUNSTATUS
 CEMT INQUIRE TASK [176](#)

RUWAPool, system initialization parameter [316](#)

S

S
 CEMT INQUIRE TASK [177](#)

SADDRLENGTH option
 EXTRACT TCPIP command [68](#)

sample application DFHOWBCA [132](#)

samples
 application [46](#)
 security [47](#)

scenarios
 multiple interactive components [218](#)
 one channel—one program [216](#)
 one channel—several programs [217](#)
 several channels, one component [217](#)

SCHEME option

SCHEME option (*continued*)
 WEB EXTRACT command [85](#)

scope of a channel
 example, with LINK commands [224](#)
 example, with XCTL commands [226](#)
 overview [224](#)

screen copying [316](#)

SCRNSIZE attribute
 PROFILE definition [159](#)

SD
 CEMT INQUIRE TASK [177](#)

SDSASZE, system initialization parameter [316](#)

SEC, system initialization parameter [316](#)

SECPRFX, system initialization parameter [316](#)

Secure Sockets Layer (see also SSL) [123](#)

security
 DFLTUSER, system initialization parameter [316](#)
 ESMEXITS, system initialization parameter [316](#)
 for transactions [316](#)
 MRO bind-time security [316](#)
 of attached entries [316](#)
 PLTPISEC, system initialization parameter [316](#)
 PLTPIUSR, system initialization parameter [316](#)
 resource class names [316](#)
 RESSEC, system initialization parameter [316](#)
 SEC, system initialization parameter [316](#)
 SECPRFX, system initialization parameter [316](#)
 security checking for EXEC CICS system commands [316](#)
 security checking for program entries [316](#)
 security checking for temporary storage entries [316](#)
 security checking of destination control entries [316](#)
 security checking of EXEC-started transaction entries [316](#)
 security checking of file control entries [316](#)
 security checking of journal entries [316](#)
 security checking of PSB entries [316](#)
 specifying a prefix to resource name [316](#)
 specifying security checking of DCT entries [316](#)
 using an ESM to establish APPC sessions [316](#)
 XAPPC, system initialization parameter [316](#)
 XCMD, system initialization parameter [316](#)
 XDCT, system initialization parameter [316](#)
 XFCT, system initialization parameter [316](#)
 XJCT, system initialization parameter [316](#)
 XPCT, system initialization parameter [316](#)
 XPPT, system initialization parameter [316](#)
 XPSB, system initialization parameter [316](#)
 XTRAN, system initialization parameter [316](#)
 XTST, system initialization parameter [316](#)
 XUSER, system initialization parameter [316](#)

SERIALNUM option
 EXTRACT CERTIFICATE command [70](#)

SERIALNUMLEN option
 EXTRACT CERTIFICATE command [70](#)

SERVADDRNU option
 EXTRACT TCPIP command [68](#)

server certificate
 overview of use [127](#)

SERVERADDR option
 EXTRACT TCPIP command [68](#)

SERVERCONV option
 WEB RECEIVE command [93](#)
 WEB SEND command [97](#)

SERVERNAME option
 SERVERNAME option (*continued*)
 EXTRACT TCPIP command [68](#)
 service functions for client authentication [132](#)

SET option
 READQ TS command [77](#)
 WEB READ FORMFIELD command [87](#)
 WEB RECEIVE command [93](#)

signing certificates [126](#)

single keystroke retrieval (SKR) [316](#)

SIT, system initialization parameter [316](#)

SKR (single keystroke retrieval) [316](#)

SKRxxxx, system initialization parameter [316](#)

SNAMELENGTH option
 EXTRACT TCPIP command [68](#)

SNSCOPE, system initialization parameter [316](#)

SOCKETCLOSE attribute
 TCPIP SERVICE definition [163](#)

SPCTR, system initialization parameter [316](#)

SPCTRxx, system initialization parameter [316](#)

SPOOL, system initialization parameter [316](#)

SRT (system recovery table)
 SRT, system initialization parameter [316](#)

SRT, system initialization parameter [316](#)

SSL
 authorization level [129](#)
 cipher suites supported [132](#)
 client authentication [132](#)
 client keyring file on Web clients or middle-tier [131](#)
 configuring CICS clients for client authentication [132](#)
 configuring z/VSE system for [131](#)
 defining a TCPIP SERVICE resource for [130](#)
 ENCRYPTION, system initialization parameter [316](#)
 introduction to [125](#)
 mapping client certificates to VSE User IDs [132](#)
 sample program DFHOWBCA [132](#)
 security in [126](#)
 server and client authentication [129](#)
 server authentication [129](#)
 SIT parameters for [130](#)

SSL attribute
 NO, YES, CLIENTAUTH, options [163](#)
 TCPIP SERVICE definition [163](#)

SSLDELAY, system initialization parameter [316](#)

STANDARD option of system initialization parameter BMS [316](#)

STANDBY start option [316](#)

standby start-up for XRF [316](#)

START CHANNEL command [206](#)

START command [71](#)

START data, migrating to use channels [238](#)

START, system initialization parameter
 (option,ALL) [316](#)
 START=AUTO [312](#)
 START=COLD [312](#)
 START=LOGTERM [312](#), [316](#)
 START=STANDBY [312](#)

STARTBROWSE CONTAINER command [209](#)

STARTCODE
 CEMT INQUIRE TASK [176](#)

STARTER, system initialization parameter [316](#)

starting CICS
 ACB at CICS startup [315](#)
 defining your SIT to CICS at startup [302](#)

starting CICS regions

- starting CICS regions (*continued*)
 - specifying the type of startup [310](#)
 - START=AUTO [312](#)
 - START=STANDBY, for an XRF alternate CICS [312](#)
- STATE option
 - EXTRACT CERTIFICATE command [70](#)
- STATELEN option
 - EXTRACT CERTIFICATE command [70](#)
- statistics
 - effect of START= parameter [312](#)
 - STATRCD, system initialization parameter [316](#)
- Statistics [253](#), [275](#)
- STATRCD, system initialization parameter [316](#)
- STATUS
 - CEMT INQUIRE TRANSACTION [171](#)
- STATUS attribute
 - TCPIP SERVICE definition [164](#)
- STATUSCODE option
 - WEB SEND command [98](#)
- STATUSLEN option
 - WEB SEND command [98](#)
- STATUSTEXT option
 - WEB SEND command [98](#)
- STGPROT, system initialization parameter [316](#)
- STGRVCY, system initialization parameter [316](#)
- STNTR, system initialization parameter [316](#)
- STNTRxx, system initialization parameter [316](#)
- storage
 - CHKSTRM system initialization parameter [316](#)
 - CHKSTSK system initialization parameter [316](#)
 - DSALIM, system initialization parameter [316](#)
 - EDSALIM, system initialization parameter [316](#)
 - for the trace table above the 16MB line [316](#)
 - RENTPGM, system initialization parameter [316](#)
 - STGPROT, system initialization parameter [316](#)
 - STGRVCY, system initialization parameter [316](#)
- storage protection system initialization parameter, STGPROT [316](#)
- storage trace
 - auxiliary [316](#)
 - main [316](#)
 - trace option in transaction dump [316](#)
 - trace table size in main storage [316](#)
 - trace table size in transaction dump [316](#)
- strings and buffers, VSAM [316](#)
- sublibrary [41](#)
- SUFFIX, system initialization parameter [316](#)
- SUSPENDED
 - CEMT INQUIRE TASK [176](#)
- SVA (shared virtual area)
 - PRVMOD, system initialization parameter [316](#)
 - SVA system initialization parameter [316](#)
- SVA, system initialization parameter [316](#)
- SYDUMAX, system initialization parameter [316](#)
- SYMBOL option
 - DOCUMENT INSERT command [61](#)
 - DOCUMENT SET command [64](#)
- SYMBOLERR condition
 - DOCUMENT SET command [65](#)
- SYMBOLLIST option
 - DOCUMENT CREATE command [59](#)
 - DOCUMENT SET command [64](#)
- SYSID option
 - DELETEQ TS command [75](#), [80](#)
- SYSID option (*continued*)
 - READQ TS command [77](#)
- SYSIDERR condition
 - DELETEQ TS command [76](#)
 - READQ TS command [78](#)
 - WRITEQ TS command [81](#)
- SYSIDNT, system initialization parameter [316](#)
- SYSIN (SI), PARM keyword [303](#)
- System Definition Data Set (CSD) [253](#), [261](#)
- system identifier, system initialization parameter SYSIDNT [316](#)
- system initialization
 - for an alternate CICS (XRF=YES)
 - START=STANDBY [312](#)
 - how CICS determines the type of startup [310](#)
 - START=AUTO [312](#)
- system initialization parameters
 - ADI, alternate delay interval (XRF) [316](#)
 - AIEXIT [316](#)
 - AILDELAY [316](#)
 - AIQMAX [316](#)
 - AIRDELAY [316](#)
 - AKPFREQ [316](#)
 - APPLID [316](#)
 - AUTCONN [316](#)
 - AUXTR [316](#)
 - AUXTRSW [316](#)
 - BMS [316](#)
 - CDSASZE [316](#)
 - CHKSTRM [316](#)
 - CHKSTSK [316](#)
 - CLSDSTP [316](#)
 - CLT [316](#)
 - CMDPROT [316](#)
 - CMDSEC [316](#)
 - CONFDATA [316](#)
 - CONFTXT [316](#)
 - CSDACC [316](#)
 - CSDBUFND [316](#)
 - CSDBUFNI [316](#)
 - CSDFRLOG [316](#)
 - CSDJID [316](#)
 - CSDLRNO [316](#)
 - CSDRECOV [316](#)
 - CSDSTRNO [316](#)
 - DATFORM [316](#)
 - DBP [316](#)
 - DBUFSZ [316](#)
 - DCT [316](#)
 - DFLTUSER [316](#)
 - DIP [316](#)
 - DISMACP [316](#)
 - DLI (DL1) [316](#)
 - DLIOER [316](#)
 - DOCCODEPAGE [316](#)
 - DSALIM [316](#)
 - DSHIPIDL [316](#)
 - DSHIPINT [316](#)
 - DTRPGM [316](#)
 - DTRTRAN [316](#)
 - DUMP [316](#)
 - DUMPDS [316](#)
 - DUMPSW [316](#)
 - ECDASZE [316](#)

system initialization parameters (*continued*)

[EDSALIM 316](#)
[ENCRYPTION 316](#)
[entering data at the console 307](#)
[EODI 316](#)
[ERDSASZE 316](#)
[ESDSASZE 316](#)
[ESMEXITS 316](#)
[EUDSASZE 316](#)
[FCT 316](#)
[FEPI 316](#)
[FLDSEP 316](#)
[FLDSTRT 316](#)
[from operator's console 307](#)
[FSSTAFF 316](#)
[GMTEXT 316](#)
[GMTRAN 316](#)
[GNTRAN 316](#)
[GRPLIST 316](#)
[how to specify 293](#)
[ICP 316](#)
[ICV 316](#)
[ICVR 316](#)
[ICVTSD 316](#)
[INITPARM 316](#)
[INTRR 316](#)
[IRCSTRT 316](#)
[ISC 316](#)
[JCT 316](#)
[JSTATUS 316](#)
[KEYFILE 316](#)
[LEVSE 316](#)
[LGNMSG 316](#)
[MCT 316](#)
[migration considerations 294](#)
[MN 316](#)
[MNCONV 316](#)
[MNEXC 316](#)
[MNFREQ 316](#)
[MNPER 316](#)
[MNSYNC 316](#)
[MNTIME 316](#)
[MROBTCH 316](#)
[MROFSE 316](#)
[MROLRM 316](#)
[MSGCASE 316](#)
[MSGLVL 316](#)
[MXT 316](#)
[NATLANG 316](#)
[NEWSIT 316](#)
[OPERTIM 316](#)
[PARMERR 316](#)
[PDI 316](#)
[PGAICTLG 316](#)
[PGAIXIT 316](#)
[PGAIPGM 316](#)
[PGCHAIN 316](#)
[PGCOPY 316](#)
[PGPURGE 316](#)
[PGRET 316](#)
[PLTPI 316](#)
[PLTPISEC 316](#)
[PLTPIUSR 316](#)
[PLTSD 316](#)

system initialization parameters (*continued*)

[PRGDLAY 316](#)
[PRINT 316](#)
[PRTYAGE 316](#)
[PRVMOD 316](#)
[PSDINT 316](#)
[PVDELAY 316](#)
[RAMAX 316](#)
[RAPOOL 316](#)
[RDSASZE 316](#)
[RENTPGM 316](#)
[RESP 316](#)
[RESSEC 316](#)
[RMTRAN 316](#)
[RUWAPOOL 316](#)
[SDSASZE 316](#)
[SEC 316](#)
[SECPRFX 316](#)
[SIT 316](#)
[SKRxxxx 316](#)
[SNSCOPE 316](#)
[SPCTR 316](#)
[SPCTRxx 316](#)
[SPOOL 316](#)
[SRT 316](#)
[SSLDELAY 316](#)
[START 316](#)
[STARTER 316](#)
[STATRCD 316](#)
[STGPROT 316](#)
[STGRVCY 316](#)
[STNTR 316](#)
[STNTRxx 316](#)
[SUFFIX 316](#)
[SVA 316](#)
[SYDUMAX 316](#)
[SYSIDNT 316](#)
[SYSTR 316](#)
[TAKEOVR 316](#)
[TBEXITS 316](#)
[TCP 316](#)
[TCPIP 316](#)
[TCSACTN 316](#)
[TCSWAIT 316](#)
[TCT 316](#)
[TCTUAKEY 316](#)
[TCTUALOC 316](#)
[TD 316](#)
[TRAP 316](#)
[TRDUMAX 316](#)
[TRTABSZ 316](#)
[TRTRANSZ 316](#)
[TRTRANTY 316](#)
[TS 316](#)
[TSMGSET 316](#)
[TST 316](#)
[TYPE 316](#)
[UDSASZE 316](#)
[USERTR 316](#)
[USRDELAY 316](#)
[VTAM 316](#)
[VTPREFIX 316](#)
[WEBDELAY 316](#)
[WRKAREA 316](#)

system initialization parameters (*continued*)

- XAPPC [316](#)
- XCMD [316](#)
- XDCT [316](#)
- XFCT [316](#)
- XJCT [316](#)
- XLT [316](#)
- XPCT [316](#)
- XPPT [316](#)
- XPSB [316](#)
- XRF [316](#)
- XRFSOFF [316](#)
- XRFSTME [316](#)
- XRFTODI [316](#)
- XSWITCH [316](#)
- XTRAN [316](#)
- XTST [316](#)
- XUSER [316](#)

system initialization programs

- INITPARM, system initialization parameter [316](#)
- PLTPI, system initialization parameter [316](#)
- PLTPISEC, system initialization parameter [316](#)
- PLTPIUSR, system initialization parameter [316](#)

system initialization table (SIT)

- .END, PARM keyword [304](#)
- assembling the SIT [301](#)
- coding the PARM parameter over two lines [305](#)
- CONSOLE (CN), PARM keyword [304](#)
- creating a SIT [300](#)
- creating more than one SIT [300](#)
- defining your SIT to CICS at startup [302](#)
- DFHSIT TYPE=CSECT [316](#)
- parameters that cannot be coded in DFHSIT [301](#)
- PARM parameter, processing [305](#)
- SIT, system initialization parameter [316](#)
- SITs supplied with z/VSE [301](#)
- supplied SITs [300](#)
- SYSIN (SI), PARM keyword [303](#)
- SYSIPT data set, processing [306](#)

System initialization table DFHSIT [253](#), [257](#)

system spooling interface [316](#)

SYSTR, system initialization parameter [316](#)

T

takeover action for XRF [316](#)

TAKEOVR, system initialization parameter [316](#)

TASK

- CEMT INQUIRE TASK [175](#), [177](#)

TASK command

- CEMT INQUIRE/SET transaction [173](#)

TASKDATAKEY

- CEMT INQUIRE TRANSACTION [171](#)

TASKDATALOC

- CEMT INQUIRE TRANSACTION [171](#)

tasks

- CEMT INQUIRE/SET requests [173](#)

TBEXITS, system initialization parameter [316](#)

TCLASS

- CEMT INQUIRE TASK [174](#)
- CEMT INQUIRE TRANSACTION [171](#)

TCP, system initialization parameter [316](#)

TCP/IP

- CICS ECI over TCP/IP [137](#)

- security in [125](#)

TCP/IP services, CEMT requests [171](#)

TCPIP

- CEMT INQUIRE/SET transaction [171](#)

TCPIP system initialization parameter [39](#)

TCPIP, system initialization parameter [316](#)

TCPIPSERVICE

- CEMT INQUIRE/SET transaction [171](#)

- for ECI [137](#)

- for SSL [130](#)

TCPIPSERVICE attribute

- TCPIPSERVICE definition [164](#)

TCPIPSERVICE definition

- ATTACHSEC attribute [161](#)

- BACKLOG attribute [162](#)

- CERTIFICATE attribute [162](#)

- DESCRIPTION attribute [162](#)

- GROUP attribute [162](#)

- IPADDRESS attribute [162](#)

- PORTNUMBER attribute [162](#)

- RDO command (CEDA DEFINE) [161](#)

- SOCKETCLOSE attribute [163](#)

- SSL attribute [163](#)

- STATUS attribute [164](#)

- TCPIPSERVICE attribute [164](#)

- TRANSACTION attribute [164](#)

- TSQPREFIX attribute [164](#)

- URM attribute [164](#)

TCPIPSERVICE definitions [263](#)

TCPIPSERVICE option

- EXTRACT TCPIP command [68](#)

TCSACTN, system initialization parameter [316](#)

TCSWAIT, system initialization parameter [316](#)

TCT (terminal control table)

- TCT, system initialization parameter [316](#)

TCTUAKEY, system initialization parameter [316](#)

TCTUALOC, system initialization parameter [316](#)

TD queue [41](#)

TD, system initialization parameter [316](#)

TDQUEUE attribute

- DOCTEMPLATE definition [151](#)

TEMPLATE option

- DOCUMENT INSERT command [61](#)

TEMPLATENAME

- CEMT INQUIRE DOCTEMPLATE [171](#)

TEMPLATENAME attribute

- DOCTEMPLATE definition [151](#)

TEMPLATETYPE

- CEMT INQUIRE DOCTEMPLATE [171](#)

temporary storage

- message set, TSMGSET [316](#)

- TS, system initialization parameter [316](#)

- TSMGSET, system initialization parameter [316](#)

- VSAM buffers and strings [316](#)

temporary storage browse transaction, CEBR [167](#)

temporary storage queue [41](#)

TERM

- CEMT INQUIRE TASK [175](#)

terminal control table (TCT)

- dummy control table, DFHTCTDY [310](#)

terminal control table (TCT) *(continued)*
 user area storage key
 system initialization parameter [316](#)

terminal scan delay, ICVTSD [316](#)

terminals
 CLSDSTP system initialization parameter [316](#)

TEXT option
 DOCUMENT INSERT command [61](#)

time interval, region exit [316](#)

timeout limit, userid [316](#)

TO
 CEMT INQUIRE TASK [177](#)

TO option
 DOCUMENT INSERT command [61](#)

TP
 CEMT INQUIRE TASK [177](#)

trace entries, interpretation
 AP 21xx, bridge facility management [397](#)
 AP 28xx, bridge facility management 2 [401](#)
 SO xxxx, Socket domain [376](#)

trace entries, Web domain
 WB xxxx, Web domain [394](#)

Trace formatting utility program DFHTU430 [279](#)

tracing facilities
 auxiliary storage trace [316](#)
 auxiliary trace autoswitch facility [316](#)
 AUXTR, system initialization parameter [316](#)
 AUXTRSW, system initialization parameter [316](#)
 CICS standard tracing, setting levels of [316](#)
 INTTR, system initialization parameter [316](#)
 option in transaction dump [316](#)
 SM component, warning when setting trace level [316](#)
 SPCTR, system initialization parameter [316](#)
 SPCTRxx, system initialization parameter [316](#)
 special tracing, setting levels of [316](#)
 STNTR, system initialization parameter [316](#)
 STNTRxx, system initialization parameter [316](#)
 SYSTR, system initialization parameter [316](#)
 table size in main storage [316](#)
 table size in transaction dump [316](#)
 TRTABSZ, system initialization parameter [316](#)
 TRTRANSZ, system initialization parameter [316](#)
 TRTRANTY, system initialization parameter [316](#)
 USERTR, system initialization parameter [316](#)

TRANID
 CEDX [169](#)
 CEMT INQUIRE TASK [177](#)

TRANSACTION
 CEMT INQUIRE TRANSACTION [171](#)
 CEMT INQUIRE/SET transaction
[171](#)

TRANSACTION attribute
 TCPIP SERVICE definition [164](#)

transaction backout exit programs [316](#)

Transaction dump utility program DFH DU430 [279](#)

transactions
 CEDF [169](#)

TRANSID option
 START BREXIT command [72](#)

TRANSIDERR condition
 START BREXIT command [72](#)

transient data
 TD, system initialization parameter [316](#)
 VSAM buffers and strings [316](#)

transient data queue [41](#)

TRAP, system initialization parameter [316](#)

TRDUMAX, system initialization parameter [316](#)

TRPROF
 CEMT INQUIRE TRANSACTION [171](#)

TRTABSZ, system initialization parameter [316](#)

TRTRANSZ, system initialization parameter [316](#)

TRTRANTY, system initialization parameter [316](#)

TS queue [41](#)

TS, system initialization parameter [316](#)

TSMGSET, system initialization parameter [316](#)

TSQPREFIX attribute
 TCPIP SERVICE definition [164](#)

TSQUEUE
 CEMT INQUIRE transaction [171](#)
 CEMT INQUIRE TSQUEUE [171](#)

TSQUEUE attribute
 DOCTEMPLATE definition [151](#)

TST (temporary storage table)
 specifying security checking of temporary storage
 entries [316](#)
 TST, system initialization parameter [316](#)

TST, system initialization parameter [316](#)

TYPE attribute
 CEMT INQUIRE DOCTEMPLATE [171](#)
 DOCTEMPLATE definition [151](#)

TYPE option
 WEB RECEIVE command [93](#)

TYPE, system initialization parameter [316](#)

TYPE=CSECT, DFHSIT [316](#)

TYPE=DSECT, DFHSIT [316](#)

U

U
 CEMT INQUIRE TASK [177](#)

UCTRAN attribute
 PROFILE definition [159](#)

UDATAKEY
 CEMT INQUIRE TRANSACTION [171](#)

UDSASZE, system initialization parameter [316](#)

URM attribute
 TCPIP SERVICE definition [164](#)

user workspace for CICS Explorer
 changing [28](#)

user-replaceable programs [43](#)

User-replaceable programs
 DFHCNV [269](#)
 DFHUCNV [269](#)

USERID
 CEMT INQUIRE TASK [177](#)

USERID option
 EXTRACT CERTIFICATE command [70](#)
 START BREXIT command [72](#)

userid timeout limit [316](#)

USERIDERR condition
 START BREXIT command [72](#)

USERTR, system initialization parameter [316](#)

USRDELAY, system initialization parameter [316](#)

V

value

value (*continued*)

- CEMT INQUIRE TASK [174](#)
- CEMT INQUIRE TRANSACTION [171](#)
- CEMT INQUIRE TSQUEUE [171](#)

VALUE option

- DOCUMENT SET command [65](#)
- WEB READ FORMFIELD command [87](#)
- WEB READ HTTPHEADER command [88](#)
- WEB READNEXT FORMFIELD command [89](#)
- WEB READNEXT HTTPHEADER command [90](#)
- WEB WRITE command [102](#)

VALUELENGTH option

- WEB READ FORMFIELD command [87](#)
- WEB READ HTTPHEADER command [88](#)
- WEB READNEXT FORMFIELD command [89](#)
- WEB READNEXT HTTPHEADER command [90](#)
- WEB WRITE command [102](#)

VERSIONLEN option

- WEB EXTRACT command [85](#)

views

- CICS Explorer [6](#)

VSAM (virtual storage access method)

- buffers and strings [316](#)
- TS, system initialization parameter [316](#)

VSAM buffers and strings [316](#)

VSE Keyring Library, specifying [316](#)

VTAM (virtual telecommunications access method)

- ACB at CICS startup [315](#)
- CONFTXT, system initialization parameter [316](#)
- LGNMSG, system initialization parameter [316](#)
- logon data [316](#)
- PSDINT, system initialization parameter [316](#)
- VBUILD TYPE=APPL statement [316](#)
- VTAM, system initialization parameter [316](#)

VTAM, system initialization parameter [316](#)

VTPREFIX, system initialization parameter [316](#)

W

WAIT JOURNALNUM command [288](#)

warm start [316](#)

Web attach transaction CWXN [42](#)

WEB ENDBROWSE FORMFIELD command [83](#)

WEB ENDBROWSE HTTPHEADER command [83](#)

WEB EXTRACT command [84](#)

WEB READ FORMFIELD command [86](#)

WEB READ HTTPHEADER command [88](#)

WEB READNEXT FORMFIELD command [89](#)

WEB READNEXT HTTPHEADER command [90](#)

WEB RECEIVE command [91](#)

WEB RETRIEVE command [94](#)

WEB SEND command [95](#)

WEB STARTBROWSE FORMFIELD command [100](#)

WEB STARTBROWSE HTTPHEADER command [101](#)

WEB WRITE command [102](#)

WEBDELAY, system initialization parameter [316](#)

welcome (good morning) message [316](#)

workspace (user) for CICS Explorer

- changing [28](#)

write-to-operator timeout limit [316](#)

WRITEQ TS command [78](#)

writing data

- to temporary storage queue [78](#)

WRKAREA, system initialization parameter [316](#)

X

X.509 certificates [126](#)

XAPPC, system initialization parameter [316](#)

XCMD, system initialization parameter [316](#)

XCTL

- migrating to channels and containers [237](#)

XCTL command [210](#)

XDBDERR, global user exit [316](#)

XDCT, system initialization parameter [316](#)

XFCT, system initialization parameter [316](#)

XJCT, system initialization parameter [316](#)

XLT, system initialization parameter [316](#)

XPCC, cross-partition communication component [316](#)

XPCT, system initialization parameter [316](#)

XPPT, system initialization parameter [316](#)

XPSB, system initialization parameter [316](#)

XRF (extended recovery facility)

- ADI, system initialization parameter [316](#)

AIRDELAY parameter (active and alternate CICS) [316](#)

APPLID system initialization parameter (active and alternate) [316](#)

AUTCONN, system initialization parameter (alternate) [316](#)

CLT system initialization parameter (alternate) [316](#)

command list table (CLT) [316](#)

DUMP system initialization parameter (active and alternate) [316](#)

generic and specific applids [316](#)

GOOD MORNING transaction [316](#)

PDI system initialization parameter [316](#)

PDI, system initialization parameter [316](#)

primary delay interval (PDI) [316](#)

reconnection delay [316](#)

reconnection transaction [316](#)

RMTRAN, system initialization parameter [316](#)

START=STANDBY (alternate) [316](#)

TAKEOVR system initialization parameter (alternate) [316](#)

VTAM ACB at startup [315](#)

XRF system initialization parameter (active and alternate) [316](#)

XRFTODI, system initialization parameter [316](#)

XSWITCH, system initialization parameter [316](#)

XRF, system initialization parameter [316](#)

XRFSOFF, system initialization parameter [316](#)

XRFSTME, system initialization parameter [316](#)

XRFTODI, system initialization parameter [316](#)

XSWITCH, system initialization parameter [316](#)

XTRAN, system initialization parameter [316](#)

XTST, system initialization parameter [316](#)

XUSER, system initialization parameter [316](#)

Z

z/VSE

- configuring host to use CICS Explorer [17](#)

z/VSE sublibrary [41](#)



Product Number: 5655-VSE

SC34-2685-01

