CICS Transaction Server for z/OS



User Exit Reference

Version 5 Release 4

CICS Transaction Server for z/OS



User Exit Reference

Version 5 Release 4

Note
Before using this information and the product it supports, read the information in "Notices" on page 277.

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

© Copyright IBM Corporation 1977, 2017.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this PDF vii

Global user exit points (by function) . . 1

Global user exit points (in alphabetical order) . . . 1
Activity keypoint program exit (XAKUSER) 9
Application association data exit in the AP domain (XAPADMGR) 10
Basic mapping support exits (XBMIN, XBMOUT) . . 11
Exit XBMIN 12
Exit XBMOUT 12
The field element table structure 13
Programming the XBMIN exit 14
Programming the XBMOUT exit 15
Bridge facility exit XFAINTU 15
Data tables management exits XDTRD, XDTAD, and XDTLC 17
Exit XDTRD 18
Exit XDTAD 20
Exit XDTLC 21
DBCTL interface control program exit (XXDFA) . . 22
DBCTL tracking program exits (XXDFB, XXDTO) . . 23
Exit XXDFB 23
Exit XXDTO 23
Dispatcher domain exits XDSBWT and XDSAWT . . 24
Exit XDSBWT 25
Exit XDSAWT 25
DL/I interface program exits XDLIPRE and XDLIPOST. 26
Exit XDLIPRE 26
Exit XDLIPOST 28
Example use of global user exit XDLIPRE . . . 30
Dump domain exits XDUREQ, XDUREQC, XDUCLSE, and XDUOUT 36
Exit XDUREQ 36
Exit XDUREQC 39
Exit XDUCLSE 42
Exit XDUOUT 42
Enqueue EXEC interface program exits XNQEREQ and XNQEREQC 43
Exit XNQEREQ 44
Exit XNQEREQC 45
The command-level parameter structure 46
Event capture exit XEPCAP 50
EXEC interface program exits XEIIN, XEIOUT, XEISPIN, and XEISPOUT 50
The command parameter list 51
Bypassing commands 52
Exit XEIIN. 52
Exit XEISPIN 53
Exit XEIOUT 53
Exit XEISPOUT 54
Front End Programming Interface exits XSZARQ and XSZBRQ 55
XSZBRQ 56
XSZARQ 57
The UEPSZACT and UEPSZACN exit-specific parameters 58
Using XMEOUT to control message output . . . 59
File control domain exits, XFCFRIN and XFCFROUT 59
Exit XFCFRIN 60
Exit XFCFROUT 66
File control EXEC interface API exits XFCREQ and XFCREQC 72
The command-level parameter structure 73
Modifying fields in the command-level parameter structure. 77
Modifying the EID 79
Use of the parameter UEPFSHIP 80
EIB (EXEC interface block) 80
Example of how XFCREQ and XFCREQC can be used 81
Exit XFCREQ 81
Exit XFCREQC 83
File control EXEC interface SPI exits XFCAREQ and XFCAREQC 84
Exit XFCAREQ 85
Exit XFCAREQC. 86
The command-level parameter structure 87
Modifying fields in the command-level parameter structure. 94
Modifying the EID 97
Modifying user arguments 98
File control file state program exits XFCSREQ and XFCSREQC 98
Exit XFCSREQ 100
Exit XFCSREQC 103
File control open/close program exit XFCNREC 108
XFCNREC exit with a backout recovery setting mismatch. 109
Using XFCNREC with a BWO mismatch . . . 109
File control quiesce receive exit, XFCVSDS. . . . 110
Exit XFCVSDS 110
File control quiesce send exit XFCQUIS. 112
File control recovery program exits XFCBFAIL, XFCBOUT, XFCBOVER, and XFCLDEL. 113
Order of invocation 114
Enabling the exit programs 114
Exit XFCBFAIL, file control backout failure exit 114
Exit XFCBOUT, file control backout exit . . . 118
Exit XFCBOVER, file control backout override exit 119
Exit XFCLDEL, file control logical delete exit 121
File control RLS coexistence program exit XFCRLSCO 122
Good morning message program exit (XGMTEXT) 124
HTTP client open and send exits: XWBAUTH, XWBOPEN and XWBSNDO 124
HTTP client send exit XWBAUTH 125
Typical use of the LDAP XPI functions by XWBAUTH 127
HTTP client open exit XWBOPEN 128

© Copyright IBM Corp. 1977, 2017

HTTP client send exit XWBSNDO 130
Intersystem communication program exits, XISCONA, XISLCLQ, and XISQLCL. 131
The XISCONA exit 131
The XISLCLQ exit 134
The XISQLCL exit 136
Interval control program exits XICREQ, XICEXP, and XICTENF 137
Exit XICREQ 138
Exit XICEXP. 139
Exit XICTENF 139
Interval control EXEC interface program exits (XICEREQ, XICERES, and XICEREQC) 139
Parameters passed to each of the exits 141
Exit XICEREQ 141
Exit XICERES 143
Exit XICEREQC 144
The command-level parameter structure . . . 145
Loader domain exits XLDLOAD and XLDELETE 157
Exit XLDLOAD. 158
Exit XLDELETE 159
Log manager domain exit XLGSTRM 160
Exit XLGSTRM 162
Example of how to use the XLGSTRM exit . . 163
Message domain exit XMEOUT 163
Exit XMEOUT 165
Monitoring domain exit (XMNOUT). 167
Exit XMNOUT 168
Pipeline domain exits. 169
Exit XWSPRRWI 171
Exit XWSPRROI 172
Exit XWSPRROO 173
Exit XWSPRRWO 174
Exit XWSRQRWO 176
Exit XWSRQROO 177
Exit XWSRQROI 178
Exit XWSRQRWI 179
Exit XWSSRRWO 180
Exit XWSSRROO 181
Exit XWSSRROI 182
Exit XWSSRRWI 183
Program control program exits (XPCREQ, XPCERES, XPCREQC, XPCFTCH, XPCHAIR, XPCTA, and XPCABND) 184
Program control exits XPCREQ, XPCERES, XPCREQC 184
Exit XPCFTCH 195
Exit XPCHAIR 197
Exit XPCTA 198
Exit XPCABND. 201
Resource manager interface program exits (XRMIIN, XRMIOUT) 202
Exit XRMIIN 202
Exit XRMIOUT 203
Resource management installation and discard exit XRSINDI 204
Exit XRSINDI 205
Signon and signoff exits XSNON, XSNOFF, and XSNEX 211
Exit XSNON. 211
Exit XSNOFF 213
Exit XSNEX 213
Statistics domain exit XSTOUT 214
Exit XSTOUT 215
System recovery program exit XSRAB 216
System termination program exit XSTERM . . . 219
Temporary storage domain exits (XTSQRIN, XTSQROUT, XTSPTIN, XTSPTOUT). 220
Exit XTSQRIN 220
Exit XTSQROUT 222
Exit XTSPTIN 223
Exit XTSPTOUT 224
Temporary storage EXEC interface program exits XTSEREQ and XTSEREQC 225
Exit XTSEREQ 226
Exit XTSEREQC 227
The command-level parameter structure . . . 229
Terminal allocation program exit XALCAID . . . 235
Terminal control program exits (XTCIN, XTCOUT, XTCATT). 236
Exit XTCIN 236
Exit XTCOUT 237
Exit XTCATT 237
'Terminal not known' condition exits XALTENF and XICTENF 238
The exits 238
Exit XALTENF 239
Exit XICTENF 242
Transaction manager domain exit XXMATT . . . 244
Transient data program exits (XTDREQ, XTDIN, XTDOUT) 246
Exit XTDREQ 246
Exit XTDIN 247
Exit XTDOUT 247
Transient data EXEC interface program exits XTDEREQ and XTDEREQC 248
Exit XTDEREQ 249
Exit XTDEREQC 250
The command-level parameter structure . . . 251
User log record recovery program exits XRCINIT and XRCINPT 256
Coding the exit programs 257
Enabling the exit programs. 258
Exit XRCINIT 258
Exit XRCINPT 259
SNA LU management program exit (XZCATT) . . 260
SNA working-set module exits (XZCIN, XZCOUT, XZCOUT1, and XZIQUE) 260
Exit XZCIN 260
Exit XZCOUT 261
Exit XZCOUT1 262
XZIQUE exit for managing MRO and APPC intersystem queues 262
Designing an XZIQUE global user exit program 268
XISQUE exit for managing IPIC intersystem queues 269
Exit XISQUE 270
Using an XISQUE global user exit program . . 272
Statistics fields in DFHISRDS 273
Designing an XISQUE global user exit program 273
XRF request-processing program exit XXRSTAT 274
Exit XXRSTAT 275

Notices 277

Index 283

About this PDF

This PDF is a reference of the global user exit points that are provided to allow CICS to transfer control to a global user exit program that you have written. To find out how to use these global user exit points in programs, see the PDF called *Developing CICS System Programs*. Before CICS TS V5.4, the information in this PDF was in the *Customization Guide*.

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

Date of this PDF

This PDF was created on June 6th 2017.

© Copyright IBM Corp. 1977, 2017

Global user exit points (by function)

The exit points are grouped according to their functional relationships.

Grouping is generally based on the CICS® module or domain in which the exit points occur. However, where exit points in different modules serve a similar function, the exits are grouped under a generic name. The groups of exits are presented in alphabetical order of module or generic name.

The following information is provided for each global user exit point:

- Exit identifier
- Exit location
- DFHUEPAR parameters, if any, that are unique to the exit
- Valid return codes
- XPI calls that can be invoked.

Global user exit points (in alphabetical order)

For each exit, this table shows the exit name, the module or domain, where or when the exit is invoked, and includes a link to additional information.

Table 1. Alphabetical list of global user exit points

Exit name	Module or domain	Where or when invoked	Topic
XAKUSER	Activity keypoint program	Immediately before the 'end of keypoint' record is written.	“Activity keypoint program exit (XAKUSER)” on page 9
XALCAID	Terminal allocation program	Whenever an AID with data is canceled.	“Terminal allocation program exit XALCAID” on page 235
XALTENF	Terminal allocation program	When an ATI request from transient data or interval control requires a terminal that is unknown in this system.	“Exit XALTENF” on page 239
XAPADMGR	Application domain	When a non-system task that has no inherited Associated Data Origin Descriptor data is attached.	“Application association data exit in the AP domain (XAPADMGR)” on page 10
XBMIN	Basic Mapping Support	When an input mapping operation completes successfully.	“Exit XBMIN” on page 12
XBMOUT	Basic Mapping Support	When a page of output has been built successfully.	“Exit XBMOUT” on page 12
XDLIPOST	DL/I interface program	On exit from the DL/I interface program.	“Exit XDLIPOST” on page 28
XDLIPRE	DL/I interface program	On entry to the DL/I interface program.	“Exit XDLIPRE” on page 26

© Copyright IBM Corp. 1977, 2017

Table 1. Alphabetical list of global user exit points (continued)

Exit name	Module or domain	Where or when invoked	Topic
XDSAWT	Dispatcher domain	After an operating system wait.	"Exit XDSAWT" on page 25
XDSBWT	Dispatcher domain	Before an operating system wait.	"Exit XDSBWT" on page 25
XDTAD	Data tables management	When a write request is issued to a data table.	"Exit XDTAD" on page 20
XDTLC	Data tables management	At the completion of loading of a data table.	"Exit XDTLC" on page 21
XDTRD	Data tables management	During the loading of a data table, whenever a record is retrieved from the source data set.	"Exit XDTRD" on page 18
XDUCLSE	Dump domain	After the domain closes a transaction dump data set.	"Exit XDUCLSE" on page 42
XDUOUT	Dump domain	Before the domain writes a record to the transaction dump data set.	"Exit XDUOUT" on page 42
XDUREQ	Dump domain	Before the domain takes a system or transaction dump.	"Exit XDUREQ" on page 36
XDUREQC	Dump domain	After a system or transaction dump has been taken (or failed or been suppressed).	"Exit XDUREQC" on page 39
XEIIN	EXEC interface program	Before any EXEC CICS API or SPI command runs.	"Exit XEIIN" on page 52
XEIOUT	EXEC interface program	After any EXEC CICS API or SPI command runs.	"Exit XEIOUT" on page 53
XEISPIN	EXEC interface program	Before any EXEC CICS SPI command *except* EXEC CICS ENABLE, EXEC CICS DISABLE, EXEC CICS EXTRACT EXIT, EXEC CICS PERFORM DUMP, or EXEC CICS RESYNC ENTRYNAM runs.	"Exit XEISPIN" on page 53
XEISPOUT	EXEC interface program	After any EXEC CICS SPI command *except* **EXEC CICS ENABLE**, **EXEC CICS DISABLE**, **EXEC CICS EXTRACT EXIT**, **EXEC CICS PERFORM DUMP**, or **EXEC CICS RESYNC ENTRYNAME** runs.	"Exit XEISPOUT" on page 54
XEPCAP	Event capture	Before an event is captured by CICS event processing.	"Event capture exit XEPCAP" on page 50
XFAINTU	3270 bridge facility management program	When a bridge facility is created or deleted.	"Bridge facility exit XFAINTU" on page 15
XFCAREQ	File control EXEC interface program	Before CICS processes a file control SPI request.	"File control EXEC interface SPI exits XFCAREQ and XFCAREQC" on page 84
XFCAREQC	File control EXEC interface program	After a file control SPI request has completed.	"File control EXEC interface SPI exits XFCAREQ and XFCAREQC" on page 84

Table 1. Alphabetical list of global user exit points (continued)

Exit name	Module or domain	Where or when invoked	Topic
XFCBFAIL	File control recovery control program	When an error occurs during the backout of a UOW.	"Exit XFCBFAIL, file control backout failure exit" on page 114
XFCBOUT	File control recovery control program	When CICS is about to back out a file update.	"Exit XFCBOUT, file control backout exit" on page 118
XFCBOVER	File control recovery control program	When CICS is about to skip backout of a UOW because a batch program has overridden RLS retained lock protection and opened a data set for batch processing.	"Exit XFCBOVER, file control backout override exit" on page 119
XFCFRIN	File control domain	Before a file control request runs.	"Exit XFCFRIN" on page 60
XFCFROUT	File control domain	After a file control request runs.	"Exit XFCFROUT" on page 66
XFCLDEL	File control recovery control program	When backing out writes to a VSAM ESDS or a BDAM data set.	"Exit XFCLDEL, file control logical delete exit" on page 121
XFCNREC	File control open/close program	When a mismatch is detected between the backout recovery setting for a file and its associated data set during file open processing.	"File control open/close program exit XFCNREC" on page 108
XFCQUIS	File control quiesce send program	On completion, successful or failed, of a SET DSNAME QUIESCESTATE command.	"File control quiesce send exit XFCQUIS" on page 112
XFCREQ	File control EXEC interface program	Before CICS processes a file control API request.	"Exit XFCREQ" on page 81
XFCREQC	File control EXEC interface program	After a file control API request has completed.	"Exit XFCREQC" on page 83
XFCRLSCO	File control RLS coexistence program	When the opening of a VSAM RLS file or non-RLS read-only file otherwise fails with an RLS coexistence failure.	"File control RLS coexistence program exit XFCRLSCO" on page 122
XFCSREQ	File control file state program	Before a file OPEN, CLOSE, ENABLE, or DISABLE command is attempted.	"File control file state program exits XFCSREQ and XFCSREQC" on page 98
XFCSREQC	File control file state program	After a file OPEN, CLOSE, CANCEL CLOSE, ENABLE, or DISABLE command has been completed.	"File control file state program exits XFCSREQ and XFCSREQC" on page 98
XFCVSDS	File control quiesce receive program	After RLS has informed CICS that processing is required as a result of a data set-related action occurring in the sysplex.	"File control quiesce receive exit, XFCVSDS" on page 110

Table 1. Alphabetical list of global user exit points (continued)

Exit name	Module or domain	Where or when invoked	Topic
XGMTEXT	"Good morning" message program	Before the “good morning” message is sent.	“Good morning message program exit (XGMTEXT)” on page 124
XICEREQ	Interval control EXEC interface program	Before CICS processes an interval control API request.	“Exit XICEREQ” on page 141
XICEREQC	Interval control EXEC interface program	After an interval control API request has completed.	“Exit XICEREQC” on page 144
XICERES	Interval control EXEC interface program	Before CICS processes a non-terminal-related EXEC CICS START request that has been dynamically routed to this region, where the routing region supports the "resource unavailable" (RESUNAVAIL) condition.	“Exit XICERES” on page 143
XICEXP	Interval control program	After expiry of an interval control time interval.	“Exit XICEXP” on page 139
XICREQ	Interval control program	At the start of the interval control program, before request analysis.	“Exit XICREQ” on page 138
XICTENF	Interval control program	When an EXEC CICS START command requires a terminal that is unknown in this system.	“Exit XICTENF” on page 242
XISCONA	Intersystem communication program	When a function shipping or DPL request is about to be queued because no sessions to the remote region are immediately available.	“Intersystem communication program exits, XISCONA, XISLCLQ, and XISQLCL” on page 131
XISLCLQ	Intersystem communication program	After an attempt to allocate a session for a function shipped START NOCHECK request fails because the remote system is not in service, a connection to the remote system cannot be established, or no sessions are immediately available and your XISCONA exit program has specified that the request is not to be queued in the issuing region.	“The XISLCLQ exit” on page 134
XISQLCL	Intersystem communication program	After an attempt to allocate a session for a START NOCHECK request, that is scheduled for an IPIC connection, fails because the IPIC connection is out of service, the IPIC connection is not acquired, or a session is not available and CICS does not queue the request for a new session.	“The XISQLCL exit” on page 136
XISQUE	To control the number of queued requests for sessions on IPCONNs	When: 1. An allocate request for a session on an IPCONN is about to be queued 2. An IP allocate request succeeds following previous suppression of queuing	“XISQUE exit for managing IPIC intersystem queues” on page 269
XLDELETE	Loader domain	After an instance of a program is released by CICS and just before the program is freed from storage.	“Exit XLDELETE” on page 159
XLDLOAD	Loader domain	After an instance of a program is brought into storage, and before the program is made available for use.	“Exit XLDLOAD” on page 158

Table 1. Alphabetical list of global user exit points (continued)

Exit name	Module or domain	Where or when invoked	Topic
XLGSTRM	Log manager domain	After the CICS log manager detects that a log stream does not exist, and before calling the MVS™ system logger to define the log stream.	"Log manager domain exit XLGSTRM" on page 160
XMEOUT	Message domain	Before a message is sent from the message domain to its destination.	"Exit XMEOUT" on page 165
XMNOUT	Monitoring domain	Before a record is either written to SMF or buffered before a write to SMF.	"Exit XMNOUT" on page 168
XNQEREQ	Enqueue EXEC interface program	Before CICS processes an enqueue API request.	"Exit XNQEREQ" on page 44
XNQEREQC	Enqueue EXEC interface program	After an enqueue API request has completed.	"Exit XNQEREQC" on page 45
XPCABND	Program control program	After a transaction abend and before a dump call is made.	"Exit XPCABND" on page 201
XPCERES	Program control program	Before CICS processes a program link or Link3270 bridge request that has been dynamically routed to this region, where the routing region supports the "resource unavailable" (RESUNAVAIL) condition.	"Exit XPCERES" on page 187
XPCFTCH	Program control program	Before an application program is given control.	"Exit XPCFTCH" on page 195
XPCHAIR	Program control program	Before a HANDLE ABEND routine is given control.	"Exit XPCHAIR" on page 197
XPCREQ	Program control program	Before a LINK request is processed.	"Exit XPCREQ" on page 185
XPCREQC	Program control program	After a LINK request has been completed.	"Exit XPCREQC" on page 188
XPCTA	Program control program	After an abend occurs and before the environment is modified.	"Exit XPCTA" on page 198
XRCINIT	User log record recovery program	During warm and emergency restart, if user recovery log records are detected in the CICS system log.	"Exit XRCINIT" on page 258
XRCINPT	User log record recovery program	During warm and emergency restart, for each user recovery log record found in the CICS system log.	"Exit XRCINPT" on page 259
XRMIIN	Resource manager interface program	Before an EXEC DLI, EXEC SQL, or RMI command runs.	"Exit XRMIIN" on page 202
XRMIOUT	Resource manager interface program	After an EXEC DLI, EXEC SQL, or RMI command runs.	"Exit XRMIIN" on page 202
XRSINDI	Resource management modules	Immediately after successfully installing or discarding a resource.	"Resource management installation and discard exit XRSINDI" on page 204
XSNEX	Security manager domain	Restore old CICS sign-on and sign-off behavior (pre-CICS TS 2.1)	"Exit XSNEX" on page 213
XSNOFF	Security manager domain	After a terminal user signs off.	"Exit XSNOFF" on page 213

Table 1. Alphabetical list of global user exit points (continued)

Exit name	Module or domain	Where or when invoked	Topic
XSNON	Security manager domain	After a terminal user signs on.	"Exit XSNON" on page 211
XSRAB	System recovery program	When the system recovery program finds a match for an MVS abend code in the SRT.	"System recovery program exit XSRAB" on page 216
XSTERM	System termination program	During a normal system shutdown, immediately before TD buffers are cleared.	"System termination program exit XSTERM" on page 219
XSTOUT	Statistics domain	Before a statistics record is written to SMF.	"Exit XSTOUT" on page 215
XSZARQ	Front End Programming Interface	After a FEPI request has completed.	"Front End Programming Interface exits XSZARQ and XSZBRQ" on page 55
XSZBRQ	Front End Programming Interface	Before a FEPI request is actioned.	"Front End Programming Interface exits XSZARQ and XSZBRQ" on page 55
XTCATT	Terminal control program	Before task attach.	"Exit XTCATT" on page 237
XTCIN	Terminal control program	After an input event.	"Exit XTCIN" on page 236
XTCOUT	Terminal control program	Before an output event.	"Exit XTCOUT" on page 237
XTDEREQ	Transient data EXEC interface program	Before CICS processes a transient data API request.	"Exit XTDEREQ" on page 249
XTDEREQC	Transient data EXEC interface program	After a transient data API request has completed.	"Exit XTDEREQC" on page 250
XTDIN	Transient data program	After receiving data from QSAM (extrapartition) or VSAM (intrapartition).	"Exit XTDIN" on page 247
XTDOUT	Transient data program	Before passing data to a QSAM (extrapartition) or VSAM (intrapartition) user-defined transient data queue.	"Exit XTDOUT" on page 247
XTDREQ	Transient data program	Before request analysis.	"Exit XTDREQ" on page 246
XTSEREQ	Temporary storage EXEC interface program	Before CICS processes a temporary storage API request.	"Exit XTSEREQ" on page 226
XTSEREQC	Temporary storage EXEC interface program	After a temporary storage API request has completed.	"Exit XTSEREQC" on page 227

Table 1. Alphabetical list of global user exit points (continued)

Exit name	Module or domain	Where or when invoked	Topic
XTSPTIN	Temporary storage domain	Before invocation of a TSPT function.	"Exit XTSPTIN" on page 223
XTSPTOUT	Temporary storage domain	After invocation of a TSPT function.	"Exit XTSPTOUT" on page 224
XTSQRIN	Temporary storage domain	Before invocation of a TSQR function.	"Exit XTSQRIN" on page 220
XTSQROUT	Temporary storage domain	After invocation of a TSQR function.	"Exit XTSQROUT" on page 222
XWBAUTH	web domain	During processing of an **EXEC CICS WEB SEND** or **EXEC CICS WEB CONVERSE** command.	"HTTP client open and send exits: XWBAUTH, XWBOPEN and XWBSNDO" on page 124
XWBOPEN	web domain	During processing of an **EXEC CICS WEB OPEN** or **EXEC CICS INVOKE SERVICE** command.	"HTTP client open and send exits: XWBAUTH, XWBOPEN and XWBSNDO" on page 124
XWBSNDO	web domain	During processing of an **EXEC CICS WEB SEND** or **EXEC CICS WEB CONVERSE** command.	"HTTP client open and send exits: XWBAUTH, XWBOPEN and XWBSNDO" on page 124
XWSPRROI	Pipeline domain	After any instance of the XWSPRRWI exit is invoked and before the web services provider business application.	"Exit XWSPRROI" on page 172
XWSPRROO	Pipeline domain	After the web service provider application returns and before CICS creates the body of the response message.	"Exit XWSPRROO" on page 173
XWSPRRWI	Pipeline domain	After CICS has converted the web services request body into a language structure and before any instance of the XWSPRROI exit is invoked.	"Exit XWSPRRWI" on page 171
XWSPRRWO	Pipeline domain	After any instance of the XWSPRROO exit and before CICS creates the body of the response message.	"Exit XWSPRRWO" on page 174
XWSRQROI	Pipeline domain	After CICS has processed the outbound web service response and before any instance of the XWSRQRWI exit.	"Exit XWSRQROI" on page 178
XWSRQROO	Pipeline domain	After any instance of the XWSRQRWO exit has been processed and before the data flows outbound on the web services transport.	"Exit XWSRQROO" on page 177
XWSRQRWI	Pipeline domain	After CICS has processed the outbound web service response and after any instance of the XWSRQROI exit.	"Exit XWSRQRWI" on page 179

Table 1. Alphabetical list of global user exit points (continued)

Exit name	Module or domain	Where or when invoked	Topic
XWSRQRWO	Pipeline domain	After CICS has converted the application's language structure into a Web services request body and before CICS processes the optional XWSRQROO exit point.	"Exit XWSRQRWO" on page 176
XWSSRROI	Pipeline domain	After CICS has processed the outbound web service response and before any instance of the XWSSRRWI exit.	"Exit XWSSRROI" on page 182
XWSSRROO	Pipeline domain	After any instance of the XWSSRRWO exit has been processed and before the encryption of data flowing outbound on the web services transport.	"Exit XWSSRROO" on page 181
XWSSRRWI	Pipeline domain	After CICS has processed the outbound web service response and after any instance of the XWSSRROI exit.	"Exit XWSSRRWI" on page 183
XWSSRRWO	Pipeline domain	After CICS has converted the application's language structure into a Web services request body and before CICS processes the optional XWSSRRWO exit point, and before being encrypted by the pipeline's security handler.	"Exit XWSSRRWO" on page 180
XXDFA	DBCTL interface control program	In the active CICS when CICS-DBCTL connection fails.	"DBCTL interface control program exit (XXDFA)" on page 22
XXDFB	DBCTL tracking program	In the alternate CICS when DBCTL fails.	"Exit XXDFB" on page 23
XXDTO	DBCTL tracking program	In the alternate CICS when active DBCTL fails.	"Exit XXDTO" on page 23
XXMATT	Transaction manager domain	When a user transaction is attached.	"Transaction manager domain exit XXMATT" on page 244
XXRSTAT	XRF request processing program	After a z/OS® Communications Server failure or a predatory takeover.	"Exit XXRSTAT" on page 275
XZCATT	z/OS Communications Server terminal management program	Before task attach.	"SNA LU management program exit (XZCATT)" on page 260
XZCIN	z/OS Communications Server working set module	After an input event.	"Exit XZCIN" on page 260
XZCOUT	z/OS Communications Server working set module	Before an output event.	"Exit XZCOUT" on page 261
XZCOUT1	z/OS Communications Server working set module	Before a message is broken into RUs.	"Exit XZCOUT1" on page 262

Table 1. Alphabetical list of global user exit points (continued)

Exit name	Module or domain	Where or when invoked	Topic
XZIQUE	z/OS Communications Server working set module	1. When an allocate request for a session is about to be queued. 2. When an allocate request succeeds following previous suppression of queuing.	"XZIQUE exit for managing MRO and APPC intersystem queues" on page 262

Activity keypoint program exit (XAKUSER)

The XAKUSER exit is invoked during the activity keypointing process. You can use this exit to record, on the system log, user data that must be restored following an emergency restart.

For best performance, journal control requests should not specify WAIT. CICS will force the records by writing a synchronous end of keypoint record upon return from the exit program.

Your exit program should be translated with the NOEDF option. Any program it links to should also be translated with this option. It is not possible to link to programs written in PL/I.

To ensure that it is called during every keypoint, your exit program should be enabled by means of a first phase PLTPI program - see Writing initialization programs. However, if it enabled at this stage, your program should not attempt to link to any program coded in COBOL or C, as it might be invoked before these are initialized.

Note: Your exit program forms part of a critical CICS system activity. If it fails, CICS terminates. Only the listed EXEC CICS commands are allowed in the XAKUSER exit. The exit should link only to other programs with the same restrictions.

Exit XAKUSER

When invoked

During the activity keypointing process.

Exit-specific parameters

UEPAKTYP

Address of a 1-byte field indicating the type of keypoint for which the exit is invoked. The possible values are:

UEPAKPER

Activity keypoint

UEPAKWSD

Warm shutdown keypoint.

Return codes

UERCNORM

Continue processing.

XPI calls

XPI must not be used.

API and SPI calls

The following commands are supported:

- ADDRESS CWA
- ADDRESS EIB
- LINK (but only to local programs; distributed program links may not be used).
- RETURN
- WRITE JOURNALNAME.

Application association data exit in the AP domain (XAPADMGR)

Use the XAPADMGR exit for distributed transactions. XAPADMGR allows you to add user information to the association data of a task, at the point of origin of the distributed transaction. This information could be used later, for example, as a search key for processing carried out through CICSPlex® SM.

The exit program is called, if enabled, at the attach of nonsystem tasks for which no input Origin Descriptor Record is provided.

On input, the exit program is passed the association data of the task. The exit might find other relevant information, for inclusion in the association data, from other sources, using CICS commands.

Note: Distributed transactions that use DPL over IPIC connections pass their transaction group ID and origin data, including the user correlator, to be inherited by the mirror task in the target region.

The exit program could perform other activities, such as logging of information found in the association data, for purposes such as auditing or accounting of workloads. For more information on association data and origin data, see Association data.

Exit XAPADMGR

When called

At the attach of a nonsystem task that has no inherited association data passed to it.

Exit-specific parameters

UEPADCB

Address of the selectable association data control block. This is mapped by the DFHMNADS DSECT.

UEPADCBL

Length, in bytes, of the association data control block.

UEPUCD

Address of a 64-byte output area in which the exit program can place the user correlation data.

This area will be cleared to zeros by CICS before invoking the first exit program that is active in the XAPADMGR user exit point, but will not be reset by CICS between programs if multiple exit programs are active. The same storage area will be passed to subsequent exit programs that are active in the same exit point.

Return codes

UERCNORM
Continue processing.

XPI calls
All can be used.

API and SPI calls
All can be used, except for:
- EXEC CICS ABEND
- EXEC CICS PERFORM SHUTDOWN

Sample exit program

DFH$APAD

Basic mapping support exits (XBMIN, XBMOUT)

Two basic mapping support exits are provided: XBMIN and XBMOUT. The XBMIN exit allows you to intercept a RECEIVE MAP request after BMS has successfully processed the request. The XBMOUT exit allows you to intercept a SEND MAP request after BMS has successfully processed the request; or, if cumulative mapping is in progress, on completion of each page of output.

The XBMIN exit is called, if enabled, when all the following are true:
- A RECEIVE MAP command has been successfully processed.
- The map referenced in the command contains at least one field specified as VALIDN=USEREXIT.
- At least one USEREXIT field has been returned in the inbound datastream and has been mapped into the application data structure.

Using XBMIN, you can:
- Analyze each field defined as VALIDN=USEREXIT mapped to the application on this request
- Use the mapset name, map name, and field length defined in the map, and the actual length of field data returned in the inbound datastream
- Modify the data in each field.

The XBMOUT exit is called, if enabled, when all the following are true:
- A SEND MAP command has been successfully processed.
- The map referenced in the command contains at least one field specified as VALIDN=USEREXIT.
- At least one USEREXIT field has been generated in the outbound datastream.

Using XBMOUT, you can:
- Analyze each field defined as VALIDN=USEREXIT which has been generated in the outbound datastream
- Use the mapset name, map name, and field length defined in the map, and the actual length of field data placed in the outbound datastream
- Modify the data in each field
- Modify the attributes sent with each field.

Both exits are passed four exit-specific parameters:
1. The address of the TCTTE associated with the mapping request

2. The address of the system EIB associated with the task issuing the mapping request
3. The address of a halfword binary count of the number of elements in the *field element table*
4. The address of the field element table.

Sample program, DFH$BMXT

CICS supplies a sample program, DFH$BMXT, that shows how mapped input and output data can be modified with reference to the information provided in the "field element" table. A copybook, DFHXBMDS, is also supplied. This copybook is a DSECT which defines the structure of the field element.

Exit XBMIN

This exit is invoked after basic mapping support (BMS) has successfully processed an input mapping operation.

When invoked
After BMS has successfully processed an input mapping operation.

Exit-specific parameters

UEPBMTCT
Address of the TCTTE associated with the mapping request.

UEPEXECB
Address of the system EIB associated with the task.

UEPBMCNT
Address of the halfword binary number of "field elements" in the field element table.

UEPBMTAB
Address of the field element table.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

Exit XBMOUT

This exit is invoked after basic mapping support (BMS) has successfully completed a page of output during an output mapping operation.

When invoked
After BMS has successfully completed a page of output during an output mapping operation.

Exit-specific parameters

UEPBMTCT
Address of the TCTTE associated with the mapping request.

UEPEXECB
Address of the system EIB associated with the task.

UEPBMCNT
Address of the halfword binary number of "field elements" in the field element table.

UEPBMTAB
Address of the field element table.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

The field element table structure

The *field element table* contains one or more elements which provide information about each "field of interest" passed to the exit.

A "field of interest" is a field which has been defined as VALIDN=USEREXIT in the map source file used to create the mapset referenced in the mapping operation.

Each field element has the following structure:

BMXMAPST
is an 8-byte area which contains the name of the mapset associated with this field. If terminal or alternate suffixes are used with mapset names in your CICS installation, the mapset name may have a suffix appended to the name specified in the mapping request.

BMXMAP
is a 7-byte area which contains the name of the map associated with this field.

BMXFDFB
is a one-byte field copied from the field specification in the map load module. It contains indicators as follows:

X'80' CASE=MIXED

X'40' Group field entry

X'20' Group field descriptor

X'10' ATTRB=DET

X'08' JUSTIFY=ZERO

X'04' JUSTIFY=RIGHT

X'02' INITIAL,XINIT, or GINIT specified

X'01' Named field (DSECT entry exists)

BMXMAPLN
is a halfword binary value which contains the field length defined in the LENGTH option of the DFHMDF macro.

BMXACTLN
is a halfword binary value which contains the actual length of the data received or transmitted in this field.

BMXDATA
is the address of the field data.

In the XBMIN exit, BMXDATA points into a work area which BMS has obtained for input mapping purposes. When the exit returns control, this work area is copied to the application data structure associated with this map.

In the XBMOUT exit, BMXDATA points into a terminal input/output area (TIOA) in which BMS has generated an output datastream. When the exit returns control, the TIOA is disposed of in accordance with the disposition of the TERMINAL (the default), SET, or PAGING option specified on the SEND MAP request.

BMXATTR
is only relevant in the XBMOUT exit. It is the address of the attributes (if any) which BMS has placed in the output datastream preceding this field.

BMXMAPOF
is the offset of the field in the map. For example, if a map is defined as

```
MYMAP  DFHMDI SIZE=(12,40)
```

and a field in this map is defined as

```
FLDA   DFHMDF POS=(5,1)
```

the offset of this field (relative to zero) is 160 in decimal notation. In this example, BMXMAPOF would contain the value X'00A0'.

BMXBUF
is the offset of the field in the device buffer. Usually—that is, when the map dimensions are the same as the current screensize in use by the device—this value will be the same as that of BMXMAPOF. However, using the example given in the BMXMAPOF description, if MYMAP is sent to a device currently using a 24 by 80 screensize, the offset of the field in the device buffer (again relative to zero) is 320 in decimal notation. In this example, BMXBUF would contain the value X'0140'.

Programming the XBMIN exit

When programming the XBMIN exit it is important to consider data length.

The actual data length (in BMXACTLN) might be less than the length defined in the map (in BMXMAPLN). This could happen, for example, if a terminal operator does not completely fill a data entry field. In this case, BMS will have right- or left-justified the data in the field and padded the field with blank or zero characters. This justification and padding occurs before the exit is invoked. Your exit program can, by checking the bit settings in the BMXFDFB field, determine how BMS performed justification and padding for the field.

The actual data length (in BMXACTLN) might be greater than the length defined in the map (in BMXMAPLN). This could happen, for example, if a map contains an unprotected field which is not immediately followed by another field. This allows the terminal operator to enter data past the end of the field. When this occurs, the data field is truncated by BMS according to the length defined for the field in the map. However, BMXACTLN contains the length of data found in the inbound datastream.

When modifying data in the XBMIN exit, the safest method is to use the length provided in BMXMAPLN, but to ensure that any pad characters added by BMS are preserved.

BMXATTR must be ignored in the XBMIN exit; it always contains binary zeroes.

Programming the XBMOUT exit

When programming the XBMOUT exit it is important to consider the actual data length.

The actual data length (in BMXACTLN) may be less than the length defined in the map (in BMXMAPLN). This occurs due to the compression of trailing nulls performed by BMS for each output field.

The actual length of data cannot be changed in the exit program. The exit is invoked after the output datastream has been generated; consequently, an attempt to alter the data length could result in an invalid datastream. Therefore, if an XBMOUT exit program modifies data, it must do so with reference to the length value in BMXACTLN.

BMXDATA may contain a null value. This can be caused by a SEND MAP request with the MAPONLY option when the map has fields without default data; this causes BMS to send an attribute sequence for the field but no data.

BMXATTR may contain a null value. This can be caused by a SEND MAP request with the DATAONLY option, when the application is updating the data in a field and not the attributes.

Cumulative mapping operations

When an application is performing cumulative mapping—that is, issuing a sequence of SEND MAP commands with the ACCUM option—BMS builds composite display in which a single page of output might be constructed from multiple SEND MAP requests.

When cumulative mapping occurs, the XBMOUT exit is called when a page has been built, not as each SEND MAP request is processed.

Message routing

When an application builds a routing message—for example, it issues a ROUTE command followed by one or more SEND MAP commands with the SET or PAGING option specified—the XBMOUT exit is invoked in the same way as for a non-routed mapping request.

However, the UEPBMTCT parameter is passed as a null value for a routed message. This is because a routed message may be destined for multiple devices, and BMS has optimized the features supported by the devices targeted by the routed message. When processing a routed message in the XBMOUT exit, referencing the TCTTE for any of these devices would probably not be relevant.

Bridge facility exit XFAINTU

The bridge facility exit is called just after a new bridge facility has been built and just before the bridge facility is deleted.

The bridge facility might be deleted at the end of a task, when zero keep time is specified, or when a keep time expires before the facility is reused.

Exit XFAINTU

When invoked

Just after a bridge facility is created and just before it is deleted.

Exit-specific parameters

UEPFAREQ

Address of a 1-byte field that indicates why the exit has been called. Possible values are:

UEPFAIN

Initialization.

UEPFATU

Tidy-up.

UEPFATUT

Address of a 1-byte field that indicates the type of tidy-up required. Possible values are:

UEPFANTU

Normal tidy-up.

UEPFAETU

Expired tidy-up.

UEPFANAM

Address of the bridge facility name.

UEPFATYP

Address of a 1-byte field that indicates the facility type. The value is always:

UEPFABR

3270 bridge facility.

UEPFAUAA

Address of the bridge facility user area (TCTUA).

UEPFAUAL

Address of a one-byte field containing the length of the bridge facility user area.

UEPFATK

Address of the 8–byte facilitytoken.

UEPFAMCH

Address of a 1-byte field that indicates the mechanism used to start the bridged transaction using this bridge facility. Possible values are:

UEPFASTA

Started using START BREXIT.

UEPFALNK

Started using a link to DFHL3270.

UEPFAREG

Address of a 1-byte field that indicates whether the region owns the bridge facility, or whether it is remote. A bridge facility is owned by the AOR, where it is local, and is remote to the router region. Note that XFAINTU can be called twice in the same region if the AOR and the router are the same region. Possible values are:

UEPFAROU
This region is the router for this bridge facility.

UEPFAAOR
This region is the AOR for this bridge facility.

Return codes

UERCNORM
Continue processing.

XPI calls
All can be used, except those that use recoverable resources.

API calls
All can be used except those that invoke task-related user exits or use recoverable resources.

Data tables management exits XDTRD, XDTAD, and XDTLC

Data tables management exits apply to both CICS shared data tables and CICS coupling facility data tables.

XDTRD and XDTAD allow you to control the selection of records for inclusion in a data table, XDTRD being used to make such selections during loading, and XDTAD being invoked when records are subsequently added to a loaded data table (or to a CFDT that did not require loading). XDTRD also allows the contents of records that are included in a user-maintained table, or a coupling facility data table, to be modified before they are added.

For CICS shared data tables, XDTLC enables you to take action based on the fact that a data table has completed loading, which might be to end some restrictions that you have decided to place on access to the data table during loading, or to cater for an unsuccessful completion of the loading.

For a coupling facility data table, XDTLC allows your global user exit program to decide whether to accept an unsuccessfully loaded coupling facility data table. If the user exit program decides to accept the table, it remains open and available for access, but CICS does not mark it as loading completed. This is also the default action if no XDTLC exit is enabled. This means that application programs continue to receive the LOADING condition for any records that are beyond the key range of records successfully loaded into the table. This ensures that application programs are aware that not all the expected data is available. It also allows you to retry the load, when the cause of the failure has been corrected, by closing the file that initiated the load and reopening it. Alternatively, you could open another load-capable file that refers to the same data table. If your exit program decides to reject the table, it is closed and the records already loaded remain in the table. If the cause of the failure is corrected, a subsequent open for the data table allows the load to complete. XDTLC is not invoked for a coupling facility data table that is not loaded from a source data set.

Note that a program invoked from any of these exit points must declare a DSECT defining the data tables user exit parameter list pointed to by field UEPDTPL. (Although UEPDTPL is defined by a DFHUEXIT call, the parameter list that it addresses is not.) To do this, your program can include the copybook DFHXDTDS, which defines the DT_UE_PLIST DSECT.

If any tables specify OPENTIME=STARTUP or are opened implicitly, you should provide a program list table post-initialization (PLTPI) program to activate the user exits. Otherwise, the data table might start loading before the exits can be enabled. For more details about PLTPI programs, see Writing initialization and shutdown programs.

Note: For additional information about using these exits with CICS shared data table support, see Shared data tables overview.

Exit XDTRD

The XDTRD user exit is invoked just before CICS attempts to add to the data table a record that has been retrieved from the source data set.

This normally occurs when the loading process retrieves a record during the sequential copying of the source data set. However, it can also occur when an application retrieves a record that is not in the data table and:

- For a user-maintained data table, loading is still in progress, or
- For a CICS-maintained data table, loading terminated before the end of the source data set was reached (because, for example, the data table was full).

Note: For a coupling facility data table the XDTRD exit is invoked only for a table that is loaded from a source data set.

The record retrieved from the source data set is passed as a parameter to the user exit program—see fields UEPDTRA and UEPDTRL. Your exit program can choose (depending, for example, on the key value—see fields UEPDTKA and UEPDTKL) whether to include the record in the data table or not.

Alternatively, the exit program can request that all subsequent records up to a specified key are skipped—see field UEPDTSKA; these records are not passed to the exit program. This facility is available only during loading. You can specify the key as a complete key, or you can specify just the leading characters by padding the skip-key area with binary zeros.

For a user-maintained data table, the program can also modify the data in the record to reduce the storage needed for the data table. Application programs that use the data table must be aware of any changes made to the record format by the exit program. If the record length is changed, the exit program must set the new length in the parameter list—see field UEPDTRL. The new length must not exceed the data buffer length—see field UEPDTRBL.

When invoked

Just before CICS tries to add to the data table a record that has been retrieved from the source data set.

Exit-specific parameters

UEPDTPL

Address of the data table user exit parameter list, which is mapped by DSECT DT_UE_PLIST in copybook DFHXDTDS. The data table user exit parameter list contains:

UEPDTNAM

The 8-character data table name.

UEPDTFLG

A 1-byte flag field. The possible bit settings are:

UEPDTSDT (X'80')
The exit has been invoked by CICS shared data table support.

UEPDTCMT (X'40')
This is a CICS-maintained table. Only meaningful if UEPDTSDT is on.

UEPDTOPT (X'20')
The exit has been invoked for table loading. This means that optimization by skipping can be requested.

UEPDTCFT(X'10')
The exit has been invoked by coupling facility data table support.

UEPDTUMT (X'08')
This is a user-maintained table. Only meaningful if UEPDTSDT is on.

UEPDTRA
The address of the data record.

UEPDTRBL
The fullword length of the data table buffer.

UEPDTRL
The fullword length of the data record.

For user-maintained tables, the exit program can set a new length in this field, if it amends the record.

UEPDTKA
The address of the data table key.

UEPDTKL
The fullword length of the data table key.

UEPDTDSL
The fullword length of the name of the source data set. Only meaningful if either UEPDTSDT or UEPDTCFT is on.

UEPDTDSN
A 44-character field containing the name of the source data set. Only meaningful if either UEPDTSDT or UEPDTCFT is on.

UEPDTSKA
The address of a skip-key area. When invoked for table loading, your exit program can return a key of length UEPDTKL in this area, and request load optimization by setting a return code of UERCDTOP. Only meaningful if either UEPDTSDT or UEPDTCFT is on.

Return codes

UERCDTAC
Add the record to the data table.

UERCDTRJ
Reject the record: that is, do not add it to the table.

UERCDTOP
Skip this and the following records until a key is found that is equal to or greater than the key specified in the skip-key area. Only meaningful if either UEPDTSDT or UEPDTCFT is on.

XPI calls
All can be used.

Exit XDTAD

Exit XDTAD is invoked when a write request is issued to a data table.

For a user-maintained data table and coupling facility data table, the user exit is invoked once - before the record is added to the data table. For a CICS-maintained data table, the user exit is invoked twice - before the record is added to the source data set and then again before the record is added to the data table.

The record written by the application is passed as a parameter to the user exit program - see fields UEPDTRA and UEPDTRL. Your exit program can choose (depending on the key value, for example see fields UEPDTKA and UEPDTKL) whether to include the record in the data table or not. This decision is indicated by setting the return code.

The XDTAD exit must not modify the data in the record. If you used XDTRD to truncate the data records when the data table was loaded, you must code your application so that it only tries to write records of the correct format for the data table.

A sample XDTAD exit program is listed in Shared data tables overview.

When invoked
One or more times during the processing of a write request to a data table.

Exit-specific parameters

UEPDTPL
Address of the data table user exit parameter list, which is mapped by DSECT DT_UE_PLIST in copybook DFHXDTDS. The data table user exit parameter list contains:

UEPDTNAM
The 8-character data table name.

UEPDTFLG
A 1-byte flag field. The possible bit settings are:

UEPDTSDT (X'80')
The exit has been invoked by CICS shared data table support.

UEPDTCMT (X'40')
This is a CICS-maintained table. Only meaningful if UEPDTSDT is on.

UEPDTCFT(X'10')
The exit has been invoked by coupling facility data table support.

UEPDTUMT (X'08')
This is a user-maintained table. Only meaningful if UEPDTSDT is on.

UEPDTRA
The address of the data record.

UEPDTRBL
The fullword length of the data table buffer.

UEPDTRL
The fullword length of the data record.

UEPDTKA
The address of the data table key.

UEPDTKL
The fullword length of the data table key.

UEPDTDSL
The fullword length of the name of the source data set. Only meaningful if either UEPDTSDT or UEPDTCFT is on.

UEPDTDSN
A 44-character field containing the name of the source data set. Only meaningful if either UEPDTSDT or UEPDTCFT is on.

Return codes

UERCDTAC
Add the record to the data table.

UERCDTRJ
Reject the record: that is, do not add it to the table.

XPI calls
All can be used.

Exit XDTLC

The XDTLC user exit is invoked at the completion of data table loading—whether successful or not. The user exit is not invoked if the data table is closed for any reason before loading is complete. The XDTLC exit is invoked for a coupling facility data table only if the table is loaded from a source data set.

The exit program is informed if the loading did not complete successfully—see field UEPDTORC. This could occur, for example, if the maximum number of records was reached or there was insufficient virtual storage. In this case, the exit program can request that the file is closed immediately, by setting the return code.

When invoked
At the completion of table loading. It is not invoked if the loading process was terminated because the data table had been closed.

Exit-specific parameters

UEPDTPL
Address of the data table user exit parameter list, which is mapped by DSECT DT_UE_PLIST in copybook DFHXDTDS. The data table user exit parameter list contains:

UEPDTNAM
The 8-character data table name.

UEPDTFLG
A 1-byte flag field. The possible bit settings are:

UEPDTSDT (X'80')
The exit has been invoked by CICS shared data table support.

UEPDTCMT (X'40')
This is a CICS-maintained table. Only meaningful if UEPDTSDT is on.

UEPDTCFT(X'10')
The exit has been invoked by coupling facility data table support.

UEPDTUMT (X'08')
This is a user-maintained table. Only meaningful if UEPDTSDT is on.

UEPDTORC
Data table open result code. The possible values are:

UEPDTLCS
Load successful

UEPDTLFL
Load unsuccessful.

UEPDTDSL
The fullword length of the name of the source data set. Only meaningful if either UEPDTSDT or UEPDTCFT is on.

UEPDTDSN
A 44-character field containing the name of the source data set. Only meaningful if either UEPDTSDT or UEPDTCFT is on.

Return codes

UERCDTOK
Accept the data table in its present state

UERCDTCL
Close the data table.

XPI calls
All can be used.

DBCTL interface control program exit (XXDFA)

This exit is invoked by an active CICS if its connection to DBCTL fails.

When invoked
By an active CICS when its connection to DBCTL fails. Your exit program is invoked after the active CICS has informed the alternate CICS of the failure.

Exit-specific parameters

UEPDBXR
Address of CICS XRF information for use with DBCTL. The CICS XRF information can be mapped using the DSECT DFHDXUEP.

Return codes

UERCNOAC
Take no action.

UERCSWCH
Switch to the alternate DBCTL.

UERCABNO
Abend CICS without a dump.

UERCABDU
Abend CICS with a dump.

XPI calls
TRANSACTION_DUMP must not be used.

DBCTL tracking program exits (XXDFB, XXDTO)

These exits are invoked if the connection to DBCTL fails, or if CICS performs takeover.

Exit XXDFB

The XXDFB exit is invoked when a message is received from the active CICS indicating that the connection to DBCTL failed.

When invoked
By the alternate CICS when it receives a message from the active CICS indicating that connection to DBCTL has failed. The alternate and active CICS systems are running in different MVS images, perhaps in different central processing complexes (CPCs). More information about these exits, see Overview of Database Control (DBCTL).

Exit-specific parameters

UEPDBXR
Address of CICS XRF information for use with DBCTL. The CICS XRF information can be mapped using the DSECT DFHDXUEP.

Return codes

UERCNOAC
Take no action.

UERCSWCH
Switch to the alternate DBCTL.

UERCABNO
Abend CICS without a dump.

UERCABDU
Abend CICS with a dump.

The return code 'UERCNORM' is not available for use at this exit point.

XPI calls
The following must not be used:

- INQUIRE_MONITORING_DATA
- MONITOR
- TRANSACTION_DUMP
- WRITE_JOURNAL_DATA.

Exit XXDTO

Exit XXDTO is invoked by an alternate CICS when it performs takeover.

When invoked

By an alternate CICS when it performs takeover under the following conditions:

- The active and alternate CICS systems are in different MVS images, perhaps in different processors.
- The active CICS was connected to, or trying to connect to, a DBCTL subsystem. (This does not include disconnecting from one DBCTL and reconnecting to another.)
- The takeover was not initiated by the XXDFB exit, or the takeover was initiated by XXDFB but the active system reestablished a DBCTL connection before takeover occurred and XXDTO was driven for a new DBCTL takeover decision.

Exit-specific parameters

UEPDBXR
Address of CICS XRF information for use with DBCTL. The CICS XRF information can be mapped using the DSECT DFHDXUEP.

Return codes

UERCNOAC
Take no action.

UERCSWCH
Switch to the alternate DBCTL.

UERCABNO
Abend CICS without a dump.

UERCABDU
Abend CICS with a dump.

The return code UERCNORM is not available for use at this exit point.

XPI calls

The following must not be used:

- INQUIRE_MONITORING_DATA
- MONITOR
- TRANSACTION_DUMP
- WRITE_JOURNAL_DATA.

Dispatcher domain exits XDSBWT and XDSAWT

The XDSBWT and XDSAWT exit points are located before and after the operating system wait. You cannot use CICS services in an exit program that is invoked from these exit points.

The XDSBWT and XDSAWT exits can be used to control the swapping state of the CICS address space. However, if the default state of the address space is non-swappable, you cannot use these exits to override this state.

CICS uses a counter that is incremented for every SYSEVENT DONTSWAP request and decremented for every SYSEVENT OKSWAP request down to a minimum of 0. A SYSEVENT DONTSWAP request is issued when this counter goes up from 0 to 1. A SYSEVENT OKSWAP request is issued when this counter goes down from 1 to 0. In all other circumstances, the SYSEVENT is not issued.

Exit XDSBWT

This exit is invoked before an operating system wait issued by the quasi-reentrant CICS TCB.

When invoked

Before an operating system wait issued by the quasi-reentrant CICS TCB.

Exit-specific parameters

None.

Return codes

UERCNORM

Continue processing.

UERCSWAP

Issue SYSEVENT to allow address space swapping.

XPI calls

Must not be used.

Exit XDSAWT

This exit is invoked after an operating system wait issued by the quasi-reentrant CICS TCB.

When invoked

After an operating system wait issued by the quasi-reentrant CICS TCB.

Exit-specific parameters

UEPSYSRC

Address of the 4-byte return code from the SYSEVENT request made before the operating system wait. This return code will be in one of two different forms:

1. The SYSEVENT OKSWAP return code, or
2. If the SYSEVENT request was rejected by CICS, a special CICS return code which will take one of the following decimal values:

   17 The SYSEVENT OKSWAP was not issued. The outstanding count of SYSEVENT OKSWAP requests exceeds the count of SYSEVENT DONTSWAP requests. Before a SYSEVENT OKSWAP can be issued, a SYSEVENT DONTSWAP must be requested.

   19 The SYSEVENT OKSWAP was not issued. The outstanding count of SYSEVENT DONTSWAP requests still exceeds the count of SYSEVENT OKSWAPs. Further SYSEVENT OKSWAPs must be requested before a SYSEVENT OKSWAP is issued by CICS.

Return codes

UERCNORM

Continue processing.

UERCNOSW

Issue SYSEVENT to suppress address-space swapping.

XPI calls

Must not be used.

DL/I interface program exits XDLIPRE and XDLIPOST

The XDLIPRE and XDLIPOST exit points are invoked following the issue of an **EXEC DLI** command or DL/I call. Exit XDLIPRE is invoked before the request is processed and XDLIPOST is invoked after the request is processed.

When the request is function shipped, the exits are invoked from both the application-owning region and the database-owning region. However, there are restrictions when they are invoked in a database-owning region:

1. The descriptions of the exits show the general format of the parameter list of the application. For detailed information about the format of the CALL-level DL/I parameter list, refer to *IMS Application programming: DL/I calls reference*.
2. For all EXEC DLI calls, the parameter list of the application is in assembler language format; that is, the value of the program language byte pointed to by UEPLANG is always UEPASM, and the parameter list pointed to by UEPAPLIST is always in assembler language format. This format is used because all EXEC DLI calls are converted into assembler-language CALL-level requests.

   An EXEC DLI online request is converted by DFHEDP into a CALL-level request for DFHDLI. IMS™ does not deal directly with EXEC-level parameter lists. The first parameter in the CALL parameter list contains the address of the parameter count. The second parameter in the CALL parameter list contains the address of the function. All other parameters are dependent on the function.
3. In an XDLIPRE exit program, you can change the PSB name and the SYSID name. Changing the name helps availability if the originally specified SYSID fails.

   You can change the SYSID in the following ways:

   - A remote value to another remote value
   - The local value to a remote value
   - A remote value to the local value.

   Changing the SYSID has an effect only if the associated PSB has a PDIR entry. The SYSID can be the local CICS (that is, the SYSIDNT specified on the CICS region) or a remote connection name. For the new SYSID to be used, the PSB name must have a PDIR entry; if it does not have a PDIR entry, the assumption is made that the local CICS is connected to DBCTL, and an attempt is made to run the IMS request there. An IMS schedule failure is handled in the same way as a failure to route to a connection that does not exist. If the SYSID is changed to either the same value as the SYSIDNT of the local CICS, or blanks (hex '40404040'), CICS attempts to run the IMS request on the local system.

Exit XDLIPRE

Exit XDLIPRE is invoked on entry to the DL/I interface program.

Programs running in this exit must be coded to threadsafe standards and defined to CICS as threadsafe.

Exit-specific parameters

UEPCTYPE
Address of type-of-request byte. Values are:

UEPCEXEC
The original request was an EXEC DLI request.

UEPCCALL
The original request was a CALL-level request.

UEPCSHIP
The request has been function shipped from another region. When this value is set, restrictions apply to the setting and use of the rest of the exit parameters, as described.

UEPAPLIST
Address of application's parameter list. The general format for COBOL and assembler language is:

```
plist address --> parm1 address --> parm1
                  parm2 address --> parm2
                  parm3 address --> parm3
                  ..............
                  up to a maximum of 18 parameters
                  excluding the optional parmcount.
```

The general format for PL/I is:

```
plist address --> parm1 address --> parm1 (parmcount)
       parm2 address --> locator descriptor --> parm2
       parm3 address --> locator descriptor --> parm3
       ..............
       up to a maximum of 18 parameters
```

When UEPCTYPE is not UEPCSHIP, your exit program can change any of the parameters in the application parameter list. For UEPCSHIP requests, your exit program **cannot** change any of the parameters. Furthermore, for UEPCSHIP requests, UEPAPLIST points to a copy of the parameter list in the above format, but which contains only the first two parameters, `parm1` and `parm2`.

Note: For PL/I applications, `parm1` may or may not contain a parameter-count. Your exit program should check this field before using it.

UEPLANG
Address of program language byte. Values are:

UEPPLI
PL/I

UEPCBL
COBOL

UEPASM
Assembler language.

For UEPCSHIP requests, the language is always assembler.

UEPIOAX
Address of I/O area existence flag byte:

UEPIOA1
I/O area exists.

For UEPCSHIP requests, the I/O area existence flag is always off.

UEPIOA
Address of I/O area. This is the application's IOAREA, or DFHEDP's IOAREA in the case of EXEC DLI. The contents of the IOAREA can be overwritten in the exit: the new contents are used

when the DL/I request is processed. However, it should be noted that IOAREAs can be in a program's static storage and, in this case, should not be overwritten.

For UEPCSHIP requests, UEPIOA is always zero.

UEPPSBNX
Address of PSB existence flag byte:

UEPPSB1
A PSB exists.

UEPPSBNM
Address of an area containing the 8-character PSB name. The contents of the area can be overwritten by the exit, for all types of request including UEPCSHIP; the new contents are used when the DL/I request is processed.

UEPSYSDX
Address of the SYSID existence flag byte:

UEPSYS1
A SYSID exists.

UEPSYSID
Address of an area containing the 4-character SYSID name. The contents of the area can be overwritten by the exit, for all types of request including UEPCSHIP; the new contents are used when the DL/I request is processed.

Return codes

UERCNORM
Continue processing

UERCBYP
Bypass DL/I request and return

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

Exit XDLIPOST

Exit XDLIPOST is invoked on exit from the DL/I interface program.

Programs running in this exit must be coded to threadsafe standards and defined to CICS as threadsafe.

Exit-specific parameters

UEPCTYPE
Address of type-of-request byte. Values are:

UEPCEXEC
An EXEC DLI request.

UEPCCALL
A CALL-level request.

UEPCSHIP
The request has been function shipped from another

region. When this value is set, restrictions apply to the setting and use of the rest of the exit parameters, as described.

UEPAPLIST
Address of application's parameter list. The general format for COBOL and assembler language is:

```
plist address --> parm1 address --> parm1
                  parm2 address --> parm2
                  parm3 address --> parm3
                  ..............
                  up to a maximum of 18 parameters
                  excluding the optional parmcount.
```

The general format for PL/I is:

```
plist address --> parm1 address --> parm1 (parmcount)
       parm2 address --> locator descriptor --> parm2
       parm3 address --> locator descriptor --> parm3
       ..............
       up to a maximum of 18 parameters.
```

When UEPCTYPE is not UEPCSHIP, your exit program can change any of the parameters in the application parameter list. For UEPCSHIP requests, your exit program **cannot** change any of the parameters. Furthermore, for UEPCSHIP requests, UEPAPLIST points to a copy of the parameter list in the previous format, but which contains only the first two parameters `parm1` and `parm2`. See also "DL/I interface program exits XDLIPRE and XDLIPOST" on page 26.

Note: For PL/I applications, `parm1` might or might not contain a parameter-count. Your exit program should check this field before using it.

UEPLANG
Address of program language byte. Its values are:

UEPPLI
PL/I

UEPCBL
COBOL

UEPASM
assembler language.

For UEPCSHIP requests, the language is always assembler.

UEPIOAX
Address of I/O area existence flag byte:

UEPIOA1
I/O area exists.

For UEPCSHIP requests, the I/O area existence flag is always off.

UEPIOA
Address of I/O area. This is the application's IOAREA, or DFHEDP's IOAREA in the case of EXEC DLI. The contents of the IOAREA can be overwritten in the exit and are returned to the application program in the new form. However, it should be noted

that the application's IOAREA could be in the program's static storage and, in this case, should not be overwritten.

For UEPCSHIP requests, UEPIOA is always zero.

UEPUIBX
Address of UIB existence flag byte:

UEPUIB1
a UIB exists.

UEPUIB
Address of the UIB, which is mapped by DFHUIB in module DFHDBCOP. The contents of the UIB can be overwritten in the exit for all types of request, including UEPCSHIP. The new contents are returned to the application or to the region that function shipped the request.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

Example use of global user exit XDLIPRE

You can use the global user exit XDLIPRE to change the PSB name that the application program has scheduled at execution time. You can also use the XDLIPRE exit to change the identity of the SYSID, enabling work to be rerouted from a SYSID that becomes unavailable to one that is available.

This section contains Product-sensitive Programming Interface information.

The following figures show an example of XDLIPRE that you can copy and modify. This example is provided for guidance only. For programming information about global user exits, see DFHZNEPI TYPE=INITIAL—specifying the default routine.

```
*********************************************************************
* This is an example for global user exit XDLIPRE                   *
*                                                                   *
* It is invoked before any DLI call being passed to                *
* the remote or DBCTL processors.                                   *
*                                                                   *
* A check is made for the presence of a PSB.                        *
* If not, a normal return is made                                   *
*                                                                   *
* If the PSB is in a predefined table, it is changed to a           *
* different value, and a normal return is made.                     *
*                                                                   *
* If not, set PSB name to blanks and normal return.                 *
*                                                                   *
* In all cases,a trace entry is written describing the action       *
* taken, using TRACE-POINT 384  (hex '0180')                        *
*                                                                   *
*********************************************************************
*                                                                   *
* The first few instructions set up the global user exit            *
* environment, identify the user exit point, prepare for the use of*
* the exit programming interface, and copy in the definitions that *
* are to be used by the XPI function.                               *
*                                                                   *
*********************************************************************
*
         DFHUEXIT TYPE=EP,ID=XDLIPRE    PROVIDE DFHUEPAR PARAMETER
*                                       LIST AND LIST OF EXITID
*                                       EQUATES
*
         DFHUEXIT TYPE=XPIENV           SET UP ENVIRONMENT FOR
*                                       EXIT PROGRAMMING INTERFACE
*                                       MUST BE ISSUED BEFORE ANY
*                                       XPI MACROS ARE ISSUED
```

Figure 1. Example of XDLIPRE user exit to change PSB names 1/6

```
*
         COPY  DFHTRPTY                DEFINE PARAMETER LIST FOR
*                                      USE BY DFHTRPTX MACRO
*
         COPY  DFHSMMCY                DEFINE PARAMETER LIST FOR
*                                      USE BY DFHSMMCX MACRO
*
*****************************************************************
*The following DSECT maps a storage area to be used as work area *
*for the information in the TRACE entry.                         *
*****************************************************************
*
DSA      DSECT                         DSECT FOR GETMAINED STORAGE
         USING DSA,R7
*
RETCODE  DS    F                       store return code
MESSAGEA DS    F                       message address for trace
MESSAGEL DS    F                       message length for trace
MESSAGE  DS    0CL37
OLDPSB   DS    CL8
MESS1    DS    CL21
NEWPSB   DS    CL8
*********************************************************************
*The next  instructions  form  the  normal  start  of a global user *
*exit program, setting the program addressing mode to 31-bit, saving*
*the calling program's registers, establishing base addressing*
*and establishing the addressing of the user exit parameter list.   *
*********************************************************************
*
DLIPR    CSECT
DLIPR    AMODE 31
*
         SAVE (14,12)                  SAVE CALLING PROGRAM'S RGSTRS
*
         LR    R11,R15                 SET UP USER EXIT PROGRAM'S
         USING DLIPR,R11               BASE REGISTER
*
         LR    R2,R1                   SET UP ADDRESSING FOR USER
         USING DFHUEPAR,R2             EXIT PARAMETER LIST -- USE
*                                      REGISTER 2 AS XPI CALLS USE
*                                      REGISTER 1
*
*********************************************************************
*Before issuing an XPI function call, set up addressing to XPI     *
*parameter list.                                                   *
*********************************************************************
*
         L     R5,UEPXSTOR             SET UP ADDRESSING FOR XPI
*                                      PARAMETER LIST
```

Figure 2. Example of XDLIPRE user exit to change PSB names 2/6

```
*********************************************************************
*  Before issuing an XPI function call, you must ensure that register*
*  13 addresses the kernel stack.                                    *
*********************************************************************
*
         L     R13,UEPSTACK             ADDRESS KERNEL STACK
*
*********************************************************************
*  Issue a GETMAIN to get storage for work area                     *
*********************************************************************
*
         USING DFHSMMC_ARG,R5           MAP PARAMETER LIST
*
         DFHSMMCX CALL,                                              X
               CLEAR,                                                X
               IN,                                                   X
               FUNCTION(GETMAIN),                                    X
               GET_LENGTH(100),                                      X
               STORAGE_CLASS(USER),                                  X
               SUSPEND(NO),                                          X
               OUT,                                                  X
               ADDRESS((R7)),                                        X
               RESPONSE(*),                                          X
               REASON(*)
*
*********************************************************************
*  SET UP THE NORMAL RETURN CODE                                    *
*********************************************************************
*
         LA    R6,UERCNORM
         ST    R6,RETCODE
*
*********************************************************************
*  See if a PSB exists                                              *
*********************************************************************
*
         L     R6,UEPPSBNX              PSB EXISTENCE FLAG
         TM    0(R6),UEPPSB1            PSB EXISTS?
         BO    PSBCALL                  YES
         MVC   MESSAGE,MESS3T           NO-MOVE MESSAGE TO DSA
         B     TRACE
*
*********************************************************************
*  See if we want to change a PSB name                              *
*********************************************************************
*
PSBCALL  EQU   *
         L     R6,UEPPSBNM              ADDRESS OF PASSED PSB NAME
         LA    R8,PSBS                  ADDRESS OF table of PSB pairs
         CLC   0(8,R6),0(R8)            SAME?
```

Figure 3. Example of XDLIPRE user exit to change PSB names 3/6

```
         BE    FOUND                   YES
         LA    R8,16(R8)               BUMP TO NEXT PAIR
         CLC   0(8,R6),0(R8)
         BE    FOUND
         LA    R8,16(R8)               BUMP TO NEXT PAIR
         CLC   0(8,R6),0(R8)
         BE    FOUND
         B     NOTFOUND                NO MATCH - END
*
*********************************************************************
*  Move new PSB name in                                             *
*********************************************************************
*
FOUND    EQU   *
         MVC   0(8,R6),8(R8)
*
*********************************************************************
*  SET UP MESSAGE BLOCK FOR TRACE ENTRY FOR CHANGED NAME            *
*********************************************************************
*
         MVC   MESS1,MESS1T            SET UP MESSAGE
         MVC   NEWPSB,8(R8)            NEW PSB NAME
         MVC   OLDPSB,0(R8)            OLD PSB NAME
         B     TRACE                   GO PUT TRACE ENTRY
*
*********************************************************************
* SET UP MESSAGE BLOCK FOR TRACE ENTRY FOR PSB NOT FOUND            *
* SETUP THE NORMAL RETURN CODE                                      *
*********************************************************************
*
NOTFOUND EQU   *
         MVC   0(8,R6),DUMMYPSB
         MVC   MESS1,MESS2T            SET UP MESSAGE
         MVC   OLDPSB,0(R6)            SUPPLIED PSB NAME
         MVC   NEWPSB,=CL8''           CLEAR FIELD
         LA    R1,UERCNORM             SET UP NORMAL RETURN CODE
         B     TRACE                   GO PUT TRACE ENTRY
*
*********************************************************************
*  Issue trace put macro                                            *
*********************************************************************
*
TRACE    EQU   *
         LA    R6,MESSAGE              STORE ADDRESS...
         ST    R6,MESSAGEA             ...INTO BLOCK DESCRIPTOR
         LA    R6,L'MESSAGE            STORE LENGTH...
         ST    R6,MESSAGEL             ...INTO BLOCK DESCRIPTOR
         LA    R8,384                  SET UP TRACE-ID
*
```

Figure 4. Example of XDLIPRE user exit to change PSB names 4/6

```
         DROP  R5                      REUSE R5 TO MAP DFHTRPT
         USING DFHTRPT_ARG,R5          XPI PARAMETER LIST
*
         DFHTRPTX CALL,                                                X
               CLEAR,                                                  X
               IN,                                                     X
               FUNCTION(TRACE_PUT),                                    X
               POINT_ID((R8)),                                         X
               DATA1(MESSAGEA,MESSAGEL),                               X
               OUT,                                                    X
               RESPONSE(*)
*
*****************************************************************
*When the rest of the exit program is complete, free the storage   *
*and return.                                                       *
*****************************************************************
*
      DROP  R5                    REUSE REGISTER 5 TO MAP DFHSMMC
      USING DFHSMMC_ARG,R5        XPI PARAMETER LIST
*
*****************************************************************
*  Issue the DFHSMMCX macro call                                  *
*  Store the return code in register 6                            *
*****************************************************************
*
          L     R6,RETCODE              PICK UP SAVED RETURN CODE
*
          DFHSMMCX CALL,                                             X
                CLEAR,                                               X
                IN,                                                  X
                FUNCTION(FREEMAIN),                                  X
                ADDRESS((R7)),                                       X
                STORAGE_CLASS(USER),                                 X
                OUT,                                                 X
                RESPONSE(*),                                         X
                REASON(*)
*
*****************************************************************
*Restore registers, set return code, and return to user exit handler*
*****************************************************************
*
          L     R13,UEPEPSA
          ST    R6,16(13)               STORE INTO R15 SLOT OF SA
          RETURN (14,12)
*
*****************************************************************
*old and new PSB names, in pairs                                  *
*****************************************************************
*
```

Figure 5. Example of XDLIPRE user exit to change PSB names 5/6

```
PSBS     EQU   *
          DC    CL8'PC3CONEW'            VALID
          DC    CL8'PC3CONE2'            VALID
          DC    CL8'PC3FRED'             INVALID
          DC    CL8'PC3CONEW'            VALID
          DC    CL8'PC3JOE'              INVALID
          DC    CL8'PC3JOEX'             INVALID
*
MESS1T    DC    CL21' HAS BEEN CHANGED TO '
MESS2T    DC    CL21' WAS NOT FOUND'
MESS3T    DC    CL37'THIS WAS NOT A DLI SCHEDULE CALL'
DUMMYPSB  DC    CL8' '
          LTORG
          END   DLIPR
```

Figure 6. Example of XDLIPRE user exit to change PSB names 6/6

Dump domain exits XDUREQ, XDUREQC, XDUCLSE, and XDUOUT

You can use dump domain exits to capture information before and after a transaction dump or system dump.

Exit XDUREQ

Exit XDUREQ is invoked immediately before a system or transaction dump is taken.

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID.

UEPUSER
Address of the 8-byte user ID.

UEPTERM
Address of the 4-byte terminal ID.

UEPPROG
Address of the 8-byte application program name, or nulls if there is no current application.

UEPDUMPC
Address of copy of the 8-byte dump code.

UEPABCDE
Address of a copy of the 8-byte Kernel error code in the format xxx/yyyy. xxx denotes the 3-digit hexadecimal MVS completion code (for example 0C1 or D37). If an MVS completion code is not applicable, xxx is three hyphens. The 4-digit code yyyy is a user abend code produced either by CICS or by another product on your system. UEPABCDE is completed only for abend codes corresponding to the following dump codes:

- AP0001
- SR0001
- ASRA
- ASRB
- ASRD

Otherwise this field contains null characters.

UEPDUMPT
Address of the 1-byte dump-type identifier, which contains one of the following values:

UEPDTRAN
A transaction dump was requested.

UEPDSYST
A system dump was requested.

Note: The dump-type identifier indicates the type of dump request that was passed to the dump domain. It does not reflect any modification that may have been made to the original request by a user entry in the dump table.

UEPXDSCP
Address of a 1-byte field indicating the current dump table DUMPSCOPE setting. It contains one of the following values:

UEPXDLOC
A system dump will be taken on the local MVS image only.

UEPXDREL
System dumps will be taken on both the local MVS image, and on related MVS images within the sysplex.

This field may be modified by the exit to update the dump table DUMPSCOPE setting.

UEPXDTXN
Address of a 1-byte field indicating the current dump table TRANDUMP setting. It contains one of the following values:

UEPXDYES
A transaction dump will be taken.

UEPXDNO
A transaction dump will not be taken.

This field may be modified by the exit to update the dump table TRANDUMP setting.

Note: This field is only valid if UEPDUMPT contains the value UEPDTRAN.

UEPXDSYS
Address of a 1-byte field indicating the current dump table SYSDUMP setting. It contains one of the following values:

UEPXDYES
A system dump will be taken.

UEPXDNO
A system dump will not be taken.

This field may be modified by the exit to update the dump table SYSDUMP setting.

UEPXDTRM
Address of a 1-byte field indicating the current dump table SHUTDOWN setting. It contains one of the following values:

UEPXDYES
The CICS system is to shutdown.

UEPXDNO
The CICS system is not to shutdown.

This field may be modified by the exit to update the dump table SHUTDOWN setting.

UEPXDMAX
Address of a 4-byte field which contains the current dump table MAXIMUM setting. This field may be modified by the exit to change the current dump table MAXIMUM setting. A change to the MAXIMUM setting will not suppress this dump request. A return code of UERCBYP may be used to suppress the current dump request.

UEPDXDCNT
Address of a 4-byte field which contains the current dump table CURRENT setting.

UEPXDTST
Address of a 16-byte field which contains the current dump table statistics for this dump code. The addressed field consists of four 4-byte fields containing binary integers:

- Number of transaction dumps taken
- Number of transaction dumps suppressed
- Number of system dumps taken
- Number of system dumps suppressed

Note: Statistics for transaction dumps are valid only if UEPDUMPT contains the value UEPDTRAN.

UEPXDDAE
Address of a 1-byte field which represents the current dump table DAEOPTION setting. It contains one of the following values:

UEPXDYES
The dump is eligible for DAE suppression.

UEPXDNO
The dump will not be suppressed by DAE.

This field may be modified by the exit to update the dump table DAEOPTION setting.

UEPDMPID
Address of a 9-character field in the format xxxx/xxxx, containing the dump identifier. The dump ID is the same as that output by the corresponding dump message.

UEPFMOD
Address of an 8-byte area containing, if the dump code is AP0001, the name of the failing module; otherwise null characters.

Note that field UEPPROG always addresses the name of the *current* application, regardless of where the failure occurred. UEPFMOD addresses the name of the module where the failure occurred, if known.

If the dump code is AP0001, there are three possibilities:

1. The field addressed by UEPFMOD contains the same name as the field addressed by UEPPROG—the failure occured in application code.

2. The field addressed by UEPFMOD contains a different name from the field addressed by UEPPROG—the failure occurred in non-application code.
3. The field addressed by UEPFMOD contains '????????'—the failure was not in application code, but CICS was unable to determine the name of the failing module.

Return codes

UERCNORM
Continue processing.

UERCBYP
Suppress dump.

UERCPURG
Task purged during XPI call.

XPI calls
WAIT_MVS can be used **only** when a UEPDUMPT indicates that a transaction dump is being taken. **Do not use any other calls**.

Exit XDUREQC

Exit XDUREQC is invoked immediately after a system or transaction dump has been taken, or has failed or been suppressed.

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID.

UEPUSER
Address of the 8-byte user ID.

UEPTERM
Address of the 4-byte terminal ID.

UEPPROG
Address of the 8-byte application program name.

UEPDUMPC
Address of copy of the 8-byte dump code.

UEPABCDE
Address of a copy of the 8-byte Kernel error code in the format xxx/yyyy. xxx denotes the 3-digit hexadecimal MVS completion code (for example X'0C1' or X'D37'). If an MVS completion code is not applicable, xxx is three hyphens. The 4-digit code yyyy is a user abend code produced either by CICS or by another product on your system. UEPABCDE is completed only for abend codes corresponding to the following dump codes:

- AP0001
- SR0001
- ASRA
- ASRB
- ASRD

Otherwise this field contains null characters.

UEPDUMPT
Address of the 1-byte dump-type identifier, which contains one of the following values:

UEPDTRAN
A transaction dump was requested.

UEPDSYST
A system dump was requested.

Note: The dump-type identifier indicates the type of dump request that was passed to the dump domain. It does not reflect any modification that may have been made to the original request by a user entry in the dump table.

UEPXDSCP
Address of a 1-byte field indicating the current dump table DUMPSCOPE setting. It contains one of the following values:

UEPXDLOC
A system dump will be taken on the local MVS image only.

UEPXDREL
System dumps will be taken on both the local MVS image, and on related MVS images within the sysplex.

This field may be modified by the exit to update the dump table DUMPSCOPE setting.

UEPXDTXN
Address of a 1-byte field indicating the current dump table TRANDUMP setting. It contains one of the following values:

UEPXDYES
A transaction dump will be taken.

UEPXDNO
A transaction dump will not be taken.

This field may be modified by the exit to update the dump table TRANDUMP setting.

Note: This field is only valid if UEPDUMPT contains the value UEPDTRAN.

UEPXDSYS
Address of a 1-byte field indicating the current dump table SYSDUMP setting. It contains one of the following values:

UEPXDYES
A system dump will be taken.

UEPXDNO
A system dump will not be taken.

This field may be modified by the exit to update the dump table SYSDUMP setting.

UEPXDTRM
Address of a 1-byte field indicating the current dump table SHUTDOWN setting. It contains one of the following values:

UEPXDYES
The CICS system is to shutdown.

UEPXDNO
The CICS system is not to shutdown.

This field may be modified by the exit to update the dump table SHUTDOWN setting.

UEPXDMAX
Address of a 4-byte field which contains the current dump table MAXIMUM setting. This field may be modified by the exit to change the current dump table MAXIMUM setting.

UEPDXDCNT
Address of a 4-byte field which contains the current dump table CURRENT setting.

UEPXDTST
Address of a 16-byte field which contains the current dump table statistics for this dumpcode. The addressed field consists of four 4-byte fields containing binary integers:

- Number of transaction dumps taken
- Number of transaction dumps suppressed
- Number of system dumps taken
- Number of system dumps suppressed.

Note: Statistics for transactions dumps are valid only if UEPDUMPT contains the value UEPDTRAN.

UEPXDDAE
Address of a 1-byte field which represents the current dump table DAEOPTION setting. It contains one of the following values:

UEPXDYES
The dump was suppressed by DAE.

UEPXDNO
The dump was not suppressed by DAE.

This field may be modified by the exit to update the dump table DAEOPTION setting.

UEPDMPID
Address of a 9-character field in the format xxxx/xxxx, containing the dump identifier. The dump ID is the same as that output by the corresponding dump message.

UEPDRESP
Address of the 2-byte dump response code.

UEPDREAS
Address of the 2-byte dump reason code.

Return codes

UERCNORM
Continue processing.

XPI calls
WAIT_MVS can be used only when a UEPDUMPT indicates that a transaction dump is being taken. Do not use any other calls.

Exit XDUCLSE

This exit is invoked immediately after a transaction dump data set has been closed.

When invoked

Immediately after a transaction dump data set has been closed.

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID.

UEPUSER
Address of the 8-byte user ID.

UEPTERM
Address of the 4-byte terminal ID.

UEPPROG
Address of the 8-byte application program name.

UEPDMPDD
Address of the 8-byte dump data set ddname.

UEPDMPDSN
Address of the 44-byte dump data set dsname.

Return codes

UERCNORM
Continue processing.

UERCSWCH
The autoswitch flag is set on.

XPI calls

WAIT_MVS can be used. **Do not use any other calls**.

Exit XDUOUT

This exit is invoked before a record is written to the transaction dump data set.

When invoked

Before a record is written to the transaction dump data set.

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID.

UEPUSER
Address of the 8-byte user ID.

UEPTERM
Address of the 4-byte terminal ID.

UEPPROG
Address of the 8-byte application program name.

UEPDMPFC
Address of the 1-byte function code. The equated values are:

UEPDMPWR
Buffer is about to be written.

UEPDMPRE
Dump is about to restart after autoswitch.

UEPDMPAB
Abnormal termination of dump.

UEPDMPDY
Buffer is about to be written, and the CICS dump data set is a dummy file or is closed.

UEPDMPBF
Address of the dump buffer, whose length is addressed by the parameter UEPDMPLEN.

UEPDMPLEN
Address of the 2-byte dump-buffer length.

Return codes

UERCNORM
Continue processing.

UERCBYP
Suppress dump record output.

XPI calls
WAIT_MVS can be used. **Do not use any other calls**.

Enqueue EXEC interface program exits XNQEREQ and XNQEREQC

You can use the XNQEREQ exit to intercept enqueue API requests before any action has been taken on the request. You can use the XNQEREQC exit to intercept the response after an enqueue API request has completed.

The API requests affected are:

- `EXEC CICS ENQ`
- `EXEC CICS DEQ`

XNQEREQ
Using **XNQEREQ**, you can:

- Analyze the API parameter list (function, keywords, argument values, and responses).
- Modify any input parameter value before execution of a request.
- Prevent execution of a request. This enables you to replace the CICS function with your own processing.

Notes for using XNQEREQ to alter the ENQ or DEQ scope:

1. XNQEREQ enables you to allow existing applications to be converted to use sysplex enqueues without changing the application.

   Note: Use of either the ENQMODEL resource definition or the user exit allows this in most cases, but those applications where the resource name is determined dynamically and not known in advance can only be so converted by use of this exit.
2. Sysplex and region scope enqueues use separate namespaces. A region scope enqueue will never wait on a sysplex enqueue, nor will a sysplex scope enqueue wait on a region enqueue.

   Note: This situation can only arise when you use the exit. Use of the ENQMODEL resource definitions as your only method of defining the SCOPE of an ENQ or DEQ avoids this potential risk.

3. Both region and sysplex scope are supported for string ENQs, but sysplex scope is not supported for address ENQ.

XNQEREQC

Using **XNQEREQC**, you can:

- Analyze the API parameter list.
- Pass data between your XNQEREQ and XNQEREQC exit programs when they are invoked for the same request
- Pass data between your enqueue exit programs when they are invoked within the same task.

CICS supplies a sample exit program, DFH$XNQE, for the enqueue EXEC interface. For more information, see Enqueue EXEC interface sample exit program: DFH$XNQE.

Exit XNQEREQ

This exit is invoked before CICS processes an EXEC CICS ENQ or DEQ request, or attempts to match it to an installed ENQMODEL resource definition.

When invoked

Before CICS processes an EXEC CICS ENQ or DEQ request, or attempts to match it to an installed ENQMODEL resource definition.

Exit-specific parameters

UEPCLPS

Address of a copy of the command parameter list. See "The command-level parameter structure" on page 46.

UEPNQTOK

Address of a 4-byte area which can be used to pass information between XNQEREQ and XNQEREQC for a single enqueue request.

UEPRCODE

Address of a 6-byte hexadecimal copy of the EIB return code EIBRCODE. For details of EIB return codes, see EIB fields.

UEPRESP

Address of a 4-byte binary copy of the EIB response code EIBRESP.

UEPRESP2

Address of a 4-byte binary copy of the EIB response code EIBRESP2.

UEPTSTOK

Address of a 4-byte token which can be used to pass information between successive enqueue requests within the same task (for example, between successive invocations of the XNQEREQ exit).

UEPRECUR

Address of a halfword recursion counter. The counter is set to 0 when the exit is first invoked, and is incremented for each recursive call.

UEPSCOPE

Address of the 4-byte ENQSCOPE name to be used.

Return codes

UERCBYP

Bypass this request.

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

UERCSCPE
An ENQSCOPE name has been supplied.

XPI calls
All can be used.

API and SPI commands
All can be used, except for:
EXEC CICS SHUTDOWN
EXEC CICS XCTL

Note: Take care when issuing recursive commands. For example, you must avoid entering a loop when issuing an enqueue request from the XNQEREQ exit. Use of the recursion counter UEPRECUR is recommended.

Exit XNQEREQC

Exit XNQEREQC is invoked after an enqueue API request has completed, before return from the enqueue EXEC interface program.

Exit-specific parameters

UEPCLPS
Address of a copy of the command parameter list. See "The command-level parameter structure" on page 46.

UEPNQTOK
Address of a 4-byte area which can be used to pass information between XNQEREQ and XNQEREQC for a single enqueue request.

UEPRCODE
Address of a 6-byte hexadecimal copy of the EIB return code EIBRCODE. For details of EIB return codes, see EIB fields.

UEPRESP
Address of a 4-byte binary copy of the EIB response code EIBRESP.

UEPRESP2
Address of a 4-byte binary copy of the EIB response code EIBRESP2.

UEPTSTOK
Address of a 4-byte token which can be used to pass information between successive enqueue requests within the same task (for example, between successive invocations of the XNQEREQC exit).

UEPRECUR
Address of a halfword recursion counter. The counter is set to 0 when the exit is first invoked, and is incremented for each recursive call.

UEPSCOPE
Address of the 4-byte ENQSCOPE name used.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

API and SPI commands
All can be used, except for:
EXEC CICS SHUTDOWN
EXEC CICS XCTL

You can update the copies of EIBRCODE, EIBRESP, and EIBRESP2 that you are given in the parameter list. If you update the values, CICS copies the new values into the application program's EIB after the completion of XNQEREQC or if you specify a return code of UERCBYP in XNQEREQ.

You must set valid enqueue responses. You must set all three of EIBRCODE, EIBRESP, and EIBRESP2 to a consistent set of values, such as would be set by the enqueue domain to describe a valid completion. CICS does not check the consistency of EIBRCODE, EIBRESP, and EIBRESP2. If EIBRCODE is set to a non-zero value and EIBRESP is set to zero, CICS will override EIBRESP with a non-zero value. To help you set values for EIBRCODE, EIBRESP, and EIBRESP2, the values used by the enqueue domain are specified in DSECT DFHNQUED.

Note: Take care when issuing recursive commands not to cause a loop. For example, it is your responsibility to avoid entering a loop when issuing an enqueue request from the XNQEREQC exit. Use of the recursion counter UEPRECUR is recommended.

The command-level parameter structure

The command-level parameter structure consists of a series of addresses. The first address points to the EXEC interface descriptor (EID), which consists of a bit string that describes the type of request and identifies each keyword specified with the request. The remaining addresses point to pieces of data associated with the request.

End of parameter list indicator

You can examine the EID to determine the type of request and the keywords specified. You can examine the other parameters in the list to determine the values of the keywords. You can also modify values of keywords specified on the request.

The high-order bit is set on in the last address set in the parameter list to indicate that it is the last one in the list. On return from your user exit program, CICS scans the parameter list for the high-order bit to find the last parameter. Therefore, if you modify the length of the parameter list, you must also reset the high-order bit to indicate which is the new last address.

The UEPCLPS exit-specific parameter

The UEPCLPS exit-specific parameter is included in both exit XNQEREQ and exit XNQEREQC, and contains the address of the command-level parameter structure.

The command-level parameter structure contains four addresses, NQ_ADDR0 through NQ_ADDR3. It is defined in the DSECT NQ_ADDR_LIST, which you should copy into your exit program by including the statement COPY DFHNQUED.

The command-level parameter list is made up as follows.

NQ_ADDR0
is the address of a 9-byte area called the EID, which is made up as follows:

- **NQ_GROUP**
- **NQ_FUNCT**
- **NQ_BITS1**
- **NQ_BITS2**
- **NQ_EIDOPT5**
- **NQ_EIDOPT6**
- **NQ_EIDOPT7**
- **NQ_EIDOPT8**

NQ_GROUP
Always X'12', indicating that this is a task control request.

NQ_FUNCT
One byte that defines the type of request:

X'04' ENQ

X'06' DEQ

NQ_BITS1
Existence bits that define which arguments were specified. To obtain the argument associated with a keyword, you need to use the appropriate address from the command-level parameter structure. Before using this address, you must check the associated existence bit. If the existence bit is set off, the argument was not specified in the request and the address should not be used.

X'80' Set if the request contains an argument for the RESOURCE keyword. If set, **NQ_ADDR1** is meaningful.

X'40' Set if the request contains an argument for the LENGTH keyword. If set, **NQ_ADDR2** is meaningful.

X'20' Set if the request contains an argument for the MAXLIFETIME keyword. If set, **NQ_ADDR3** is meaningful.

NQ_BITS2
Two bytes not used by the enqueue domain.

NQ_EIDOPT5
One byte not used by the enqueue domain.

NQ_EIDOPT6
One byte not used by the enqueue domain.

NQ_EIDOPT7
One byte not used by the enqueue domain.

NQ_EIDOPT8
Indicates whether certain keywords were specified on the request.

X'04' NOSUSPEND was specified.

X'02' DEQ was specified.

X'01' ENQ was specified.

NQ_ADDR1
is the address of an area containing the value from RESOURCE.

NQ_ADDR2
is the address of the halfword value of LENGTH.

NQ_ADDR3
is the address of the fullword value of MAXLIFETIME.

The relationship between arguments, keywords, data types, and input/output types is summarized for the enqueue commands in Table 2 on page 49.

Modifying fields in the command-level parameter structure

The fields that are passed to the enqueue domain are used as input to the request. The correct method of modifying an input field is to create a new copy of it, and to change the address in the command-level parameter list to point to your new data.

Important:

1. Do not modify an input field by altering the data that is pointed to by the command-level parameter list. To do so would corrupt storage belonging to the application program and would cause a failure when the program attempted to reuse the field.
2. There are no output fields on EXEC CICS ENQ and DEQ requests.

Modifying the EID

It is not possible to modify the EID to make major changes to requests. It is not possible, for example, to change an ENQ request to a DEQ request. However, you can make minor changes to requests, such as to turn on the existence bit for LENGTH.

The list that follows shows the bits in the EID that can be modified. Any attempt to modify any other part of the EID is ignored.

NQ_BITS1

X'40' The existence bit for LENGTH

X'20' The existence bit for MAXLIFETIME.

NQ_EIDOPT7
A user exit program at XNQEREQ can set the following on or off for ENQ commands:

X'04' The existence bit for NOSUSPEND.

The EID is reset to its original value before return to the application program. That is, changes made to the EID are retained for the duration of the enqueue request only.

Note: Your user exit program is prevented from making major changes to the EID. However, you must take great care when making the minor modifications that **are** permitted.

Use of the task token UEPTSTOK

The task token, UEPTSTOK, provides the address of a 4-byte area that you can use to pass information between successive enqueue requests in the same task. For example, you can use the task token if you need to pass information between successive invocations of the XNQEREQ exit.

By contrast, UEPNQTOK is usable only for the duration of a single enqueue request, because its contents can be destroyed at the end of the request.

Note:

1. The lifetime of the area pointed to by UEPTSTOK is the lifetime of the task.
2. The value of UEPTSTOK is shared by all the exits to which it is passed during the lifetime of the task.

Table 2. User arguments and associated keywords, data types, and input/output types

Argument	Keyword	Data type	Input/output type
Arg1	RESOURCE	DATA-AREA	input
Arg2	LENGTH	BIN(15)	input
Arg3	MAXLIFETIME	CVDA	input

Modifying user arguments

User exit programs can modify user input arguments either by obtaining/setting storage, or by setting a pointer.

User exit programs can modify user input arguments by:

1. Obtaining sufficient storage to hold the modified argument
2. Setting the storage to the required value
3. Setting the associated pointer in the parameter list to the address of the newly-acquired area.

Note:

1. CICS does not check changes to argument values, so any changes must be verified by the user exit program making the changes.
2. It is not advisable for XNQEREQC to modify input arguments.

Adding user arguments

Global user exit programs can add arguments associated with the LENGTH and MAXLIFETIME keywords. You must ensure that the arguments you specify or modify in your exit programs are valid.

The valid values for MAXLIFETIME are DFHVALUE(TASK) and DFHVALUE(UOW), which are 233 and 246 respectively.

Assuming that the argument to be added does not already exist, the user exit program must:

1. Obtain storage for the argument to be added
2. Initialize the storage to the required value
3. Select and set up the appropriate pointer from the parameter list
4. Select and set up the appropriate argument existence bit in the EID
5. Modify the parameter list to reflect the new end of list indicator.

Removing user arguments

User exit programs can remove arguments (for which the program is totally responsible) associated with the LENGTH and MAXLIFETIME keywords.

Assuming that the argument to be removed exists, the user exit program must:

1. Switch the corresponding argument existence bit to '0'b in the EID
2. Modify the parameter list to reflect the new end of list indicator.

Event capture exit XEPCAP

The XEPCAP exit is invoked just before an event is captured by CICS event processing. Use the XEPCAP exit to detect when events are captured.

Exit-specific parameters

UEPEPCX
Address of the EPCX (event context data structure). This parameter contains information about the event being captured. For more information about EPCX, see EPCX Event Processing Context Container.

UEPEPTASK
Address of a 4-byte (packed decimal) field containing the task number.

UEPLOAD
Address of the capturing transaction program load point.

UEPRSA
Address of the capturing transaction register save area, which contains the contents of the registers at the point when the program issued the EXEC CICS command. If the event is captured from program initialization rather than an API call then this parameter is set to 0.

Return codes

UERCNORM
Continue processing.

XPI calls
You can use all XPI calls.

API and SPI commands
No EXEC CICS commands can be used.

EXEC interface program exits XEIIN, XEIOUT, XEISPIN, and XEISPOUT

There are four global user exit points in the EXEC interface program that you can use before or after an API or SPI call.

XEIIN Invoked before the execution of any EXEC CICS application programming interface (API) or system programming interface (SPI) command.

XEISPIN
Invoked before the execution of any EXEC CICS SPI command except:

- **EXEC CICS ENABLE**
- **EXEC CICS DISABLE**
- **EXEC CICS EXTRACT EXIT**
- **EXEC CICS PERFORM DUMP**
- **EXEC CICS RESYNC ENTRYNAME**

The sequence is:

```
TRACE  –  XEIIN  –  XEISPIN  –  EDF  –  command
```

XEIOUT
Invoked after the execution of any EXEC CICS API or SPI command.

XEISPOUT

Invoked after the execution of any EXEC CICS SPI command except those listed for XEISPIN.

The sequence is:

```
command  –  EDF  –  XEISPOUT  –  XEIOUT  –  TRACE
```

Note: Asynchronous processing of these exits might occur if the transaction is suspended; for example, during file I/O wait. This situation might also occur under CEDF because CEDF issues its own EXEC CICS commands between the application's XEISPIN and XEISPOUT exits.

If, for example, the same GWA is shared between the XEIIN and XEIOUT exits, you must allow for the possibility of asynchronous processing, in order to ensure integrity of the data and to prevent unpredictable results.

On entry to the exits, the exit-specific parameter UEPARG contains the address of the command parameter list.

The command parameter list

The first parameter in the list points to a string of data known as *argument 0*. The other parameters point to the values specified for the parameters passed on the command.

Argument 0 begins with a 2-byte function code that identifies the command. Function codes are documented in EIB fields and in EXEC interface block (EIB) response and function codes. The function code is followed by a 2-byte field that contain "existence bits", which indicate whether arguments are passed on the command. For example, consider the command:

```
EXEC CICS LINK PROGRAM('MYPROG')
```

Here, argument 0 begins with the function code X'0E02' (LINK). Existence bit 1 is set, indicating that there is an argument 1 (namely, 'MYPROG').

The correspondence between command parameters (such as PROGRAM) and their positions and values in the parameter list (in this case, argument 1, 'MYPROG') can be deduced from the translated code for the particular command.

Important:

Modifying CICS commands by changing argument 0 is not supported, and leads to unexpected errors or results.

For example, if an application program is written in assembler or PL/I and you modify argument 0, you will be writing to program storage (that is, storage occupied by the program itself), which could cause 0C4 abends. Furthermore, modifying argument 0 not only alters the CICS command for this execution of the command in the application program, it changes the CICS command in the virtual storage copy of the application program. This means that the next task to invoke the same copy of the program will also execute the modified command.

This particular example of the danger of modifying argument 0 does not apply to COBOL or C application programs, but nevertheless you should not modify CICS commands for application programs written in any supported language.

Bypassing commands

An XEIIN or XEISPIN exit program can bypass execution of a command by setting the UERCBYP return code. If it does this, EDF is not invoked, but XEISPOUT, XEIOUT, and exit trace are invoked if they are active.

Bypassing an EXEC CICS command allows an exit program to replace the CICS function with its own processing, for example.

Before setting UERCBYP, your program should check the value pointed to by UEPPGM, to ensure that it is not bypassing an EXEC CICS command issued by CICS.

Exit XEIIN

Exit XEIIN is invoked before the execution of any EXEC CICS API or SPI command.

Exit-specific parameters

UEPARG
Address of the EXEC command parameter list.

UEPEXECB
Address of the system EIB.

UEPUSID
Address of the 8-character userid.

UEPPGM
Address of the 8-character application program name.

UEPLOAD
Address of the application program's load-point.

UEPRSA
Address of the application's register save area. This contains the contents of the registers at the point when the program issued the EXEC CICS command.

UEP_EI_PBTOK
Address of a 4-byte field containing the z/OS Workload Manager (WLM) Performance Block Token. An exit program can use this token to access information (such as the service class token, SERVCLS) in the WLM Performance Block. To do so, it must use the WLM EXTRACT macro, IWMMEXTR, passing the Performance Block Token as the MONTKN input parameter. For more information about the IWMMEXTR macro, see *z/OS MVS Programming: Workload Management Services*.

An exit program must not attempt to modify the Performance Block: if it does so, the results are unpredictable.

Return codes

UERCNORM
Continue processing.

UERCBYP
Bypass the execution of this command.

UERCPURG
Task purged during XPI call.

XPI calls

All can be used.

Exit XEISPIN

Exit XEISPIN is invoked before the execution of some SPI commands. The exit is not invoked for the **ENABLE**, **DISABLE**, **EXTRACT EXIT**, **PERFORM DUMP** and **RESYNC ENTRYNAME** commands.

Exit-specific parameters

UEPARG

Address of the EXEC command parameter list.

UEPEXECB

Address of the system EIB.

UEPUSID

Address of the 8-character userid.

UEPPGM

Address of the 8-character application program name.

UEPLOAD

Address of the load-point of the application program.

UEPRSA

Address of the register save area of the application program. This area contains the contents of the registers at the point when the program issued the EXEC CICS command.

UEP_EI_PBTOK

Address of a 4-byte field that contains the z/OS Workload Manager (WLM) Performance Block Token. An exit program can use this token to access information in the WLM Performance Block, for example, the service class token, SERVCLS. To do so, it must use the WLM EXTRACT macro, IWMMEXTR, passing the Performance Block Token as the MONTKN input parameter. For more information about the IWMMEXTR macro, see z/OS MVS Programming: Workload Management Services.

An exit program must not attempt to modify the Performance Block: if it does so, the results are unpredictable.

Return codes

UERCNORM

Continue processing.

UERCBYP

Bypass the execution of this command.

UERCPURG

Task purged during XPI call.

XPI calls

All can be used.

Exit XEIOUT

Exit XEIOUT is invoked after the execution of any EXEC CICS API or SPI command.

Exit-specific parameters

UEPARG
Address of the EXEC command parameter list.

UEPEXECB
Address of the system EIB.

UEPUSID
Address of the 8-character userid.

UEPPGM
Address of the 8-character application program name.

UEPLOAD
Address of the application program's load-point.

UEPRSA
Address of the application's register save area. This contains the contents of the registers at the point when the program issued the EXEC CICS command.

UEP_EI_PBTOK
Address of a 4-byte field containing the z/OS Workload Manager (WLM) Performance Block Token. An exit program can use this token to access information (such as the service class token, SERVCLS) in the WLM Performance Block. To do so, it must use the WLM EXTRACT macro, IWMMEXTR, passing the Performance Block Token as the MONTKN input parameter. For more information about the IWMMEXTR macro, see *z/OS MVS Programming: Workload Management Services*.

An exit program must not attempt to modify the Performance Block: if it does so, the results are unpredictable.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

Exit XEISPOUT

Exit XEISPOUT is invoked after the execution of some SPI commands. The exit is not invoked for the **ENABLE**, **DISABLE**, **EXTRACT EXIT**, **PERFORM DUMP** and **RESYNC ENTRYNAME** commands.

Exit-specific parameters

UEPARG
Address of the EXEC command parameter list.

UEPEXECB
Address of the system EIB.

UEPUSID
Address of the 8-character userid.

UEPPGM
Address of the 8-character application program name.

UEPLOAD
Address of the application program's load-point.

UEPRSA
Address of the application's register save area. This contains the contents of the registers at the point when the program issued the EXEC CICS command.

UEP_EI_PBTOK
Address of a 4-byte field containing the z/OS Workload Manager (WLM) Performance Block Token. An exit program can use this token to access information (such as the service class token, SERVCLS) in the WLM Performance Block. To do so, it must use the WLM EXTRACT macro, IWMMEXTR, passing the Performance Block Token as the MONTKN input parameter. For more information about the IWMMEXTR macro, see *z/OS MVS Programming: Workload Management Services*.

An exit program must not attempt to modify the Performance Block: if it does so, the results are unpredictable.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

Front End Programming Interface exits XSZARQ and XSZBRQ

If you have installed the Front End Programming Interface (FEPI), you can use global user exits XSZARQ and XSZBRQ before and after FEPI commands.

XSZBRQ
Invoked before a FEPI command is executed (but after the syntax of the command has been validated, and therefore after EDF processing).

XSZARQ
Invoked immediately after a FEPI command has completed (before EDF processing).

Note that both the FEPI application programming and system programming commands cause XSZBRQ and XSZARQ to be invoked, but the latter do not provide the exit programs with any meaningful information.

You cannot use exit programming interface (XPI) calls or EXEC CICS commands in programs invoked from these exits. The exits allow you to monitor the FEPI commands and data being processed; you can inhibit commands, and modify specific command options. You could use them for:

- Monitoring the issue of FEPI commands
- Workload routing
- External security on application programming commands.

XSZBRQ

XSZBRQ is invoked before a FEPI command is executed; the input parameters for the command are passed to the exit program.

The majority of the information passed is read-only, but you can write a program to update specific parameters. FEPI does not check the validity of the new values for the updated parameters. In addition, your exit program can decide whether the request is to be processed or bypassed. You could use XSZBRQ, for example, to log commands, to bypass commands that violate the conventions of your installation, or to reroute commands by changing their specified targets or pools.

Together, UEPSZALP and UEPSZALT contain the information necessary to initiate a conversation.

When invoked

Invoked by FEPI before a FEPI command runs, but after syntax and semantic checking.

Exit-specific parameters

UEPSZACT

A 2-byte field that identifies the command. The values are given in Table 3 on page 58.

UEPSZCNV

An 8-character field containing the conversation ID (CONVID) for the command. Applicable on FEPI ALLOCATE, SEND, RECEIVE, CONVERSE, EXTRACT, ISSUE, START, and FREE commands.

UEPSZALP

An 8-character field containing the name of the pool (POOL). Modifiable and applicable on FEPI ALLOCATE and CONVERSE commands.

UEPSZALT

An 8-character field containing the name of the target (TARGET). Modifiable and applicable on FEPI ALLOCATE and CONVERSE commands.

UEPSZTIM

Fullword binary field containing the timeout value (TIMEOUT). Modifiable and applicable on FEPI ALLOCATE, RECEIVE, CONVERSE, and START commands.

UEPSZSND

Address of the 'send' data-area (FROM). Applicable on FEPI CONVERSE and SEND commands.

UEPSZSNL

Fullword binary field containing the length of the 'send' data (FROMFLENGTH, FLENGTH). Applicable on FEPI CONVERSE and SEND commands.

UEPSZSTT

A 4-character field containing the transaction ID (TRANSID). Modifiable and applicable on FEPI START commands.

UEPSZSTM

A 4-character field containing the terminal ID (TERMID). Modifiable and applicable on FEPI START commands.

UEPSZSNK
A 1-bit flag field indicating whether data is in key stroke format (KEYSTROKE). Applicable on FEPI CONVERSE FORMATTED and SEND FORMATTED commands. It can contain the following values:

UEPSZSNK_OFF
Not key stroke format.

UEPSZSNK_ON
Key stroke format.

UEPSZSNE
A 1-character field containing the key stroke escape character (ESCAPE). Applicable on FEPI CONVERSE FORMATTED and SEND FORMATTED commands.

Return codes

UERCNORM
Continue processing.

UERCBYP
Do not process the request; return INVREQ to the application.

Note: Your exit program cannot bypass events (like CICS shutdown or end-of-task).

XPI calls
Do not use XPI calls.

XSZARQ

XSZARQ is invoked immediately after a FEPI command has been executed; the exit program is passed the parameters that are output from the command. All of the information passed is read-only.

When invoked
Invoked by FEPI immediately after a FEPI command has been processed.

Exit-specific parameters

UEPSZACN
A 2-byte field that identifies the command. The values are given in Table 3 on page 58.

UEPSZCON
An 8-character field containing the conversation ID (CONVID) for the command. Applicable on FEPI ALLOCATE, SEND, RECEIVE, CONVERSE, EXTRACT, ISSUE, START, and FREE commands.

UEPSZRP2
Fullword containing the response code for the command (RESP2).

UEPSZRVD
Address of the 'receive' data-area (INTO). Applicable on FEPI RECEIVE, CONVERSE, and EXTRACT FIELD commands.

UEPSZRVL
Fullword binary data field containing the length of the receive data (FLENGTH, TOFLENGTH). Applicable on FEPI RECEIVE, CONVERSE, and EXTRACT FIELD commands.

Return code

UERCNORM
Continue processing.

XPI calls
Do not use any XPI calls.

The UEPSZACT and UEPSZACN exit-specific parameters

Both XSZBRQ and XSZARQ are passed a parameter (**UEPSZACT** for XSZBRQ, and **UEPSZACN** for XSZARQ) indicating the command or event being processed.

Table 3. relates the hexadecimal values passed in UEPSZACT and UEPSZACN to the FEPI commands they represent.

Table 3. Settings of UEPSZACT for exit XSZBRQ and UEPSZACN for exit XSZARQ

Name	Setting (hex)	FEPI command or event
UEPSZNOA	820E	AP NOOP
UEPSZOAL	8210	ALLOCATE
UEPSZOCF	8212	CONVERSE FORMATTED
UEPSZOCD	8214	CONVERSE DATASTREAM
UEPSZOXC	8216	EXTRACT CONV
UEPSZOXF	8218	EXTRACT FIELD
UEPSZOXS	821A	EXTRACT STSN
UEPSZOFR	821C	FREE
UEPSZOSU	821E	ISSUE
UEPSZORF	8220	RECEIVE FORMATTED
UEPSZORD	8222	RECEIVE DATASTREAM
UEPSZOSF	8224	SEND FORMATTED
UEPSZOSD	8226	SEND DATASTREAM
UEPSZOST	8228	START
UEPSZSDN	8402	CICS normal shutdown 1
UEPSZSDI	8404	CICS immediate shutdown 1
UEPSZSDF	8406	CICS forced shutdown 1
UEPSZEOT	8408	CICS end-of-task 1
UEPSZNOS	840E	SP NOOP
UEPSZOQY	8422	INQUIRE PROPERTYSET
UEPSZOIY	8428	INSTALL PROPERTYSET
UEPSZODY	8430	DISCARD PROPERTYSET
UEPSZOQN	8442	INQUIRE NODE
UEPSZOTN	8444	SET NODE
UEPSZOIN	8448	INSTALL NODELIST
UEPSZOAD	844A	ADD POOL
UEPSZODE	844C	DELETE POOL
UEPSZODN	8450	DISCARD NODELIST
UEPSZOQP	8462	INQUIRE POOL
UEPSZOTP	8464	SET POOL
UEPSZOIP	8468	INSTALL POOL
UEPSZODP	8470	DISCARD POOL
UEPSZOQT	8482	INQUIRE TARGET
UEPSZOTT	8484	SET TARGET
UEPSZOIT	8488	INSTALL TARGETLIST
UEPSZODT	8490	DISCARD TARGETLIST
UEPSZOQC	84A2	INQUIRE CONNECTION
UEPSZOTC	84A4	SET CONNECTION

Note:

- 1 These events are generated internally by CICS; you cannot bypass them.

Using XMEOUT to control message output

You can use the XMEOUT global user exit, in the CICS message domain, to suppress or reroute FEPI messages.

Note, however, that error conditions that generate a message also generate a transient data queue record. It is more efficient to handle such events using a monitoring program, through the TD queue, than by duplicating a message and then acting on it.

File control domain exits, XFCFRIN and XFCFROUT

The XFCFRIN exit is invoked on entry to the main file control request gate, FCFR, and the XFCFROUT exit runs after the completion of a file control request. The XFCFRIN and XFCFROUT exits must be coded to threadsafe standards and declared threadsafe to take advantage of threadsafe remote file support.

XFCFRIN

XFCFRIN allows you to write a program to perform one or more of the following tasks:

- Monitor file control requests and allow them to continue, to be processed by CICS file control
- Intercept file control requests and bypass CICS file control processing altogether
- Redirect the request to a remote region.

If the exit program passes the request to CICS file control (without choosing to redirect it to a remote region), it is not allowed to make changes to any of the parameters. If the exit program intercepts the request and bypasses file control:

- It must return all the responses and output parameters that would otherwise have been returned by file control. These are marked **output** in the descriptions of the exit-specific parameters.
- It must indicate whether, if the request was function-shipped, the mirror transaction is permitted to terminate. Certain file control requests require that another request has been executed previously in the same transaction. (For example, READNEXT must be preceded by a matching STARTBR; REWRITE must be preceded by a matching READ, READNEXT, or READPREV with the UPDATE option). If the mirror transaction terminates between two such requests, the second is likely to fail. Conversely, a mirror transaction that is retained unnecessarily will hold on to CICS resources and may contribute to storage and locking problems.
- CICS terminates file browses and outstanding updates as part of syncpoint processing. However, the XFCFRIN exit is not invoked for syncpoint. If you want to emulate this aspect of CICS behavior accurately, or you want to support recoverable resources, you must invoke a task-related user exit program which schedules the syncpoint manager—see Coding a program to be started by the CICS sync point manager.

To redirect the request to a remote region, the exit program must add or change the value of the SYSID parameter. In this case, it may also need to

supply the values of the key length and record length. It is not permitted to make changes to any of the other parameters.

XFCFROUT

XFCFROUT allows you to monitor the results of completed file control requests. For example, if you didn't choose to bypass CICS file control processing, you can analyze the (CICS-internal) file control request to determine its type, the parameters passed to file control, and the values returned. It is invoked in both the following cases:

- After CICS file control has completed its processing, either normally or with an error
- If your XFCFRIN exit program chooses to bypass CICS file control processing.

All parameters are input-only; your exit program cannot modify any of the values.

To use IPIC connections for function shipping file control requests, ensure that XFCFRIN and XFCFROUT check that the UEPTERM parameter is a non-zero value before trying to use it as an address. The UEPTERM parameter is a zero for file control requests that have been function shipped over an IPIC connection.

Exit XFCFRIN

Exit XFCFRIN is invoked before the execution of a file control request.

The request can have originated from:

- The execution of an application request to process a user file
- The receipt of a function-shipped request
- An internal CICS request to process a system file.

Exit-specific parameters

UEPTRANID

Address of the 4-byte transaction ID.

UEPUSER

Address of the 8-byte user ID.

UEPTERM

Zero, or the address of the 4-byte terminal ID. If no address is returned, this could mean that this request has been function shipped over an IPIC connection.

UEPPROG

Address of the 8-byte application program name.

UEP_FC_FUNCTION

Address of a byte containing the function. The FCFR functions are derived from those available through the EXEC CICS interface, where certain request options (SET, INTO, UPDATE) have been included in the function values. For example, UEP_FC_FUN_DELETE is derived from EXEC CICS DELETE with the RIDFLD option specified; UEP_FC_FUN_REWRITE_DELETE is derived from EXEC CICS DELETE without RIDFLD. The possible values are:

- UEP_FC_FUN_READ_INTO
- UEP_FC_FUN_READ_SET
- UEP_FC_FUN_READ_UPDATE_INTO
- UEP_FC_FUN_READ_UPDATE_SET

- UEP_FC_FUN_WRITE
- UEP_FC_FUN_REWRITE
- UEP_FC_FUN_REWRITE_DELETE
- UEP_FC_FUN_DELETE
- UEP_FC_FUN_UNLOCK
- UEP_FC_FUN_START_BROWSE
- UEP_FC_FUN_READ_NEXT_INTO
- UEP_FC_FUN_READ_NEXT_SET
- UEP_FC_FUN_READ_PREVIOUS_INTO
- UEP_FC_FUN_READ_PREVIOUS_SET
- UEP_FC_FUN_READ_NEXT_UPDATE_INTO
- UEP_FC_FUN_READ_NEXT_UPDATE_SET
- UEP_FC_FUN_READ_PREVIOUS_UPDATE_INTO
- UEP_FC_FUN_READ_PREVIOUS_UPDATE_SET
- UEP_FC_FUN_RESET_BROWSE
- UEP_FC_FUN_END_BROWSE

UEPTSTOK
Address of a 4-byte token that is valid throughout the life of a task. See Using the task token UEPTSTOK.

UEP_FC_FILE_NAME
Address of an 8-byte modifiable field containing the filename.

UEP_FC_BUFFER_P
Address of a fullword containing the address of the buffer provided by the originator of the request, in which the (**output**) record is to be returned on completion of a READ, READ NEXT, or READ PREV request with the INTO option.

UEP_FC_BUFFER_L
Address of a fullword containing (for READ, READ NEXT, and READ PREV requests) the value of the LENGTH of the buffer into which the record is to be read.

UEP_FC_RECORD_P
Address of one of the following:

- If the request is a READ, READ NEXT, or READ PREV with the SET option, a fullword in which is to be returned the address (**output**) of a buffer, into which the record will be placed. The buffer itself is supplied either by CICS file control or, if the exit program bypasses file control, by the exit program.
- If the request is WRITE or REWRITE, a fullword containing the address of the record to be written.

UEP_FC_RECORD_L
Address of a fullword containing (for READ, WRITE, REWRITE, READ NEXT, and READ PREV requests) the value of LENGTH.

For all READ, READ NEXT, or READ PREV requests, this is an **output** field, in which the actual length of the record read is placed on return.

Warning: For requests that specify INTO, do not change the value of LENGTH to a value greater than the value specified by the UEP_FC_BUFFER_L field. To do so could cause a storage overlay in the application.

For a WRITE or REWRITE, this is an optional field which, if present, contains the length of the record to be written. If the field is not specified, the fullword contains binary zeroes. In this case, to modify the record length for a remote file use UEP_FC_M_RECORD_L instead.

See also the description of the UEP_FC_SYSID parameter.

UEP_FC_MAX_RECORD_L
Address of a fullword containing the (**output**) maximum record length of the file. (CICS function shipping uses this value to update the file's entry in the remote file control table.)

UEP_FC_RECORD_ID_P
Address of a fullword containing the address of the RIDFLD (record identifier) value. For a discussion of when the record identifier is an input or an **output** field, see Table 4 on page 78.

UEP_FC_RECORD_ID_L
Address of the halfword value of KEYLENGTH, which is the (possibly partial) length of the record identifier.

KEYLENGTH is an optional input parameter on READ, WRITE, DELETE, START BR, READ NEXT, READ PREV, and RESET BR requests. If the field is not specified, the halfword contains binary zeroes. In this case, to modify the key length for a remote file use UEP_FC_M_RECORD_ID_L instead.

See also the description of the UEP_FC_SYSID parameter.

UEP_FC_FULL_RECORD_ID_L
Address of the halfword value of the full length of the record identifier.

The full length of the record identifier is returned as a mandatory **output** field on READ NEXT and READ PREV requests. The value is used by CICS function shipping.

UEP_FC_RECORD_ID_TYPE
Address of a byte containing (for READ, WRITE, DELETE, START BR, READ NEXT, READ PREV, and RESET BR requests) the RIDFLD type. On input to the exit, this parameter will be set to one of:

UEP_FC_KEY
VSAM KSDS or AIX® PATH access

UEP_FC_RBA
VSAM ESDS or KSDS via RBA access

UEP_FC_RRN
VSAM RRDS access

UEP_FC_DEBKEY
BDAM deblocking by key (READ, DELETE, START BR, and RESET BR requests only)

UEP_FC_DEBREC
BDAM deblocking by relative record (READ, DELETE, START BR, and RESET BR requests only)

UEP_FC_XRBA
VSAM extended ESDS access

UEP_FC_REQID
Address (for START BR, READ NEXT, READ PREV, RESET BR, and END BR requests) of the halfword value of REQID.

UEP_FC_NUMREC
Address of the fullword value of NUMREC (**output**), in which (if the request is DELETE with RIDFLD) the number of records that have been deleted is returned.

UEP_FC_KEY_COMPARISON
Address of a byte containing (for READ, START BR, and RESET BR requests) the key comparison setting. On input to the exit, this parameter will be set to one of:

UEP_FC_EQUAL
Key-equal-to comparison is to be used.

UEP_FC_GTEQ
Key-greater-than-or-equal-to comparison is to be used.

UEP_FC_GENERIC
Address of a byte containing (for READ, DELETE, START BR, and RESET BR requests) the generic key setting. On input to the exit, this parameter will be set to one of:

UEP_FC_GENERIC_KEY
Generic key is to be used for key search.

UEP_FC_FULL_KEY
Full key is to be used for key search.

UEP_FC_MASS_INSERT
Address of a byte containing (for WRITE requests) the mass insert setting. On input to the exit, this parameter will be set to one of:

UEP_FC_SEQUENTIAL_WRITE
Records are to be written in sequential mode.

UEP_FC_DIRECT_WRITE
Records are to be written in direct mode.

UEP_FC_READ_INTEGRITY
Address of a byte containing (for non-update READ, READ NEXT, and READ PREV requests) the read integrity setting. (In current versions of CICS, this setting applies only to VSAM RLS.) On input to the exit, this parameter will be set to one of:

UEP_FC_CR
Consistent read integrity is to be used.

UEP_FC_FCT_VALUE
Read integrity is according to the setting in the FILE definition.

UEP_FC_NRI
The record is to be read with no read integrity.

UEP_FC_RR
Repeatable read integrity is to be used.

UEP_FC_TOKEN
Address of a fullword containing the value of TOKEN.

If the request is READ, READ NEXT, or READ PREV with update, and the address is not null, the area is an **output** field in which the token is to be returned.

If the request is REWRITE, DELETE without RIDFLD, or UNLOCK, the area is an input field.

UEP_FC_SYSID
Address of a 4-byte area that is to contain the SYSID identifying the remote region. On input to the exit, the area contains either:
- The value of the SYSID option of the API call, *or*
- Blanks (if SYSID was not specified).

To redirect the request to a different region, the exit program must place the SYSID of the target region in this **output** area.

If this parameter is set by the exit program, the request is function-shipped by file control without any reference to the file's attributes. If the key length has not been included on the request, the exit program must establish its value by setting the UEP_FC_RECORD_ID_L parameter.

Similarly, if the request is WRITE or REWRITE, and the record length has not been specified on the request, the exit program must establish its value by setting the UEP_FC_RECORD_L parameter.

UEP_FC_LENGTH_ERROR_CODE
Address of a 1-byte **output** area containing the length error code to be returned after a request has completed. The possible values are:
- UEP_FC_LENGTH_OK
- UEP_FC_BUFFER_LEN_TOO_SMALL
- UEP_FC_RECORD_LEN_TOO_LARGE
- UEP_FC_BUFFER_LEN_NOT_FILE_LEN
- UEP_FC_RECORD_LEN_NOT_FILE_LEN

UEP_FC_DUPLICATE_KEY_CODE
Address of a 1-byte **output** area indicating whether the request found more than one record for the supplied key. The possible values are:
- UEP_FC_DUPLICATE KEY
- UEP_FC_NOT_DUPLICATE KEY

UEP_FC_ACCMETH_RETURN_CODE
Address of a 4-byte **output** area in which access-method-dependent information is to be returned when either of the responses UEP_FC_REASON_ACCMETH_REQUEST_ERROR or UEP_FC_REASON_IO_ERROR is returned.

The returned value is placed in bytes 2–5 of the EIBRCODE.

UEP_FC_RESPONSE
Address of a 1-byte **output** area containing the response after a request has completed:

UEP_FC_RESPONSE_OK
Processing has completed without errors.

UEP_FC_RESPONSE_EXCEPTION
Processing has completed with an error condition. The reason is set in UEP_FC_REASON.

UEP_FC_RESPONSE_DISASTER
An error has occurred which prevents processing from completing. Typically, this is as a result of a DISASTER response from an XPI function call, or corruption of data addressed from UEPGAA or UEPTSTOK.

If you set this response, the caller of file control will assume that first-failure data capture has been performed. If you are percolating a DISASTER response from an XPI request, first-failure data capture will have been performed already; if not, you should attempt to capture sufficient information to successfully diagnose the error. The DFHDUDUX SYSTEM_DUMP XPI function may be suitable for this purpose.

UEP_FC_RESPONSE_INVALID
The exit program was invoked with an invalid parameter list, indicating a CICS internal logic error. Note that an invalid parameter list that indicates an application error should give an EXCEPTION response.

UEP_FC_RESPONSE_PURGED
An XPI function call has received a PURGED response. Setting this response is equivalent to setting the UERCPURG return code, except that any changes to the parameter list are honored.

UEP_FC_REASON
Address of a 1-byte **output** area containing, after a request has completed with an EXCEPTION response, the reason. The possible reasons are:

- UEP_FC_REASON_ACCMETH_REQUEST_ERROR
- UEP_FC_REASON_DELETE_AFTER_READ_UPDATE
- UEP_FC_REASON_DELETE_BEFORE_READ_UPDATE
- UEP_FC_REASON_DUPLICATE_READ_UPDATE
- UEP_FC_REASON_DUPLICATE_RECORD
- UEP_FC_REASON_DUPLICATE_REQID
- UEP_FC_REASON_END_OF_FILE
- UEP_FC_REASON_FILE_DISABLED
- UEP_FC_REASON_FILE_NOT_OPEN
- UEP_FC_REASON_FILE_NOT_FOUND
- UEP_FC_REASON_FULL_KEY_WRONG_LENGTH
- UEP_FC_REASON_GENERIC_DELETE_NOT_KSDS
- UEP_FC_REASON_GENERIC_KEY_TOO_LONG
- UEP_FC_REASON_ILLEGAL_KEY_TYPE_CHANGE
- UEP_FC_REASON_INSUFFICIENT_SPACE
- UEP_FC_REASON_INVALID_UPDATE_TOKEN
- UEP_FC_REASON_IO_ERROR
- UEP_FC_REASON_KEY_LENGTH_NEGATIVE
- UEP_FC_REASON_KSDS_AND_XRBA
- UEP_FC_REASON_NO_VARIABLE_LENGTH
- UEP_FC_REASON_NOTAUTH
- UEP_FC_REASON_NOT_EXTENDED
- UEP_FC_REASON_READPREV_IN_GENERIC_BROWSE
- UEP_FC_REASON_RECORD_NOT_FOUND
- UEP_FC_REASON_REWRITE_BEFORE_READ_UPDATE
- UEP_FC_REASON_RIDFLD_KEY_NOT_RECORD_KEY
- UEP_FC_REASON_UNKNOWN_REQID_ENDBR

- UEP_FC_REASON_UNKNOWN_REQID_READNEXT
- UEP_FC_REASON_UNKNOWN_REQID_READPREV
- UEP_FC_REASON_UNKNOWN_REQID_RESETBR

UEP_FC_EXIT_TOKEN
Address of a 4-byte token to be passed to XFCFROUT. This allows you, for example, to pass a work area to exit XFCFROUT.

UEP_FC_M_RECORD_L
Address of a fullword containing the modified record length field. You can use this to change the LENGTH, for remote files only, when the LENGTH field is not specified on the API call. If the LENGTH field is specified on the API call, to change the LENGTH use UEP_FC_RECORD_L instead, because changing the value at the address stored by UEP_FC_M_RECORD_L has no effect.

UEP_FC_M_RECORD_ID_L
Address of a halfword containing the modified key length field. You can use this to change the KEYLENGTH, for remote files only, when the KEYLENGTH field is not specified on the API call. If the KEYLENGTH field is specified, to change the KEYLENGTH use UEP_FC_RECORD_ID_L instead, because changing the value at the address stored by UEP_FC_M_RECORD_ID_L has no effect.

Return codes

UERCNORM
Continue processing.

UERCBYP
Bypass CICS processing of this request. If the exit program has been invoked for a function-shipped request, the mirror transaction is permitted to terminate.

UERCBYPL
Bypass CICS processing of this request. If the exit program has been invoked for a function-shipped request, the mirror transaction must not terminate.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

API and SPI calls
None can be used.

Exit XFCFROUT

Exit XFCFROUT is invoked after the completion of a file control request.

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID.

UEPUSER
Address of the 8-byte user ID.

UEPTERM
Zero, or the address of the 4-byte terminal ID. If no address is returned, this could mean that this request has been function shipped over an IPIC connection.

UEPPROG
Address of the 8-byte application program name.

UEP_FC_FUNCTION
Address of a byte containing the function. The possible values are:
- UEP_FC_FUN_READ_INTO
- UEP_FC_FUN_READ_SET
- UEP_FC_FUN_READ_UPDATE_INTO
- UEP_FC_FUN_READ_UPDATE_SET
- UEP_FC_FUN_WRITE
- UEP_FC_FUN_REWRITE
- UEP_FC_FUN_REWRITE_DELETE
- UEP_FC_FUN_DELETE
- UEP_FC_FUN_UNLOCK
- UEP_FC_FUN_START_BROWSE
- UEP_FC_FUN_READ_NEXT_INTO
- UEP_FC_FUN_READ_NEXT_SET
- UEP_FC_FUN_READ_PREVIOUS_INTO
- UEP_FC_FUN_READ_PREVIOUS_SET
- UEP_FC_FUN_READ_NEXT_UPDATE_INTO
- UEP_FC_FUN_READ_NEXT_UPDATE_SET
- UEP_FC_FUN_READ_PREVIOUS_UPDATE_INTO
- UEP_FC_FUN_READ_PREVIOUS_UPDATE_SET
- UEP_FC_FUN_RESET_BROWSE
- UEP_FC_FUN_END_BROWSE

UEPTSTOK
Address of a 4-byte token that is valid throughout the life of a task. See Using the task token UEPTSTOK.

UEP_FC_FILE_NAME
Address of an 8-byte field containing the filename.

UEP_FC_BUFFER_P
Address of a fullword containing the address of the buffer provided by the originator of the request, in which the record is returned on completion of a READ, READ NEXT, or READ PREV request with the INTO option.

UEP_FC_BUFFER_L
Address of a fullword containing (for READ, READ NEXT, and READ PREV requests) the value of the LENGTH of the buffer into which the record was read.

UEP_FC_RECORD_P
Address of one of the following:
- If the request is a READ, READ NEXT, or READ PREV request with the SET option, a fullword in which is returned the address of a buffer, into which the record was placed.
- If the request is WRITE or REWRITE, a fullword containing the address of the record that was written.

UEP_FC_RECORD_L
Address of a fullword containing (for READ, WRITE, REWRITE, READ NEXT, and READ PREV requests) the value of LENGTH.

For all READ, READ NEXT, or READ PREV requests, this is an output field, containing the actual length of the record read. For these types of request, this record-length value is always present, even if the LENGTH option was not specified on the EXEC CICS API call.

For a WRITE or REWRITE, this is an optional field which, if present, contains the length of the record that was written.

UEP_FC_MAX_RECORD_L
Address of a fullword containing the maximum record length of the file.

UEP_FC_RECORD_ID_P
Address of a fullword containing the address of the value of RIDFLD (record identifier). For a discussion of when the record identifier is an input or an output field, see Table 4 on page 78.

UEP_FC_RECORD_ID_L
Address of the halfword value of KEYLENGTH, which is the (possibly partial) length of the record identifier.

The length of the record identifier is an optional input parameter on READ, WRITE, DELETE, START BR, READ NEXT, READ PREV, and RESET BR requests.

UEP_FC_FULL_RECORD_ID_L
Address of the halfword value of the full length of the record identifier. (The full length of the record identifier corresponds to the KEYLENGTH keyword of the EXEC CICS interface.)

The full length of the record identifier is returned as a mandatory output field on READ NEXT and READ PREV requests.

UEP_FC_RECORD_ID_TYPE
Address of a byte containing (for READ, WRITE, DELETE, START BR, READ NEXT, READ PREV, and RESET BR requests) the RIDFLD type.

UEP_FC_KEY
VSAM KSDS or AIX PATH access

UEP_FC_RBA
VSAM ESDS or KSDS via RBA access

UEP_FC_RRN
VSAM RRDS access

UEP_FC_DEBKEY
BDAM deblocking by key (READ, DELETE, START BR, and RESET BR requests only)

UEP_FC_DEBREC
BDAM deblocking by relative record (READ, DELETE, START BR, and RESET BR requests only)

UEP_FC_XRBA
VSAM extended ESDS access

UEP_FC_REQID
Address (for START BR, READ NEXT, READ PREV, RESET BR, and END BR requests) of the halfword value of REQID.

UEP_FC_NUMREC
Address of the fullword value of NUMREC, in which (if the request is DELETE with RIDFLD) the number of records that have been deleted is returned.

UEP_FC_KEY_COMPARISON
Address of a byte containing (for READ, START BR, and RESET BR requests) the key comparison setting.

UEP_FC_EQUAL
Key-equal-to comparison.

UEP_FC_GTEQ
Key-greater-than-or-equal-to comparison.

UEP_FC_GENERIC
Address of a byte containing (for READ, DELETE, START BR, and RESET BR requests) the generic key setting.

UEP_FC_GENERIC_KEY
Generic key used for key search.

UEP_FC_FULL_KEY
Full key used for key search.

UEP_FC_MASS_INSERT
Address of a byte containing (for WRITE requests) the mass insert setting.

UEP_FC_SEQUENTIAL_WRITE
Sequential mode.

UEP_FC_DIRECT_WRITE
Direct mode.

UEP_FC_READ_INTEGRITY
Address of a byte containing (for non-update READ, READ NEXT, and READ PREV requests) the read integrity setting. (In current versions of CICS, this setting applies only to VSAM RLS.)

UEP_FC_CR
Consistent read integrity.

UEP_FC_FCT_VALUE
Read integrity according to the setting in the FILE definition.

UEP_FC_NRI
No read integrity.

UEP_FC_RR
Repeatable read integrity.

UEP_FC_TOKEN
Address of a fullword containing the value of TOKEN.

If the request is READ, READ NEXT, or READ PREV with update, and the address is not null, the area is an output field in which the token is returned.

If the request is REWRITE, DELETE without RIDFLD, or UNLOCK, the area is an input field.

UEP_FC_SYSID
Address of a 4-byte area that contains the SYSID identifying the remote region. On input to the XFCFRIN exit, the area contained either:
- The value of the SYSID option of the API call, *or*
- Blanks (if SYSID was not specified).

If the XFCFRIN exit redirected the request to a different region, the area contains the SYSID of the remote region.

UEP_FC_LENGTH_ERROR_CODE
Address of a 1-byte area containing the length error code returned after the request completed. The possible values are:
- UEP_FC_LENGTH_OK
- UEP_FC_ BUFFER_LEN_TOO_SMALL
- UEP_FC_RECORD_LEN_TOO_LARGE
- UEP_FC_BUFFER_LEN_NOT_FILE_LEN
- UEP_FC_RECORD_LEN_NOT_FILE_LEN

UEP_FC_DUPLICATE_KEY_CODE
Address of a 1-byte area indicating whether the request found more than one record for the supplied key. The possible values are:
- UEP_FC_DUPLICATE KEY
- UEP_FC_NOT_DUPLICATE KEY

UEP_FC_ACCMETH_RETURN_CODE
Address of a 4-byte area in which access-method-dependent information is returned when either of the responses UEP_FC_REASON_ACCMETH_REQUEST_ERROR or UEP_FC_REASON_IO_ERROR is returned.

UEP_FC_RESPONSE
Address of a 1-byte area containing the response after the request completed:
- UEP_FC_RESPONSE_OK
- UEP_FC_RESPONSE_EXCEPTION
- UEP_FC_RESPONSE_DISASTER
- UEP_FC_RESPONSE_INVALID
- UEP_FC_RESPONSE_PURGED

UEP_FC_REASON
Address of a 1-byte area containing, if the request completed with an EXCEPTION response, the reason. The possible reasons are:
- UEP_FC_REASON_ABEND
- UEP_FC_REASON_ACCMETH_REQUEST_ERROR
- UEP_FC_REASON_BDAM_DELETE
- UEP_FC_REASON_BDAM_LENGTH_CHANGE
- UEP_FC_REASON_BDAM_KEY_CONVERSION
- UEP_FC_REASON_BDAM_READ_PREVIOUS
- UEP_FC_REASON_BDAM_WRITE_MASS_INSERT
- UEP_FC_REASON_BROWSE_UPD_NOT_RLS
- UEP_FC_REASON_CACHE_FAILURE
- UEP_FC_REASON_CFDT_CONNECT_ERROR
- UEP_FC_REASON_CFDT_DISCONNECT_ERROR
- UEP_FC_REASON_CFDT_INVALID_CONTINUATION
- UEP_FC_REASON_CFDT_POOL_FULL

- UEP_FC_REASON_CFDT_REOPEN_ERROR
- UEP_FC_REASON_CFDT_SERVER_NOT_AVAILABLE
- UEP_FC_REASON_CFDT_SERVER_NOT_FOUND
- UEP_FC_REASON_CFDT_SYSIDERR
- UEP_FC_REASON_CFDT_TABLE_GONE
- UEP_FC_REASON_CHANGED
- UEP_FC_REASON_CR_NOT_RLS
- UEP_FC_REASON_DATASET_BEING_COPIED
- UEP_FC_REASON_DEADLOCK_DETECTED
- UEP_FC_REASON_DELETE_AFTER_READ_UPDATE
- UEP_FC_REASON_DELETE_BEFORE_READ_UPDATE
- UEP_FC_REASON_DISASTER_PERCOLATION
- UEP_FC_REASON_DUPLICATE_READ_UPDATE
- UEP_FC_REASON_DUPLICATE_RECORD
- UEP_FC_REASON_DUPLICATE_REQID
- UEP_FC_REASON_END_OF_FILE
- UEP_FC_REASON_ESDS_DELETE
- UEP_FC_REASON_FILE_DISABLED
- UEP_FC_REASON_FILE_NOT_OPEN
- UEP_FC_REASON_FILE_NOT_RECOVERABLE
- UEP_FC_REASON_FILE_NOT_FOUND
- UEP_FC_REASON_FULL_KEY_WRONG_LENGTH
- UEP_FC_REASON_GENERIC_DELETE_NOT_KSDS
- UEP_FC_REASON_GENERIC_KEY_TOO_LONG
- UEP_FC_REASON_ILLEGAL_KEY_TYPE_CHANGE
- UEP_FC_REASON_INSUFFICIENT_SPACE
- UEP_FC_REASON_INVALID_UPDATE_TOKEN
- UEP_FC_REASON_IO_ERROR
- UEP_FC_REASON_ISCINVREQ
- UEP_FC_REASON_ISC_NOT_SUPPORTED
- UEP_FC_REASON_KEY_LENGTH_NEGATIVE
- UEP_FC_REASON_KEY_STOLEN
- UEP_FC_REASON_KSDS_AND_XRBA
- UEP_FC_REASON_LOADING
- UEP_FC_REASON_LOCKED
- UEP_FC_REASON_LOST_LOCKS
- UEP_FC_REASON_LOCK_STRUCTURE_FULL
- UEP_FC_REASON_NOT_IN_SUBSET
- UEP_FC_REASON_NO_VARIABLE_LENGTH
- UEP_FC_REASON_NOSUSPEND_NOT_RLS
- UEP_FC_REASON_NOTAUTH
- UEP_FC_REASON_NOT_EXTENDED
- UEP_FC_REASON_PREVIOUS_RLS_FAILURE
- UEP_FC_REASON_RBA_ACCESS_TO_RLS_KSDS
- UEP_FC_REASON_READ_NOT_AUTHORISED
- UEP_FC_REASON_READPREV_IN_GENERIC_BROWSE
- UEP_FC_REASON_RECLEN_EXCEEDS_LOGGER_BFSZ
- UEP_FC_REASON_RECORD_BUSY
- UEP_FC_REASON_RECORD_NOT_FOUND
- UEP_FC_REASON_REMOTE_INVREQ
- UEP_FC_REASON_RESTART_FAILED
- UEP_FC_REASON_REWRITE_BEFORE_READ_UPDATE
- UEP_FC_REASON_RIDFLD_KEY_NOT_RECORD_KEY
- UEP_FC_REASON_RLS_DEADLOCK_DETECTED
- UEP_FC_REASON_RLS_DISABLED
- UEP_FC_REASON_RLS_FAILURE

- UEP_FC_REASON_RR_NOT_RLS
- UEP_FC_REASON_SECURITY_FAILURE
- UEP_FC_REASON_SELF_DEADLOCK_DETECTED
- UEP_FC_REASON_SERVREQ_VIOLATION
- UEP_FC_REASON_SHIP
- UEP_FC_REASON_SHIPPED_SECURITY_FAILURE
- UEP_FC_REASON_STORE_FAIL
- UEP_FC_REASON_SUPPRESSED
- UEP_FC_REASON_SYSIDERR
- UEP_FC_REASON_TABLE_FULL
- UEP_FC_REASON_TABLE_TOKEN_INVALID
- UEP_FC_REASON_TIMEOUT
- UEP_FC_REASON_TOO_MANY_CFDTS_IN_UOW
- UEP_FC_REASON_UNKNOWN_REQID_ENDBR
- UEP_FC_REASON_UNKNOWN_REQID_READNEXT
- UEP_FC_REASON_UNKNOWN_REQID_READPREV
- UEP_FC_REASON_UNKNOWN_REQID_RESETBR
- UEP_FC_REASON_UPDATE_NOT_AUTHORISED

UEP_FC_EXIT_TOKEN
Address of the 4-byte token passed from XFCFRIN.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

API and SPI calls
None can be used.

File control EXEC interface API exits XFCREQ and XFCREQC

The XFCREQ exit allows you to intercept a file control application programming interface (API) request before any action has been taken on it by file control. The XFCREQC exit allows you to intercept a file control API request after file control has completed its processing.

Important

The XFCREQ and XFCREQC exits are not called, on the target region, for function-shipped requests. That is, if a file control API request is function-shipped to a remote region, the exits are not called on the remote region. To intercept a function-shipped file control API request on the target region, use the XFCFRIN exit—see "File control domain exits, XFCFRIN and XFCFROUT" on page 59.

The file control API commands intercepted are:

- READ
- WRITE
- REWRITE
- DELETE
- UNLOCK

- STARTBR
- READNEXT
- READPREV
- ENDBR
- RESETBR.

The XFCREQ and XFCREQC exits can be written only in assembler language.

Using XFCREQ, you can:

- Analyze the request, to determine its type, the keywords specified, and their values.
- Modify values specified by the request before the command is executed.
- Set return codes to specify that either:
  - CICS should continue with the (possibly modified) request.
  - CICS should bypass the request. (Note that if you set this return code, you must also set up return codes for the EXEC interface block (EIB), as if you had processed the request yourself.)

Using XFCREQC, you can:

- Analyze the request, to determine its type, the keywords specified, and their values.
- Set return codes for the EIB.

Both exits are passed nine parameters as follows:

- The address of the command-level parameter structure
- The address of a token (UEPFCTOK) used to pass 4 bytes of data from XFCREQ to XFCREQC
- The addresses of copies of four pieces of return code and resource information from the EIB
- The address of a token (UEPTSTOK) that is valid throughout the life of a task
- The address of a recursion count field
- The address of a 16-byte area that is used if the request has been function shipped.

The command-level parameter structure

The command-level parameter structure consists of a series of addresses. The first address points to the EXEC interface descriptor (EID), which consists of a bit string that describes the type of request and identifies each keyword specified with the request. The remaining addresses point to pieces of data associated with the request. For example, the second address always points to the file name.

Only the first 8 addresses and the last address can be referenced by the user exit. The ninth through eleventh addresses are reserved for CICS internal use.

You can examine the EID to determine the type of request and the keywords specified. You can examine the other parameters in the list to determine the values of the keywords. You can also modify values of keywords specified on the request. (For example, you could change the name of the file involved in the request.)

End of parameter list indicator

The high-order bit is set on in the last address set in the parameter list to indicate that it is the last one in the list. On return from your user exit program, CICS scans the parameter list for the high-order bit to find the last parameter. Therefore, if you modify the length of the parameter list, you must also reset the high-order bit to indicate which is the new last address.

The original parameter list, as it was before XFCREQ was invoked, is restored after the completion of XFCREQC. It follows that the execution diagnostic facility (EDF) displays the original command before and after execution.

Note: EDF does not display any changes made by the exit.

The UEPCLPS exit-specific parameter

The UEPCLPS exit-specific parameter is the address of the command-level parameter structure, and is included in exits XFCREQ and XFCREQC.

The command-level parameter structure contains 12 addresses, FC_ADDR0 through FC_ADDRB. It is defined in the DSECT FC_ADDR_LIST, which you should copy into your exit program by including the statement COPY DFHFCEDS.

The command-level parameter list is made up as follows:

FC_ADDR0
is the address of a 9-byte area called the EID, which is made up as follows:

- **FC_GROUP**
- **FC_FUNCT**
- **FC_BITS1**
- **FC_BITS2**
- **FC_EIDOPT5**
- **FC_EIDOPT6**
- **FC_EIDOPT7**
- **FC_EIDOPT8**

The name of the DSECT mapping the EID is FC_EID.

FC_GROUP
Always X'06', indicating that this is a file control request.

FC_FUNCT
One byte that defines the type of request:

X'02' READ

X'04' WRITE

X'06' REWRITE

X'08' DELETE

X'0A' UNLOCK

X'0C' STARTBR

X'0E' READNEXT

X'10' READPREV

X'12' ENDBR

X'14' RESETBR

FC_BITS1
Existence bits that define which keywords that contain values were specified. To obtain the value associated with a keyword, you need to use the appropriate address from the command-level parameter structure. Before using this address, you must check the associated existence bit. If the existence bit is set off, the keyword was not specified in the request and the address should not be used.

X'80' Set if the request contains the keyword FILE. If set, **FC_ADDR1** is meaningful.

X'40' Set if the request contains any of the keywords INTO, SET, or FROM. If set, **FC_ADDR2** is meaningful.

X'20' Set if the request specifies LENGTH or NUMREC, or if a STARTBR, RESETBR, or ENDBR request specifies REQID. If set, **FC_ADDR3** is meaningful.

X'10' Set if the request specifies RIDFLD. If set, **FC_ADDR4** is meaningful.

X'08' Set if the request specifies KEYLENGTH. If set, **FC_ADDR5** is meaningful.

X'04' Set if the request is READNEXT or READPREV and specifies REQID. If set, **FC_ADDR6** is meaningful.

X'02' Set if the request specifies SYSID. If set, **FC_ADDR7** is meaningful.

X'01' Not used by file control.

FC_BITS2
Second set of existence bits.

X'20' Set if the request specifies TOKEN. If set, FC_ADDRB is meaningful.

FC_EIDOPT5
Indicates whether certain keywords that do not take values were specified on the request.

X'04' MASSINSERT specified.

X'02' RRN specified.

X'01' SET (and not INTO) was specified.

Note: Your program must test for keywords at the bit level, because there may be more than one of these keywords present.

FC_EIDOPT6
Indicates whether certain keywords that do not take values were specified on the request.

X'80' RBA specified.

X'40' GENERIC specified.

X'20' GTEQ specified.

X'10' UNCOMMITTED specified.

X'08' CONSISTENT specified.

X'04' REPEATABLE specified.

X'01' NOSUSPEND specified (on READ, READNEXT, READPREV, WRITE, DELETE, or REWRITE).

Note:

1. If the read integrity bits (for UNCOMMITTED, CONSISTENT, and REPEATABLE) are off (zero) on the command, the read integrity options specified on the file resource definition are used. If you need to know what these are, you can issue an EXEC CICS INQUIRE FILE command.
2. Your program must test for keywords at the bit level, because there may be more than one of these keywords present.

FC_EIDOPT7

Indicates whether certain keywords that do not take values were specified on the request.

X'04' UPDATE specified. This setting is meaningful only for READ requests. For other requests, X'04' may or may not be set.

X'01' Either DEBREC or DEBKEY specified (see **FC_EIDOPT8**). This setting is meaningful only for READ requests. For other requests, X'01' may or may not be set.

Note: Your program must test for keywords at the bit level, because there may be more than one of these keywords present.

FC_EIDOPT8

Indicates whether certain keywords that do not take values were specified on the request.

X'80' DEBKEY specified.

X'40' DEBREC specified.

X'20' TOKEN specified.

X'08' XRBA specified. If the XRBA bit is on, FC_RIDFLD (described in DSECT DFHFCEDS) points to an 8-byte extended relative byte address (XRBA).

FC_ADDR1

is the address of an 8-byte area containing the name specified on the FILE keyword.

FC_ADDR2

is the address of one of the following:

- A 4-byte address returned for SET (if the request is READ, READNEXT, or READPREV, and if **FC_EIDOPT5** indicates that this is SET).
- Data returned for INTO (if the request is READ, READNEXT, or READPREV, and if **FC_EIDOPT5** indicates that this is not SET).
- Data from FROM (if the request is WRITE or REWRITE).

FC_ADDR3

is the address of one of the following:

- The halfword value of LENGTH (if the request is READ, WRITE, REWRITE, READNEXT, or READPREV).

Warning: For requests that specify INTO, do not change the value of LENGTH to a value greater than that specified by the application. To do so causes a storage overlay in the application.

- The returned halfword value of NUMREC (if the request is DELETE).
- The halfword value of REQID (if the request is STARTBR, RESETBR, or ENDBR).

FC_ADDR4
is the address of an area containing the value of the RIDFLD keyword.

FC_ADDR5
is the address of the halfword value of KEYLENGTH.

FC_ADDR6
is the address of the halfword value of REQID (if the request is READNEXT or READPREV).

FC_ADDR7
is the address of an area containing the value of SYSID.

FC_ADDR8
is the address of a value intended for CICS internal use only. It must not be used.

FC_ADDR9
is the address of a value intended for CICS internal use only. It must not be used.

FC_ADDRA
is the address of a value intended for CICS internal use only. It must not be used.

FC_ADDRB
is the address of the fullword value of TOKEN (if the request is READ, READNEXT, READPREV, REWRITE, DELETE, or UNLOCK).

Modifying fields in the command-level parameter structure

Some fields that are passed to file control are used as input to the request, some are used as output fields, and some are used for both input and output. The method your user exit program uses to modify a field depends on the usage of the field.

A list of input and output fields

The following are always input fields:

- FILE
- FROM
- KEYLENGTH
- REQID
- SYSID

The following are always output fields:

- INTO
- NUMREC
- SET

Whether LENGTH and RIDFLD are input or output fields depends on the request, as shown in Table 4. A dash (—) means that the keyword cannot be specified on the request.

Table 4. LENGTH and RIDFLD as input and output fields

Request	LENGTH	RIDFLD
READ	Output	See Note 1.
WRITE	Input	See Note 2.
REWRITE	Input	—
DELETE	—	See Note 3.
UNLOCK	—	—
STARTBR	—	Input
READNEXT	Output	Output
READPREV	Output	Output
ENDBR	—	—
RESETBR	—	Input

Note:

1. Normally, this is an input field. However, if UPDATE is specified and the file is a BDAM file using extended key search, RIDFLD is used for both input and output.
2. The use of RIDFLD on a WRITE request depends on the file type. For a VSAM KSDS or RRDS, or a fixed-format BDAM file, RIDFLD is an input field. For all other file types, it is used either for output only, or for both input and output, and should be treated like an output field.
3. RIDFLD is an input field on DELETE requests that are not preceded by a READ UPDATE. It is not specified on requests that are preceded by a READ UPDATE.

Modifying input fields

The correct method of modifying an input field is to create a new copy of it, and to change the address in the command-level parameter list to point to your new data.

Note: You must never modify an input field by altering the data that is pointed to by the command-level parameter list. To do so would corrupt storage belonging to the application program and would cause a failure when the program attempted to reuse the field.

Modifying output fields

The technique described in “Modifying input fields” is not suitable for modifying output fields. (The results would be returned to the new area instead of the application’s area, and would be invisible to the application.)

An output field is modified by altering the data that is pointed to by the command-level parameter list. In the case of an output field, you can modify the application’s data in place, because the application is expecting the field to be modified anyway.

Modifying fields used for both input and output

An example of a field that is used for both input and output is LENGTH on a READ request that specifies INTO. You can treat such fields in the same way as output fields, and they are considered to be the same.

Modifying the EID

It is not possible to modify the EID to make major changes to requests. It is not possible, for example, to change a WRITE request to a READ request. However, you can make minor changes to requests, such as to turn on the existence bit for SYSID so that the request can be changed into one that is shipped to a remote system.

The list that follows shows the bits in the EID that can be modified. Any attempt to modify any other part of the EID is ignored.

FC_BITS1

- **X'20'** The existence bit for LENGTH, NUMREC, or (if the request is STARTBR, RESETBR, or ENDBR) REQID.
- **X'08'** The existence bit for KEYLENGTH.
- **X'04'** The existence bit for REQID if the request is READNEXT or READPREV.
- **X'02'** The existence bit for SYSID.

FC_BITS2

- **X'20'** Token specified.

FC_EIDOPT5

- **X'04'** MASSINSERT specified.

FC_EIDOPT6

- **X'40'** GENERIC specified.
- **X'20'** GTEQ specified.
- **X'10'** UNCOMMITTED specified.
- **X'08'** CONSISTENT specified.
- **X'04'** REPEATABLE specified.
- **X'02'** UPDATE specified on READNEXT or READPREV.
- **X'01'** NOSUSPEND specified (on READ, READNEXT, READPREV, WRITE, DELETE, or REWRITE).

Bits in the EID should be modified in place. You should not modify the pointer to the EID: any attempt to do so is ignored by CICS.

The EID is reset to its original value before return to the application program. That is, changes made to the EID are retained for the duration of the file control request only.

If more than one of UNCOMMITTED, CONSISTENT, or REPEATABLE is specified, CONSISTENT takes precedence over UNCOMMITTED, and REPEATABLE takes precedence over CONSISTENT and UNCOMMITTED.

Example of modifying read integrity bits

You might want all RLS read requests from all programs against a specific file to specify CONSISTENT read. You could code a user exit program that turns on the bit for CONSISTENT and turns off the other two read integrity bits in all requests to the file. You could partially achieve this effect by specifying CONSISTENT on the FILE definition. However, that would only override requests that did not explicitly specify a level of read integrity. Using a global user exit program for this purpose also overrides programs that explicitly specify UNCOMMITTED or REPEATABLE.

Warnings:

1. If a global user exit program changes a file request to request a higher level of read integrity (for example, it changes the request from UNCOMMITTED to REPEATABLE), this could cause CICS either to acquire extra read locks, or to keep its read locks for a longer period of time. This may degrade system throughput, by causing other transactions to wait, or introduce deadlocks.
2. If a global user exit program changes the request to one that requests a lower level of read integrity (for example, it changes the request from REPEATABLE to UNCOMMITTED), this could cause application logic errors to occur in the program that originated the request. The errors could occur because the application program may be relying on the record to remain unchanged while it reads a series of other, related, records. This can be guaranteed with REPEATABLE, but not if the option is changed to UNCOMMITTED.
3. Your user exit program is prevented from making major changes to the EID. However, you must take great care when making the minor modifications that **are** permitted. For instance, it is possible to change a DELETE into a GENERIC DELETE, but to make such a change may be dangerous.

Use of the parameter UEPFSHIP

UEPFSHIP contains the address of a 16-byte area. This area consists of 4 characters, followed by 3 fullwords.

If the first byte contains 'Y', this request has been function shipped to this region. In this case, if your exit program wants to bypass file control (by setting a return code of UERCBYP), it must set the 3 fullwords as follows:

Fullword 1
The length of the buffer area

Fullword 2
The length of the record

Fullword 3
The length of the modified RIDFLD.

Doing this ensures that the data and RIDFLD are correctly shipped back.

EIB (EXEC interface block)

Copies of EIBRSRCE, EIBRCODE, EIBRESP, and EIBRESP2 are passed to the exit so you can modify/set the completion and resource information in XFCREQ and XFCREQC, and examine completion and resource information in XFCREQC.

You can update the copies of EIBRSRCE, EIBRCODE, EIBRESP, and EIBRESP2 that you are given in the parameter list. File Control copies your values into the real EIB after the completion of XFCREQC; or if you specify a return code of ‘bypass' in XFCREQ.

You must set valid file control responses. You must set all three of EIBRCODE, EIBRESP, and EIBRESP2 to a consistent set of values, such as would be set by File Control to describe a valid completion. **File Control does not police the consistency of EIBRCODE, EIBRESP, and EIBRESP2**. To aid you in setting the values of EIBRCODE, EIBRESP, and EIBRESP2, the values used by File Control are specified in DFHFCEDS.

Example of how XFCREQ and XFCREQC can be used

In this example, XFCREQ and XFCREQC are used to obtain a record containing compressed data, to decompress the data, and to return it to the area specified by the user program as INTO. The example shows only the capabilities of the exits; it is not intended to indicate an ideal way of achieving the function.

In XFCREQ:

1. Issue an EXEC CICS GETMAIN to obtain an area large enough to hold the decompressed data.
2. Change the INTO pointer to point to this new area, so that File Control uses it when it processes the request. (The decompressed data is copied to the *user's* INTO area, and the INTO pointer reset, before return to the application program—see stages 4 and 7 of the processing to be done by XFCREQC.)
3. Set UEPFCTOK to be the address of the new area so that XFCREQC can also use this area.
4. Return to CICS.

In XFCREQC:

1. Check 'UEPRCODE' to make sure that the file control request completed without error.
2. Use UEPFCTOK to find the address of the area. This area now holds the compressed data.
3. Decompress the data in place.
4. Copy the data from the new area to the user's INTO area. Use the user-specified LENGTH (from the command-level parameter list) to ensure that the data fits and that the copy does not cause a storage violation.
5. Set 'LENGERR' in UEPRESP, UEPRESP2, and UEPRCODE if the data does not fit.
6. Use EXEC CICS FREEMAIN to free the work area pointed to by UEPFCTOK.
7. At this point the command-level parameter list points to the now free area as the address for INTO. This is not a problem, because after completion of XFCREQC File Control restores this pointer to point to the area supplied by the user program.
8. Return to CICS.

Exit XFCREQ

Exit XFCREQ is invoked before CICS processes a file control API request. The exit is not invoked, on the target region, for function-shipped requests.

Exit-specific parameters

UEPCLPS
Address of the command-level parameter structure. See "The UEPCLPS exit-specific parameter" on page 74.

UEPFCTOK
Address of the 4-byte token to be passed to XFCREQC. This allows you, for example, to pass a work area to exit XFCREQC.

UEPRCODE
Address of a 6-byte hexadecimal copy of the EIB return code 'EIBRCODE'. For details of EIB return codes, refer to EIB fields.

UEPRESP
Address of a 4-byte binary copy of the EIB response code 'EIBRESP'.

UEPRESP2
Address of a 4-byte binary copy of the EIB response code 'EIBRESP2'.

UEPTSTOK
Address of a 4-byte token that is valid throughout the life of a task. See Using the task token UEPTSTOK.

UEPRECUR
Address of a halfword recursion counter. The counter is set to 0 when the exit is first invoked, and is incremented for each recursive call.

UEPFSHIP
Address of a 16 byte area. See "Use of the parameter UEPFSHIP" on page 80.

UEPRSRCE
Address of an 8-character copy of the EIB resource value, EIBRSRCE.

Return codes

UERCNORM
Continue processing.

UERCBYP
The file control EXEC interface program should ignore this request.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

Although the exit permits the use of XPI GETMAIN and FREEMAIN calls, we recommend that you use the EXEC CICS GETMAIN and FREEMAIN commands instead.

API and SPI calls
All can be used, except for:
EXEC CICS SHUTDOWN
EXEC CICS XCTL

Note:

1. Take care when issuing recursive commands not to cause a loop. For example, it is your responsibility to avoid entering a loop when a file control request is issued from the XFCREQ exit. Use of the recursion counter UEPRECUR is recommended.

2. Exit programs that issue EXEC CICS commands must first address the EIB. See Using CICS services.
3. Exit programs that issue EXEC CICS commands, and that use the DFHEIENT macro, should use the DFHEIRET macro to set a return code and return to CICS. See Global user exit XPI examples, showing the use of storage.

Exit XFCREQC

Exit XFCREQC is invoked after a file control API request has completed, and before return from the file control EXEC interface program. The exit is not invoked, on the target region, for function-shipped requests.

Exit-specific parameters

UEPCLPS
Address of the command-level parameter structure. See "The UEPCLPS exit-specific parameter" on page 74.

UEPFCTOK
Address of the 4 byte token passed from XFCREQ.

UEPRCODE
Address of a 6-byte hexadecimal copy of the EIB return code EIBRCODE. For details of EIB return codes, refer to EIB fields.

UEPRESP
Address of a 4-byte binary copy of the EIB response code EIBRESP.

Note: If the file that has just been accessed is remote, the addressed field contains zeros (even if UEPRCODE is non-zero).

UEPRESP2
Address of a 4-byte binary copy of the EIB response code EIBRESP2.

Note: If the file that has just been accessed is remote, the addressed field contains zeros (even if UEPRCODE is non-zero).

UEPTSTOK
Address of a 4-byte token that is valid throughout the life of a task. See Using the task token UEPTSTOK.

UEPRECUR
Address of a halfword recursion counter. The counter is set to 0 when the exit is first invoked, and is incremented for each recursive call.

UEPRSRCE
Address of an 8-character copy of the EIB resource value, EIBRSRCE.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

Although the exit permits the use of XPI GETMAIN and FREEMAIN calls, we recommend that you use the EXEC CICS GETMAIN and FREEMAIN commands instead.

API and SPI calls

All can be used, except for:

EXEC CICS SHUTDOWN
EXEC CICS XCTL

Note:

1. Take care when issuing recursive commands not to cause a loop. For example, it is your responsibility to avoid entering a loop when a file control request is issued from the XFCREQC exit. Use of the recursion counter UEPRECUR is recommended.
2. Exit programs that issue EXEC CICS commands must first address the EIB. See Using CICS services.
3. Exit programs that issue EXEC CICS commands, and that use the DFHEIENT macro, should use the DFHEIRET macro to set a return code and return to CICS. See Global user exit XPI examples, showing the use of storage.

Example program

CICS supplies, in CICSTS54.CICS.SDFHSAMP, an example program, DFH$XTSE, that shows how to modify fields in the command-level parameter structure passed to EXEC interface exits.

File control EXEC interface SPI exits XFCAREQ and XFCAREQC

The XFCAREQ exit allows you to intercept a file control system programming interface (SPI) request before any action has been taken on it by file control. The XFCAREQC exit allows you to intercept the response after a file control SPI request has completed.

The file control SPI requests intercepted are:

- **EXEC CICS INQUIRE FILE**
- **EXEC CICS SET FILE**.

Using XFCAREQ, you can:

- Analyze the SPI parameter list (function, keywords, argument values, and responses)
- Modify any input parameter before execution of the request
- Prevent execution of a request and set appropriate responses.

Using XFCAREQC, you can:

- Analyze the SPI parameter list
- Modify any output parameter value and set responses after execution.

You can also:

- Pass data between your XFCAREQ and XFCAREQC exit programs when they are invoked for the same request.
- Pass data between your file control exit programs when they are invoked within the same task. You can pass data between successive invocations of XFCAREQ and XFCAREQC and also between invocations of other EXEC-enabled user exits.

If you make changes to file states (that is, if you open, close, enable, or disable a file) it is possible that exits in the file state change program (XFCSREQ and XFCSREQC) could modify situations set up by XFCAREQ. Therefore you must consider the order in which the exits are invoked. If all four exits are enabled, the order of invocation is as follows:

- For the **SET FILE** command:
  1. XFCAREQ
  2. XFCSREQ
  3. XFCSREQC
  4. XFCAREQC
- For the **INQUIRE FILE** command:
  1. XFCAREQ
  2. XFCAREQC

Exit XFCAREQ

This exit is invoked before CICS processes a file control SPI request.

When invoked

Before CICS processes a file control SPI request.

Exit-specific parameters

UEPCLPS

Address of a copy of the SPI command parameter list. See "The command-level parameter structure" on page 87.

UEPFATOK

Address of a 4-byte area that can be used to pass information between XFCAREQ and XFCAREQC on a single file control SPI request.

UEPRCODE

Address of a 6-byte hexadecimal copy of the EIB return code EIBRCODE. For details of EIB return codes, see EIB fields.

UEPRESP

Address of a 4-byte binary copy of the EIB response code EIBRESP.

UEPRESP2

Address of a 4-byte binary copy of the EIB response code EIBRESP2.

UEPTSTOK

Address of a 4-byte token which can be used to pass information between successive file control requests within the same task (for example, between successive invocations of the XFCAREQC exit). See Using the task token UEPTSTOK.

UEPRECUR

Address of a halfword recursion counter. The counter is set to zero when the exit is first invoked and is incremented for each recursive call.

Return codes

UERCBYP

Bypass this request.

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI commands
All can be used.

API and SPI commands
All can be used, except for:
EXEC CICS SHUTDOWN
EXEC CICS XCTL

Note: Take care when using recursive commands. For example, you must avoid entering a loop when issuing a file control SPI request from the XFCAREQ exit. Use of the recursion counter UEPRECUR is recommended.

Exit XFCAREQC

Exit XFCAREQC is invoked after a file control SPI request has completed, before return from the file control SPI EXEC interface program.

Exit specific parameters:

UEPCLPS
Address of a copy of the API command parameter list. See "The command-level parameter structure" on page 87.

UEPFATOK
Address of a 4-byte area that can be used to pass information between XFCAREQ and XFCAREQC on a single file control SPI request.

UEPRCODE
Address of a 6-byte hexadecimal copy of the EIB return code EIBRCODE. For details of EIB return codes, see EIB fields.

UEPRESP
Address of a 4-byte binary copy of the EIB response code EIBRESP.

UEPRESP2
Address of a 4-byte binary copy of the EIB response code EIBRESP2.

UEPTSTOK
Address of a 4-byte token which can be used to pass information between successive file control requests within the same task (for example, between successive invocations of the XFCAREQC exit). See Using the task token UEPTSTOK.

UEPRECUR
Address of a halfword recursion counter. The counter is set to zero when the exit is first invoked and is incremented for each recursive call.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI commands
All can be used.

API and SPI commands
All can be used, except for:
EXEC CICS SHUTDOWN
EXEC CICS XCTL

You can update the copies of EIBRCODE, EIBRESP, and EIBRESP2 that you are given in the parameter list. If you update the values, file control copies the new values into the application program's EXEC interface block (EIB) after the completion of XFCAREQC or if you specify a return code of UERCBYP in XFCAREQ.

You must set valid file control responses. You must set all three of EIBRCODE, EIBRESP, and EIBRESP2 to a consistent set of values, such as would be set by file control to describe a valid completion. CICS does not check the consistency of the values you set. If EIBRCODE is set to a non-zero value and EIBRESP is set to zero, CICS overrides EIBRESP with a non-zero value. To help you set values for EIBRCODE, EIBRESP, and EIBRESP2, the values used by file control for SPI requests are specified in DSECT DFHFAUED.

Note: Take care when using recursive commands. For example, you must avoid entering a loop when issuing a file control SPI request from the XFCAREQ exit. Use of the recursion counter UEPRECUR is recommended.

The command-level parameter structure

The command-level parameter structure consists of a series of addresses. The first address points to the EXEC interface descriptor (EID), which consists of a bit string that describes the type of request and identifies each keyword specified with the request. The remaining addresses point to pieces of data associated with the request.

You can examine the EID to determine the type of request and the keywords specified. You can examine the other parameters in the list to determine the values of the keywords. You can also modify values of keywords specified on the request.

Note: The relationship between arguments, keywords, data types, and input/output types on the file control SPI commands is summarized in the following tables:

- For INQUIRE FILE, see Table 5 on page 94.
- For SET FILE, see Table 6 on page 95.

The UEPCLPS exit-specific parameter

The UEPCLPS exit-specific parameter is passed to both XFCAREQ and XFCAREQC and contains the address of the command-level parameter structure.

The command-level parameter list contains 64 addresses, FCIS_ADDR0 through FCIS_ADDR63. These are described in DSECT DFHFAUED, which you should copy into your program by including the statement COPY DFHFAUED.

The command-level parameter list is made up as follows:

FCIS_ADDR0
is the address of an 13-byte area called the EID which is made up as follows:

- **FCIS_GROUP**

- **FCIS_FUNCT**
- **FCIS_EIDOPT2**
- **FCIS_EIDOPT3**
- **FCIS_EIDOPT4**
- **FCIS_BITS1**
- **FCIS_BITS2**
- **FCIS_BITS3**
- **FCIS_BITS4**
- **FCIS_BITS5**
- **FCIS_BITS6**
- **FCIS_BITS7**
- **FCIS_BITS8**

FCIS_GROUP
Always X'4C', indicating that this is a file control SPI request.

FCIS_FUNCT
One byte that defines the type of request:

X'02' INQUIRE FILE

X'04' SET FILE.

FCIS_EIDOPT2
Not used by file control.

FCIS_EIDOPT3
Not used by file control.

FCIS_EIDOPT4
Not used by file control.

FCIS_BITS1
Existence bits which specify which arguments were specified. To obtain the argument associated with a keyword, you need to obtain the appropriate address from the command-level parameter structure. Before using this address you must check the associated existence bit. If the existence bit is set off, the argument was not specified in the request and the address should not be used.

X'80' Set if the request contains an argument for the FILE keyword. If set, FCIS_ADDR1 is meaningful.

X'40' Set if the request contains an argument for the DSNAME keyword. If set, FCIS_ADDR2 is meaningful.

X'20' Set if the request contains an argument for the FWDRECSTATUS keyword. If set, FCIS_ADDR3 is meaningful.

X'10' Set if the request contains an argument for the STRINGS keyword. If set, FCIS_ADDR4 is meaningful.

X'08' Set if the request contains an argument for the BASEDSNAME keyword. If set, FCIS_ADDR5 is meaningful.

X'04' Set if the request contains an argument for the LSRPOOLID keyword. If set, FCIS_ADDR6 is meaningful.

X'02' Set if the request contains an argument for the READ keyword. If set, FCIS_ADDR7 is meaningful.

X'01' Set if the request contains an argument for the UPDATE keyword. If set, FCIS_ADDR8 is meaningful.

FCIS_BITS2

Existence bits which specify which arguments were specified. The comments for FCIS_BITS1 also apply to FCIS_BITS2.

X'80' Set if the request contains an argument for the BROWSE keyword. If set, FCIS_ADDR9 is meaningful.

X'40' Set if the request contains an argument for the ADD keyword. If set, FCIS_ADDR10 is meaningful.

X'20' Set if the request contains an argument for the DELETE keyword. If set, FCIS_ADDR11 is meaningful.

X'10' Set if the request contains an argument for the DISPOSITION keyword. If set, FCIS_ADDR12 is meaningful.

X'08' Set if the request contains an argument for the EMPTYSTATUS keyword. If set, FCIS_ADDR13 is meaningful.

X'04' Set if the request contains an argument for the OPENSTATUS keyword. If set, FCIS_ADDR14 is meaningful.

X'02' Set if the request contains an argument for the ENABLESTATUS keyword. If set, FCIS_ADDR15 is meaningful.

X'01' Set if the request contains an argument for the RECOVSTATUS keyword. If set, FCIS_ADDR16 is meaningful.

FCIS_BITS3

Existence bits which specify which arguments were specified. The comments for FCIS_BITS1 also apply to FCIS_BITS3.

X'80' Set if the request contains an argument for the ACCESSMETHOD keyword. If set, FCIS_ADDR17 is meaningful.

X'40' Set if the request contains an argument for the TYPE keyword. If set, FCIS_ADDR18 is meaningful.

X'20' Set if the request contains an argument for the OBJECT keyword. If set, FCIS_ADDR19 is meaningful.

X'10' Set if the request contains an argument for the REMOTESYSTEM keyword. If set, FCIS_ADDR20 is meaningful.

X'08' Set if the request contains an argument for the REMOTENAME keyword. If set, FCIS_ADDR21 is meaningful.

X'04' Set if the request contains an argument for the RECORDFORMAT keyword. If set, FCIS_ADDR22 is meaningful.

X'02' Set if the request contains an argument for the BLOCKFORMAT keyword. If set, FCIS_ADDR23 is meaningful.

X'01' Set if the request contains an argument for the KEYLENGTH keyword. If set, FCIS_ADDR24 is meaningful.

FCIS_BITS4

Existence bits which specify which arguments were specified. The comments for FCIS_BITS1 also apply to FCIS_BITS4.

X'80' Set if the request contains an argument for the KEYPOSITION keyword. If set, FCIS_ADDR25 is meaningful.

X'40' Set if the request contains an argument for the RECORDSIZE keyword. If set, FCIS_ADDR26 is meaningful.

X'20' Set if the request contains an argument for the RELTYPE keyword. If set, FCIS_ADDR27 is meaningful.

X'10' Set if the request contains an argument for the EXCLUSIVE keyword. If set, FCIS_ADDR28 is meaningful.

X'08' Set if the request contains an argument for the BLOCKKEYLEN keyword. If set, FCIS_ADDR29 is meaningful.

X'04' Set if the request contains an argument for the BLOCKSIZE keyword. If set, FCIS_ADDR30 is meaningful.

X'02' Not used by file control.

X'01' Not used by file control.

FCIS_BITS5

Existence bits which specify which arguments were specified. The comments for FCIS_BITS1 also apply to FCIS_BITS5.

X'80' Set if the request contains an argument for the TABLE keyword. If set, FCIS_ADDR33 is meaningful.

X'40' Set if the request contains an argument for the MAXNUMRECS keyword. If set, FCIS_ADDR34 is meaningful.

X'20' Set if the request contains an argument for the READINTEG keyword. If set, FCIS_ADDR35 is meaningful.

X'10' Set if the request contains an argument for the RLSACCESS keyword. If set, FCIS_ADDR36 is meaningful.

X'08' Set if the request contains an argument for the DEFINESOURCE keyword. If set, FCIS_ADDR37 is meaningful.

X'04' Set if the request contains an argument for the INSTALLAGT keyword. If set, FCIS_ADDR38 is meaningful.

X'02' Set if the request contains an argument for the INSTALLUSR keyword. If set, FCIS_ADDR39 is meaningful.

X'01' Set if the request contains an argument for the CHANGEAGENT keyword. If set, FCIS_ADDR40 is meaningful.

FCIS_BITS6

Specifies whether certain keywords were specified on the File control SPI command.

X'80' Set if the request contains the START keyword.

X'40' Set if the request contains the NEXT keyword.

X'20' Set if the request contains the END keyword.

X'10' Set if the request contains the WAIT keyword.

X'08' Set if the request contains the NOWAIT keyword.

X'04' Set if the request contains the FORCE keyword.

X'02' Set if the request contains the ENABLED keyword.

X'01' Set if the request contains the DISABLED keyword.

FCIS_BITS7
Specifies whether certain keywords were specified on the File control SPI command. Also contains the existence bit for JOURNALNUM.

X'80' Set if the request contains the OPEN keyword.

X'40' Set if the request contains the CLOSED keyword.

X'20' Set if the request contains the EMPTY keyword.

X'10' Set if the request contains an argument for the JOURNALNUM keyword. If set, FCIS_ADDR52 is meaningful.

X'08' Set if the request contains the LOADTYPE keyword.

X'04' Set if the request contains the POOL keyword.

X'02' Set if the request contains the TABLENAME keyword.

X'01' Set if the request contains the UPDATEMODEL keyword.

FCIS_BITS8

X'80' Set if the request contains the REMOTETABLE keyword.

X'40' Not used by file control.

X'20' Set if the request contains an argument for the CHANGEUSRID keyword. If set, FCIS_ADDR59 is meaningful.

X'10' Set if the request contains an argument for the CHANGEAGREL keyword. If set, FCIS_ADDR60 is meaningful.

X'08' Set if the request contains an argument for the DEFINETIME keyword. If set, FCIS_ADDR61 is meaningful.

X'04' Set if the request contains an argument for the CHANGETIME keyword. If set, FCIS_ADDR62 is meaningful.

X'02' Set if the request contains an argument for the INSTALLTIME keyword. If set, FCIS_ADDR63 is meaningful.

X'01' Not used by file control.

FCIS_ADDR1
is the address of an 8-byte area containing the name from FILE.

FCIS_ADDR2
is the address of a 44-byte area containing the name from DSNAME.

FCIS_ADDR3
is the address of a 4-byte area containing the CVDA from FWDRECOVSTATUS.

FCIS_ADDR4
is the address of a 4-byte area containing the data from STRINGS.

FCIS_ADDR5
is the address of a 44-byte area containing the name from BASEDSNAME.

FCIS_ADDR6
is the address of a 4-byte area containing the data from LSRPOOLID.

FCIS_ADDR7
is the address of a 4-byte area containing the CVDA from READ.

FCIS_ADDR8
is the address of a 4-byte area containing the CVDA from UPDATE.

FCIS_ADDR9
is the address of a 4-byte area containing the CVDA from BROWSE.

FCIS_ADDR10
is the address of a 4-byte area containing the CVDA from ADD.

FCIS_ADDR11
is the address of a 4-byte area containing the CVDA from DELETE.

FCIS_ADDR12
is the address of a 4-byte area containing the CVDA from DISPOSITION.

FCIS_ADDR13
is the address of a 4-byte area containing the CVDA from EMPTYSTATUS.

FCIS_ADDR14
is the address of a 4-byte area containing the CVDA from OPENSTATUS.

FCIS_ADDR15
is the address of a 4-byte area containing the CVDA from ENABLESTATUS.

FCIS_ADDR16
is the address of a 4-byte area containing the CVDA from RECOVSTATUS.

FCIS_ADDR17
is the address of a 4-byte area containing the CVDA from ACCESSMETHOD.

FCIS_ADDR18
is the address of a 4-byte area containing the CVDA from TYPE.

FCIS_ADDR19
is the address of a 4-byte area containing the CVDA from OBJECT.

FCIS_ADDR20
is the address of a 4-byte area containing the name from REMOTESYSTEM.

FCIS_ADDR21
is the address of an 8-byte area containing the name from REMOTENAME.

FCIS_ADDR22
is the address of a 4-byte area containing the CVDA from RECORDFORMAT.

FCIS_ADDR23
is the address of a 4-byte area containing the CVDA from BLOCKFORMAT.

FCIS_ADDR24
is the address of a 4-byte area containing the CVDA from KEYLENGTH.

FCIS_ADDR25
is the address of a 4-byte area containing the data from KEYPOSITION.

FCIS_ADDR26
is the address of a 4-byte area containing the data from RECORDSIZE.

FCIS_ADDR27
is the address of a 4-byte area containing the CVDA from RELTYPE.

FCIS_ADDR28
is the address of a 4-byte area containing the CVDA from EXCLUSIVE.

FCIS_ADDR29
is the address of a 4-byte area containing the data from BLOCKKEYLEN.

FCIS_ADDR30
is the address of a 4-byte area containing the data from BLOCKSIZE.

FCIS_ADDR31
is not used by file control.

FCIS_ADDR32
is the address of a 4-byte area containing the data from BUSY.

FCIS_ADDR33
is the address of a 4-byte area containing the CVDA from TABLE.

FCIS_ADDR34
is the address of a 4-byte area containing the data from MAXNUMRECS.

FCIS_ADDR35
is the address of a 4-byte area containing the CVDA from READINTEG.

FCIS_ADDR36
is the address of a 4-byte area containing the CVDA from RLSACCESS.

FCIS_ADDR37
is the address of a 8-byte area containing the data from DEFINESOURCE.

FCIS_ADDR38
is the address of a 4-byte area containing the CVDA from INSTALLAGENT.

FCIS_ADDR39
is the address of a 8-byte area containing the data from INSTALLUSRID.

FCIS_ADDR40
is the address of a 4-byte area containing the CVDA from CHANGEAGENT.

FCIS_ADDR41 to FCIS_ADDR51
are not used by file control.

FCIS_ADDR52
is the address of a 4-byte area containing the data from JOURNALNUM.

FCIS_ADDR53
is the address of a 4-byte area containing the data from LOADTYPE.

FCIS_ADDR54
is the address of a 4-byte area containing the data from CFDTPOOL.

FCIS_ADDR55
is the address of a 4-byte area containing the data from TABLENAME.

FCIS_ADDR56
is the address of a 4-byte area containing the data from UPDATEMODEL.

FCIS_ADDR57
is the address of a 4-byte area containing the data from REMOTETABLE.

FCIS_ADDR58
is the address of a 4-byte area containing the CVDA from RBATYPE.

FCIS_ADDR59
is the address of a 8-byte area containing the data from CHANGEUSRID.

FCIS_ADDR60
is the address of a 4-byte area containing the data from CHANGEAGREL.

FCIS_ADDR61
is the address of a 8-byte area containing the data from DEFINETIME.

FCIS_ADDR62
is the address of a 8-byte area containing the data from CHANGETIME.

FCIS_ADDR63
is the address of a 8-byte area containing the data from INSTALLTIME.

Modifying fields in the command-level parameter structure

Some fields that are passed to a file control SPI request are used as input to the request, and some are used as output to the request. The method that your user exit program uses to modify a field depends on the usage of the field.

As a general rule:

- On INQUIRE FILE requests, all fields except FILE are output fields.
- On SET FILE requests, all fields are input fields.

For a full description of the parameters to INQUIRE FILE, see Table 5. For a full description of the parameters to SET FILE, see Table 6 on page 95.

Table 5. INQUIRE FILE requests. The relationship between arguments, keywords, data types, and input/output types.

Argument	Keyword	Data Type	Input/Output
Arg1	FILE	CHAR(8)	See note.
Arg2	DSNAME	CHAR(44)	Output
Arg3	FWDRECSTATUS	BIN(31)	Output
Arg4	STRINGS	BIN(31)	Output
Arg5	BASEDSNAME	CHAR(44)	Output
Arg6	LSRPOOLNUM	BIN(31)	Output
Arg7	READ	BIN(31)	Output
Arg8	UPDATE	BIN(31)	Output
Arg9	BROWSE	BIN(31)	Output
Arg10	ADD	BIN(31)	Output
Arg11	DELETE	BIN(31)	Output
Arg12	DISPOSITION	BIN(31)	Output
Arg13	EMPTYSTATUS	BIN(31)	Output
Arg14	OPENSTATUS	BIN(31)	Output
Arg15	ENABLESTATUS	BIN(31)	Output
Arg16	RECOVSTATUS	BIN(31)	Output
Arg17	ACCESSMETHOD	BIN(31)	Output
Arg18	TYPE	BIN(31)	Output
Arg19	OBJECT	BIN(31)	Output
Arg20	REMOTESYSTEM	CHAR(4)	Output
Arg21	REMOTENAME	CHAR(8)	Output
Arg22	RECORDFORMAT	BIN(31)	Output
Arg23	BLOCKFORMAT	BIN(31)	Output
Arg24	KEYLENGTH	BIN(31)	Output

Table 5. INQUIRE FILE requests (continued). The relationship between arguments, keywords, data types, and input/output types.

Argument	Keyword	Data Type	Input/Output
Arg25	KEYPOSITION	BIN(31)	Output
Arg26	RECORDSIZE	BIN(31)	Output
Arg27	RELTYPE	BIN(31)	Output
Arg28	EXCLUSIVE	BIN(31)	Output
Arg29	BLOCKKEYLEN	BIN(31)	Output
Arg30	BLOCKSIZE	BIN(31)	Output
Arg31	*	*	*
Arg32	BUSY	BIN(31)	Output
Arg33	TABLE	BIN(31)	Output
Arg34	MAXNUMRECS	BIN(31)	Output
Arg35	READINTEG	BIN(31)	Output
Arg36	RLSACCESS	BIN(31)	Output
Arg37	DEFINESOURCE	CHAR(8)	Output
Arg38	INSTALLAGENT	BIN(31)	Output
Arg39	INSTALLUSRID	CHAR(8)	Output
Arg40	CHANGEAGENT	BIN(31)	Output
Arg41 to Arg51	*	*	*
Arg52	JOURNALNUM	BIN(15)	Output
Arg53	LOADTYPE	BIN(31)	Output
Arg54	CEDTPOOL	CHAR(8)	Output
Arg55	TABLENAME	CHAR(8)	Output
Arg56	UPDATEMODEL	BIN(31)	Output
Arg57	REMOTETABLE	BIN(31)	Output
Arg58	RBATYPE	BIN(31)	Output
Arg59	CHANGEUSRID	CHAR(8)	Output
Arg60	CHANGEAGREL	BIN(31)	Output
Arg61	DEFINETIME	CHAR(8)	Output
Arg62	CHANGETIME	CHAR(8)	Output
Arg63	INSTALLTIME	CHAR(8)	Output

Note: The file parameter on INQUIRE FILE commands is:

- An input field if the request does not specify START, NEXT, or END
- An output field if the request specifies NEXT
- Omitted if the request specifies START or END.

Table 6. SET FILE requests. The relationship between arguments, keywords, data types, and input/output types.

Argument	Keyword	Data Type	Input/Output
Arg1	FILE	CHAR(8)	Input

Table 6. SET FILE requests (continued). The relationship between arguments, keywords, data types, and input/output types.

Argument	Keyword	Data Type	Input/Output
Arg2	DSNAME	CHAR(44)	Input
Arg3	FWDRECSTATUS	BIN(31)	Input
Arg4	STRINGS	BIN(31)	Input
Arg5	*	*	*
Arg6	LSRPOOLNUM	BIN(31)	Input
Arg7	READ	BIN(31)	Input
Arg8	UPDATE	BIN(31)	Input
Arg9	BROWSE	BIN(31)	Input
Arg10	ADD	BIN(31)	Input
Arg11	DELETE	BIN(31)	Input
Arg12	DISPOSITION	BIN(31)	Input
Arg13	EMPTYSTATUS	BIN(31)	Input
Arg14	OPENSTATUS	BIN(31)	Input
Arg15	ENABLESTATUS	BIN(31)	Input
Arg16	RECOVSTATUS	BIN(31)	Input
Arg17	*	*	*
Arg18	*	*	*
Arg19	*	*	*
Arg20	*	*	*
Arg21	*	*	*
Arg22	*	*	*
Arg23	*	*	*
Arg24	*	*	*
Arg25	*	*	*
Arg26	*	*	*
Arg27	*	*	*
Arg28	EXCLUSIVE	BIN(31)	Input
Arg29	*	*	*
Arg30	*	*	*
Arg31	*	*	*
Arg32	*	*	*
Arg33	TABLE	BIN(31)	Input
Arg34	MAXNUMRECS	BIN(31)	Input
Arg35	READINTEG	BIN(31)	Input
Arg36	RLSACCESS	BIN(31)	Input
Arg37	*	*	*
Arg38	*	*	*
Arg39	*	*	*
Arg40	*	*	*

Table 6. SET FILE requests (continued). The relationship between arguments, keywords, data types, and input/output types.

Argument	Keyword	Data Type	Input/Output
Arg58	*	*	*
Arg59	*	*	*
Arg60	*	*	*
Arg61	*	*	*
Arg62	*	*	*
Arg63	*	*	*

Modifying input fields

The correct method of modifying an input field is to create a new copy of it, and to change the address in the command-level parameter list to point to your new data.

Do not modify an input field by altering the data that is pointed to by the command-level parameter list. To do so would corrupt storage belonging to the application program and would cause a failure when the program attempted to reuse the field.

Modifying output fields

The technique described in "Modifying input fields" is not suitable for modifying output fields. (The results would be returned to the new area instead of the application's area, and would be invisible to the application.)

An output field is modified by altering the data that is pointed to by the command-level parameter list. In the case of an output field, you can modify the application's data in place, because the application is expecting the field to be modified anyway.

Modifying the EID

It is not possible to modify the EID to make major changes to requests. It is not possible, for example, to change an INQUIRE FILE request to a SET FILE request. However, you can make minor changes to requests, such as to turn on the existence bit for a variable that had not been specified on the current request.

The following paragraph lists the bits in the EID that can be modified. Any attempt to modify any other part of the EID is ignored.

Your exit program can modify any bit in FCIS_BITS1, FCIS_BITS2, FCIS_BITS3, FCIS_BITS4, FCIS_BITS5, FCIS_BITS6 and FCIS_BITS7, *except for*:

- The existence bit for the FILE keyword.
- The bits for the START, NEXT, END, DEFINESOURCE, INSTALLAGENT, INSTALLUSRID and CHANGEAGENT keywords.
- Any bits described as "not used by file control".
- Any bit corresponding to a keyword that is not applicable to the command being executed. For example, the bit for the CLOSED keyword can be modified on a SET FILE request but not on an INQUIRE FILE request, because CLOSED has meaning only for a SET FILE request. See the descriptions in Table 5 on page 94 and Table 6 on page 95.

Your program can provide its own command-level parameter structure and EID, in which case you should modify UEPCLPS and TS_ADDR0 respectively to point to the new structures.

The EID is reset to its original value before return to the application program. That is, changes to the EID are retained for the duration of the file control SPI request only.

Note: If you modify the EID, you must be careful not to create inconsistent parameters. For example, if the original request specified SET FILE OPEN and your exit turned on the EID bit for CLOSED, the resulting SET FILE request would specify both OPEN and CLOSED. In this case, the results of the command would be unpredictable.

Modifying user arguments

The way in which a user exit program can modify a user argument depends on whether the argument is an input or an output.

- For input arguments, your exit program should obtain sufficient storage to hold the modified argument, set up the required value, and set the associated pointer in the parameter list to the address of the newly acquired area.
- For output and input/output arguments, your exit program can update the argument in place, because the area of storage is represented in the application by a variable that is expected to receive a value from CICS.

Adding user arguments

Your exit program can add user arguments, provided that it is allowed to modify the corresponding existence bit in the EID.

Assuming that the argument to be added does not already exist, your exit program must:

1. Obtain storage for the argument to be added
2. Initialize the storage to the required value
3. Select and set up the appropriate pointer from the parameter list
4. Select and set up the appropriate existence bit in Arg0
5. If necessary, modify the parameter list to reflect the new end-of-list indicator.

Removing user arguments

Your exit program can remove user arguments, provided that it is allowed to modify the corresponding existence bit in the EID.

Assuming that the argument to be removed exists, your exit program must:

1. Switch the corresponding argument existence bit in Arg0 to zero
2. Modify the parameter list to reflect the new end-of-list indicator.

File control file state program exits XFCSREQ and XFCSREQC

Two user exits are provided in the file control state program that you can call before and after a file request.

XFCSREQ
: This exit is called before a file ENABLE, DISABLE, OPEN, CLOSE, or CANCEL CLOSE request is acted on. You can use XFCSREQ to gather information about the state of the file; for example, which file requests

(SERVREQs) are valid and which journaling options are set. Based on this information, you can suppress the request, if appropriate. See return code UERCBYP for details.

XFCSREQC
This exit is called after the file request has been acted on. You can use XFCSREQC to gather information about the data set associated with the file; for example, which recovery options are set. XFCSREQC is invoked even if you have used XFCSREQ to suppress the file request.

For ENABLE, DISABLE, OPEN, and CANCEL CLOSE requests, each exit is invoked only once. However, for CLOSE requests, because a file can be quiesced before actual closure, the exits might be invoked more than once. There are two occasions when the user exits XFCSREQ and XFCSREQC are not invoked during a close request:

1. On a controlled, non-immediate shutdown of CICS, when CICS closes all files.
2. After loading a user maintained data table. When the data table load has completed the source data set is no longer required. CICS subsequently closes and de-allocates the file, leaving the data table open.

A single CLOSE request

For a single CLOSE request, XFCSREQ and XFCSREQC are invoked more than once if closure is attempted while the file is being accessed by other tasks. For example, the result of a CLOSE NOWAIT command in these circumstances is that XFCSREQ is invoked before the closure is attempted. Because there are still users of the file, the closure is delayed. However, because it specified NOWAIT, the CLOSE request completes, and invokes XFCSREQC with UEPFSRSP set to 'UEFSPEND', meaning closure is pending. When all activity against the file is complete, the file is closed, and XFCSREQ and XFCSREQC are invoked under the task that closed it.

A CLOSE WAIT request

For a CLOSE WAIT request, the exits are invoked as follows. XFCSREQ is invoked, the task requests a closure of the file and waits for the closure to happen. When all activity against the file is complete, the file is closed, and XFCSREQ and XFCSREQC are invoked under the task that closed it. Finally, because the closure has now been completed, the task that issued the CLOSE WAIT is resumed, completes its CLOSE request, and invokes XFCSREQC.

A CANCEL CLOSE request

A CANCEL CLOSE request is issued by CICS in response to an UNQUIESCE command that cancels a pending QUIESCE command. A QUIESCE data set command immediately sets all files opened against the specified data set as unenabled, to prevent new tasks being allowed access to the data set. The close part of the operation, however, waits until the last user task finishes before a file is closed. (This is the same as any close operation against a file.) An UNQUIESCE issued while the close is still waiting causes a CANCEL CLOSE request and the invocation of the XFCSREQ and XFCSREQC exits. Note that a CANCEL CLOSE is issued only for close requests that were initiated by a QUIESCE command, not for any other close requests.

Exit XFCSREQ

This exit is invoked before a file ENABLE, DISABLE, OPEN, CLOSE, or CANCEL CLOSE is attempted.

When invoked

Before a file ENABLE, DISABLE, OPEN, CLOSE, or CANCEL CLOSE is attempted.

Note: For function shipped requests, the exit is invoked on the system where the file is local.

Exit-specific parameters

UEPFSREQ

Address of a 2-byte field that indicates the type of file request. The first byte contains one of the following values:

UEPFSOPN
Open request

UEPFSCLS
Close request

UEPFSENB
Enable request

UEPFSDIS
Disable request

UEPFSCAN
Cancel close file request.

If the first byte indicates an open request (UEPFSOPN), the second byte shows the type of open:

UEPFSNOP
Normal open

UEPFSOFB
Open for backout.

If the first byte indicates a close request (UEPFSCLS), the second byte shows the type of close:

UEPFSNC
Normal close

UEPFSCP
Close pending

UEPFSELM
End of load mode close

UEPFSIMM
Immediate close

UEPFSICP
Immediate close pending

UEPFSQU
RLS quiesce close.

UEPFILE

Address of the 8-byte file name.

UEPFINFO
Address of a storage area containing information about the file. The area can be mapped using the DSECT DFHUEFDS, which contains the following fields:

UEFLNAME
The 8-character file name.

UEDSNAME
The 44-character dsname of the data set associated with the file, if this has been set before the file request was issued.

UEFSERV
One byte indicating the SERVREQ settings for this file. The possible values are:

UEFRDIM
Read valid

UEFUPDIM
Update valid

UEFADDIM
Add valid

UEFDELIM
Delete valid

UEFBRZIM
Browse valid.

UEFDSJL
One byte indicating the automatic journaling options set for this file. The possible values are:

UEFJRO
Journal read-only

UEFJRU
Journal read for update

UEFJWU
Journal write update

UEFJWA
Journal write add

UEFJDSN
Dsname has been journaled

UEFJSYN
Journal read synchronously

UEFJASY
Journal write asynchronously.

UEFDSVJL
One byte indicating a further automatic journaling option which applies to VSAM files only. The value is:

UEFJWAC
Write add complete.

UEFDSJID
One byte containing the number of the journal to be used for automatic journaling, if any.

UEFDSACC
One byte indicating the access method of the file. The possible values are:

UEFVSAM
VSAM file

UEFBDAM
BDAM file

UEFCFDT
Coupling facility data table

UEFBCRV
Set to nulls for this exit.

UEFFRLOG
Set to nulls for this exit.

UEFFRCLG
Set to blanks for this exit.

UEFCDATE
Set to nulls for this exit.

UEFCTIME
Set to nulls for this exit.

UEFBCAS
Set to nulls for this exit.

UEFACBCP
This field is set to nulls in this exit.

Note: Only the first seven fields of UEPFINFO are set for this exit. Of the remaining fields, URFFRCLG is set to blanks, and the others are set to nulls.

UEPRECUR
Address of a halfword recursion counter. The counter is set to 0 when the exit is first invoked, and is incremented for each recursive call.

Return codes

UERCNORM
Continue processing.

UERCBYP
Suppress the file request. You cannot use UERCBYP:

- To suppress a CLOSE request if the second byte of UEPFSREQ indicates it is one of the following types of close:
  - End of load-mode close (UEPFSELM)
  - Immediate close (UEPFSIMM)
  - Immediate close pending (UEPFSICP)

- To suppress an OPEN request if a file is being opened to carry out backout processing, because this would cause a backout failure. The second byte of UEPFSREQ is set to UEPFSOFB if the file is being opened for backout.

In the case of a valid suppression, CICS issues message DFHFC0996:

```
Open/Close/Enable/Disable/Cancel of close of file
filename suppressed due to intervention of user exit
```

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

API and SPI calls
All except EXEC CICS SHUTDOWN and EXEC CICS XCTL can be used.

Note:

1. Take care when issuing recursive commands not to cause a loop. For example, it is your responsibility to avoid entering a loop when a file control request is issued from the XFCSREQ exit. Use of the recursion counter UEPRECUR is recommended.
2. Exit programs that issue EXEC CICS commands must first address the EIB. See Using CICS services.
3. Exit programs that issue EXEC CICS commands, and that use the DFHEIENT macro, should use the DFHEIRET macro to set a return code and return to CICS. See Global user exit XPI examples, showing the use of storage.
4. Exit programs can invoke EXEC CICS SET commands against the file whose state change has led to the exit being invoked. However, dependent upon other concurrent activity within the CICS system, there is the potential for a deadlock to occur between tasks that are manipulating the state of the file by means of such SPI commands.

Exit XFCSREQC

This exit is called after a file ENABLE, DISABLE, OPEN, CLOSE, or CANCEL CLOSE command completes.

When called
After a file ENABLE, DISABLE, OPEN, CLOSE, or CANCEL CLOSE command completes.

Note: For function-shipped requests, the exit is called on the system where the file is local.

Exit-specific parameters

UEPFSREQ
Address of a 2-byte field that indicates the type of file request. The first byte contains one of the following values:

UEPFSOPN
Open request

UEPFSCLS
Close request

UEPFSENB
Enable request

UEPFSDIS
Disable request

UEPFSCAN
Cancel file close request.

If the first byte indicates an open request (UEPFSOPN), the second byte shows the type of open:

UEPFSNOP
Normal open

UEPFSOFB
Open for backout.

If the first byte indicates a close request (UEPFSCLS), the second byte shows the type of close:

UEPFSNC
Normal close

UEPFSCP
Close pending

UEPFSELM
End of load mode close

UEPFSIMM
Immediate close

UEPFSICP
Immediate close pending

UEPFSQU
RLS quiesce close.

UEPFILE
Address of the 8-byte file name.

UEPFINFO
Address of a storage area containing information about the file. The area can be mapped using the DSECT DFHUEFDS, which contains the following fields:

UEFLNAME
The 8-character file name.

UEDSNAME
The 44-character dsname of the data set associated with the file.

UEFSERV
One byte indicating the SERVREQ settings for this file. The possible values are:

UEFRDIM
Read valid

UEFUPDIM
Update valid

UEFADDIM
Add valid

UEFDELIM
Delete valid

UEFBRZIM
Browse valid.

UEFDSJL
One byte indicating the automatic journaling options set for this file. The possible values are:

UEFJRO
Journal read-only

UEFJRU
Journal read for update

UEFJWU
Journal write update

UEFJWA
Journal write add

UEFJDSN
Dsname has been journaled

UEFJSYN
Journal read synchronously

UEFJASY
Journal write asynchronously.

UEFDSVJL
One byte indicating a further automatic journaling option which applies to VSAM files only. The value is:

UEFJWAC
Write add complete.

UEFDSJID
One byte containing the number of the journal to be used for automatic journaling, if any.

UEFDSACC
One byte indicating the access method of the file. The possible values are:

UEFVSAM
VSAM file

UEFBDAM
BDAM file

UEFCFDT
Coupling facility data table

UEFBCRV
One byte indicating the recovery attributes of the data set associated with this file. The possible values are:

UEFBCFR
Forward recovery specified

UEFBCLOG
Logging specified

UEFBCVAL
Flag indicating that recovery attributes are valid.

UEFFRLOG
A 1-byte field containing the forward recovery log identifier in the range 1—99, taken from the recovery attributes in the CICS file resource definition. This number corresponds to a CICS internal journal name of the form DFHJ*nn*, where *nn* is the forward recovery log number. CICS maps this journal name to a forward recovery log stream.

The field is set to zero if forward recovery logging is not specified for the file, or if the forward recovery log stream name has been obtained from the ICF catalog.

UEFFRCLG
A 26-byte field containing the name of the forward recovery log stream taken from the ICF catalog, to be used for forward recovery. Set to blanks if not specified in the ICF catalog or if forward recovery is not being used for the file.

UEFCDATE
A date (YYYYDDD+) in packed decimal format. This field is set only when the file is the last file to close against the VSAM sphere with which it is associated. It contains the date when activity against the VSAM sphere was brought to an end (quiesced).

UEFCTIME
A time (HHMMSST+) in packed decimal format. This field is set only when the file is the last file to close against the VSAM sphere with which it is associated. It contains the time when activity against the VSAM sphere was brought to an end.

UEFBCAS
A flag-byte indicating the availability of this data set. If set, the value is:

UEPFBCAS
Data set marked unavailable.

UEFACBCP
Address of a read-only copy of the ACB for a VSAM file, or the DCB for a BDAM file. Set only after completion of a successful open.

UEPFSRSP
Address of a byte containing the return codes for the request. This has one of the following values:

UEFSNORM
Normal response.

UEFSWARN
Warning response.

UEFSFAIL
Failure response.

UEFSPEND
Pending response. The 'Pending' response can be returned only after a CLOSE request. It indicates that, as a result of the CLOSE request, a closure is pending on the file, the file is being quiesced. When all activity against the file has completed, it is closed. Note that, if enabled, the XFCSREQ and XFCSREQC exits are driven again, when the actual closure takes place.

UEPRECUR
Address of a halfword recursion counter. The counter is set to 0 when the exit is first called, and is incremented for each recursive call.

Note:

1. The first seven fields of UEPFINFO (UEFLNAME through UEFDSACC) are set for all requests; that is, following an OPEN, CLOSE, ENABLE, or DISABLE request.
2. The next three fields (UEFBCRV, UEFFRLOG, and UEFFRCLG) are valid only after a successful OPEN request.
3. The fields UEFCDATE through UEFCBCAS are set only after a successful CLOSE request. After all other requests, if the file is already closed, if the closure fails, or if the closure is pending, these fields are set to nulls.
4. Exit programs can call EXEC CICS SET commands against the file whose state change leads to the exit being called. However, dependent upon other concurrent activity within the CICS system, there is the potential for a deadlock to occur between tasks that are manipulating the state of the file with such SPI commands.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

API and SPI calls
All except EXEC CICS SHUTDOWN and EXEC CICS XCTL can be used.

Note:

1. Take care when issuing recursive commands not to cause a loop. For example, it is your responsibility to avoid entering a loop when a file control request is issued from the XFCSREQC exit. Use of the recursion counter UEPRECUR is recommended.
2. Exit programs that issue EXEC CICS commands must first address the EIB. See Using CICS services.
3. Exit programs that issue EXEC CICS commands, and that use the DFHEIENT macro, should use the DFHEIRET macro to set a return code and return to CICS. See Global user exit XPI examples, showing the use of storage.

Sample global user exit program for XFCSREQC

DFH$REQC provides sample processing for the file control state program global user exit, XFCSREQC. For more information, see File control state sample exit program: DFH$REQC.

File control open/close program exit XFCNREC

You can use XFCNREC to suppress the open failure on a non-RLS data set.

For RLS data sets, recovery is a property of the data set. Therefore it is not possible for files and their base data set to have unmatched recovery attributes. For more information about writing an XFCNREC exit program, see Configuring for recovery of CICS-managed resources.

Exit XFCNREC

When invoked

Before file open, when a mismatch is detected:

1. Between the backout recovery setting for the file and its associated non-RLS data set.
2. Because BWO is required but the recovery attributes indicate that no associated forward recovery file has been specified.

Exit-specific parameters

UEFILE
Address of the 8-byte file name. If the file name is less than 8 characters in length, it will be padded with blanks.

UEDSETN
Address of the 44-byte base data set name. If the data set name is less than 44 characters in length, it will be padded with blanks.

UEPFRCV
Address of a 1-byte field containing the backout recovery setting for the file, as specified in the FILE definition. The possible value is:

UEPFLOG
Backout logging specified.

If RECOV(NONE) is specified in the FILE definition, the addressed field contains hexadecimal zeros.

This field has no meaning if the exit is driven because of a BWO mismatch.

UEPFAIL
Address of a 1-byte field containing the reason for the mismatch. The possible values are:

UEPBWOF
A BWO mismatch

UEPATTF
A mismatch in the backout recovery settings

UEPOPEN
Address of a 1-byte field. Its default value is N. To bypass an open failure caused by a BWO mismatch, set the addressed field to Y.

Return codes

UERCNORM
Fail open as normal.

UERCBYP
Bypass open failure—accept mismatch.

XPI calls
Must not be used.

SPI calls
Must not be used.

API and SPI calls
Must not be used.

XFCNREC exit with a backout recovery setting mismatch

Use the XFCNREC global user exit if you want to continue with open processing, even though the backout recovery settings for different files associated with the same base data set are not consistent.

After an open failure has been suppressed, CICS can no longer guarantee integrity for the data set and marks it accordingly. If you use the **EXEC CICS INQUIRE DSNAME** or **CEMT INQUIRE DSNAME RECOVSTATUS** commands after an open failure has been suppressed, a response of NOTRECOVABLE is returned. Logging continues for the data set for requests. Logging uses only files that have BACKOUT defined.

The mismatched state of the data set continues until an **EXEC CICS** or **CEMT SET DSNAME REMOVE** command is issued, or until an initial or cold start of CICS, if the associated data set is not in a backout failed state.

At the point when the mismatch is accepted, CICS issues a message to warn that integrity can no longer be guaranteed. The order in which files are opened for the same base data set determines the content of the message that is received.

If the base cluster block is set as unrecoverable and a mismatch has occurred, access is granted to the data set with an unrecoverable file, before the data set is fully recovered.

Three parameters are passed to the XFCNREC exit to provide a means of selecting which mismatches to accept and which to reject. These parameters are the address of the file name, the address of the base data set name, and the address of a byte containing the file backout indicator. Because the exit is driven only if there is a mismatch, the data set backout indicator can be derived from the setting for the file.

Note: If XFCNREC is used to suppress an open failure due to a mismatch, the global user exit XFCSREQC passes the base data set backout setting as the exit parameter UEFBCRV, and not the file backout setting, which might be different.

Using XFCNREC with a BWO mismatch

Exit XFCNREC can allow the file to be opened and CICS will continue to run normally. However, forward recovery will not be available for the opened data set.

File control quiesce receive exit, XFCVSDS

The XFCVSDS exit is invoked when VSAM RLS notifies CICS that processing is required as a result of some data set-related events occurring in the sysplex.

XFCVSDS is invoked before CICS processing takes place, and only if a data set name block (DSNB) exists for the data set. The actions that cause XFCVSDS to be invoked are:

- **A data set is being quiesced throughout the sysplex**.

  CICS is informed about this action only if it has files open in RLS mode against the data set.

  If CICS is notified about a quiesce action, the XFCVSDS global user exit program can cancel the data set quiesce, in which case it cancels the quiesce throughout the sysplex, and the data set remains in the unquiesced state.
- **A data set is being unquiesced throughout the sysplex**.

  All CICS regions in the sysplex that are registered with a VSAM RLS control ACB are informed about unquiesce actions.
- **DFSMSdss wants to start a non-BWO backup of a data set**.

  CICS is notified about a non-BWO backup start action only if it has files open in RLS mode against the data set.

  If CICS is notified about a non-BWO backup start action, XFCVSDS can be used to cancel the backup.
- **DFSMS has completed a non-BWO backup of a data set**.

  All CICS regions in the sysplex that are registered with a VSAM RLS control ACB are informed about non-BWO backup complete actions.
- **DFSMS wants to start a BWO backup of a data set**.

  CICS is notified about a BWO backup start action only if it has files open in RLS mode against the data set.

  If CICS is notified about a BWO backup start action, XFCVSDS can be used to cancel the backup.
- **DFSMS has completed a BWO backup of a data set**.

  All CICS regions in the sysplex that are registered with a VSAM RLS control ACB are informed about BWO backup complete actions.

Exit XFCVSDS

Exit XFCVSDS is invoked after VSAM RLS has informed CICS that processing is required as a result of a data set-related action occurring in the sysplex.

When invoked

Invoked after VSAM RLS has informed CICS that processing is required as a result of a data set-related action occurring in the sysplex.

Exit-specific parameters

UEPDSNAM

Address of a 44-byte field containing the name of the data set to which the action applies

UEPVSACT

Address of a 1-byte field indicating the RLS action of which CICS has been informed. Possible values are:

UEQUIES

Quiesce

UEUNQUIS
Unquiesce

UENBWST
Non-BWO backup start

UENBWCMP
Non-BWO backup complete

UEBWOST
BWO backup start

UEBWOCMP
BWO backup complete.

UEPQUCLS
Address of a 1-byte field indicating, for UEQUIES only, how files open in RLS mode are to be closed. Possible values are:

UEORDCLO
Wait until all in-flight UOWs that are accessing the data set have completed syncpoint before closing.

UEIMMCLO
Abend all in-flight UOWs that are accessing the data set before closing.

UEPCPTEC
Address of a 1-byte field indicating, for UENBWST and UEBWOST only, whether the backup is to use the concurrent copy technique. Possible values are:

UEORDCOP
Concurrent copy is not being used.

UECONCOP
Concurrent copy is being used.

Return codes

UERCNORM
Continue processing—complete the actions required to support the VSAM RLS operation.

UERCBYP
This applies only to actions UEQUIES, UENBWST and UEBWOST. CICS is *not* to carry out the processing required for the VSAM RLS action, and is to cancel the action throughout the sysplex.

A return code of UERCPURG is not allowed.

XPI calls
All can be used.

API and SPI calls
You can use CICS API and SPI commands at this exit. In general all can be used, with the following restrictions:

- You should avoid the use of commands that cause the issuing task to suspend.
- You must not use EXEC CICS SHUTDOWN or EXEC CICS XCTL.
- You must not use the QUIESCESTATE option of EXEC CICS SET DSNAME for data set UEPDSNAM.

- You must not use the OPENSTATUS option of EXEC CICS SET FILE, or issue file control requests, for files that reference data set UEPDSNAM.

File control quiesce send exit XFCQUIS

The XFCQUIS global user exit is invoked on completion of a VSAM RLS quiesce or unquiesce of a data set that was requested either by a CEMT or **EXEC CICS SET DSNAME QUIESCESTATE** command.

The exit is invoked regardless of whether the QUIESCESTATE action has completed successfully or unsuccessfully. This enables you to perform, or schedule, any processing that cannot take place until quiesce or unquiesce processing has finished.

When invoked

On completion, successful or failed, of a SET DSNAME QUIESCESTATE command.

Exit-specific parameters

UEPQDSNM

Address of a 44-byte field containing the name of the data set that was being quiesced or unquiesced.

UEPQSTAT

Address of a 1-byte field indicating whether the data set was being quiesced or unquiesced. Possible values are:

UEQSD

Data set was being quiesced by QUIESCESTATE(QUIESCED). In-flight UOWs accessing the data set completed syncpoint before files open in RLS mode were closed.

UEIMQSD

Data set was being quiesced by QUIESCESTATE(IMMQUIESCED). In-flight UOWs accessing the data set were abended before files open in RLS mode were closed.

UEUNQSD

Data set was being unquiesced by QUIESCESTATE(UNQUIESCED).

UEPQRCDE

Address of a 1-byte field indicating the result of the quiesce or unquiesce. Possible values are:

UEQOK

Successful.

UEQREJEC

Rejected—see UEPQCONF for the reason code.

UEQUNKNO

Failed because data set not known to DFSMS as a VSAM data set.

UEQIOERR

Failed because of RLS error or SMSVSAM server not available.

UEQCANCL
Failed because quiesce was canceled by user (UEQSD and UEIMQSD only).

UEQTIMED
Failed because quiesce was canceled due to timeout (UEQSD and UEIMQSD only).

UEQMIGRT
Failed because the data set has been migrated.

UEPQCONF
Address of a 1-byte field indicating the reason why the quiesce or unquiesce was rejected (for UEQREJEC only). Possible values are:

UEQUIINP
Quiesce is in progress (UEQSD and UEIMQSD status only).

UEUNQINP
Unquiesce is in progress.

UENBWINP
Non-BWO copy is in progress.

UEBWOINP
BWO copy is in progress.

UEUNKINP
Unknown event is in progress.

Return codes

UERCNORM
Continue processing.

A return code of UERCPURG is not allowed.

XPI calls
All can be used.

API and SPI calls
You can use CICS API and SPI commands at this exit. In general, all except EXEC CICS SHUTDOWN and EXEC CICS XCTL can be used, but you must not use the QUIESCESTATE keyword of EXEC CICS SET DSNAME.

File control recovery program exits XFCBFAIL, XFCBOUT, XFCBOVER, and XFCLDEL

CICS provides four global user exits that you can use in connection with file control recovery operations.

XFCBFAIL
Invoked when an error occurs during backout.

XFCBOUT
Invoked when CICS is about to back out a file update.

XFCBOVER
Invoked when CICS is about to skip unit-of-work (UOW) backout because a batch program has overridden RLS retained lock protection and opened a data set for batch processing.

XFCLDEL
Invoked when backing out a write to a BDAM or a VSAM ESDS data set.

File control recovery sample exit programs

CICS includes three sample file control global user exit programs: DFH$FCBF, DFH$FCBV, and DFH$FCLD.

Order of invocation

Each of the exits in the file control recovery program might be invoked during an attempt to backout a file update. If the backout fails, each exit might be re-invoked when the backout is retried. If an exit program needs to determine whether it is being invoked during the original backout attempt, or during a retry, it can check the value of the RE_ATTACHED_TRANSACTION field returned by an XPI INQUIRE_TRANSACTION call.

The way in which the exits interact, and the order in which they are invoked, is shown in the following list. Assuming that all the exits are enabled, for each backout attempt or retried backout attempt:

1. If an open during backout fails, XFCBFAIL is invoked. **None of the other exits is invoked**.
2. If the SHCDS PERMITNONRLSUPDATE command has been issued for the data set being backed out, XFCBOVER is invoked. **If it returns UERCNORM (do not perform the backout), no further exits are invoked**.
3. Unless item 1 applies, or XFCBOVER has been invoked and returned UERCNORM, XFCBOUT is invoked.
4. Backout issues a read update request for the record being backed out.

   If the read update fails, XFCBFAIL is invoked, followed by **no further exits**.
5. If the update to be backed out was a write to a data set which does not support physical deletes (that is, a BDAM data set or a VSAM ESDS), XFCLDEL is invoked.
6. If a failure occurs after this point, XFCBFAIL is invoked.

Enabling the exit programs

To enable an exit program you must complete one of two possible actions.

To enable these exits, you must do one of the following:

- Specify the system initialization parameter TBEXITS=(name1,name2,name3,name4,name5,name6), where name1 through name6 are the names of your user exit programs for XRCINIT, XRCINPT, XFCBFAIL, XFCLDEL, XFCBOVER, and XFCBOUT.
- Enable the exits during the first stage of initialization using a PLTPI program.

If you use the TBEXITS parameter to enable the exits, a global work area of 4 bytes is provided. If you use a PLTPI program, you can select the size of the global work area. You can also enable more than one exit program for use at each exit point; the TBEXITS parameter allows only one exit program at each exit point. PLTPI processing is described in Writing initialization and shutdown programs.

Exit XFCBFAIL, file control backout failure exit

Exit XFCBFAIL is invoked whenever there is a failure during backout of an update made to a file record.

If, within a given UOW, there are backout failures for more than one record in the same file, or for records in multiple files, the exit is invoked:

- For the first record in each data set for which backout fails.

  If more than one file is associated with a single data set, only the first record in the first of the files to fail backout within UOW causes CICS to invoke the exit. All subsequent records are failed with the same error, but the exit is not invoked again.
- For the first record for each data set that fails during any retry of the backout for this UOW.

It is not invoked for backout failures to other (non-file-control) resources within the UOW.

For VSAM data sets, backout failure processing saves information that allows the backout to be retried later.

For BDAM data sets, the backout cannot be retried. If backout fails against a BDAM data set, you can use the XFCBFAIL exit to preserve data integrity by terminating CICS immediately. If the XFCBFAIL exit is not enabled, or does not terminate CICS, the BDAM data is committed and the locks are released. If the exit is enabled, you can use the XFCBFAIL global user exit program to save information that you can use to manually correct the data. However, you need to be careful that in doing this you do not back out other changes made between the time of the backout failure and the time of your own manual recovery action.

When invoked

If an error occurs during backout of a change made to a file (on the first failure in the UOW for the data set associated with the file).

Exit-specific parameters

UEPBLOGR

Address of the file control portion of the log record that represents the update that was being backed out when the file control failure occurred. The log record can be mapped using the DSECT DFHFCLGD.

UEPTRANS

Address of a 4-byte field containing the transaction id under which the update that is being backed out was made.

UEPTRMNL

Address of the 4-byte terminal id for the terminal or principal facility from which the update that is being backed out was made.

UEPTASK

Address of the 4-byte (packed decimal) field containing the task number for the task under which the update that is being backed out was made.

UEPFCRSP

Address of the file control response byte. This can have one of the following values:

UEAIXFUL

No space in non-unique alternate index.

UECACHE

RLS cache failure or cache connectivity failure.

UENBWBAK

Non-BWO backup in progress.

UEDLOCK
Deadlock detected.

UEDUPREC
Duplicate key on unique alternate index.

UEIOEROR
I/O error.

UELCKFUL
RLS lock structure full.

UENOLDEL
Logical delete not carried out (XFCLDEL exit point is either not enabled or the XFCLDEL global user exit program chose not to perform the logical delete).

UENOSPAC
Data set out of storage.

UEOPENER
Error opening the file.

UERLSERR
SMSVSAM RLS server failure.

UERLSDIS
RLS access is currently disabled.

UERLSCON
Attempt to continue a thread with a new instance of the SMSVSAM RLS server.

UEUNEXP
Unexpected error.

UEPERR
Address of a one-byte field containing the error type. The values of the error-byte and their meanings are described in "Values of the error-type byte referenced by UEPERR" on page 117.

Return codes

UERCNORM
Continue processing and invoke CICS backout failure control.

This causes a backout failure error message to be issued (DFHFC4701 for a VSAM data set, and DFHFC4702 for a BDAM data set). For a VSAM data set CICS converts the record lock into a retained lock, and the log record is saved for a later retry of the backout.

UERCBYP
Ignore the error (do not invoke CICS backout failure control) and continue. Setting this return code could be damaging to the integrity of your data.

A return code of UERCPURG is not allowed. There is no need to set a UERCPURG return code, because the conditions under which this exit is invoked mean that a purged condition cannot be returned by any XPI or API calls.

XPI calls
All can be used, but subject to the same caution as for API and SPI calls.

API and SPI calls

Although this exit is allowed to issue API and SPI calls, you should be very careful about which commands you use because the exit is invoked during file backout, which is part of syncpoint phase 2.

It is recommended that you restrict EXEC CICS commands to inquiries, and avoid commands that update CICS resources, because the resources may themselves be in a state of recovery. In particular, the following restrictions apply:

1. Do not issue any recoverable operations.
2. Do not use operations that access systems or resource owners external to this CICS, even if the target resource is non-recoverable.
3. Do not disable or close files, because this could cause further error conditions.
4. It is possible for this exit to be invoked under a different transaction environment from that under which the updates that are being backed out were originally made. If your exit program wants to perform any actions (such as writing a message to the terminal) that require it to be running under the original transaction environment, it must first check the value returned in the RE_ATTACHED_TRANSACTION parameter of a transaction manager INQUIRE_TRANSACTION XPI call.

Sample global user exit program for XFCBFAIL

DFH$FCBF provides sample processing for the file control backout failure global user exit XFCBFAIL. For more information, see File control recovery sample exit program: DFH$FCBF.

Values of the error-type byte referenced by UEPERR

The UEPERR field in the XFCBFAIL parameter list points to an error-type byte, which contains one of several possible values.

Possible values are:

XBFERU

An error response has been returned from the file control file-request-handler program while processing a READ UPDATE request. This request is issued by file control backout to retrieve the existing copy of the record before backing it out.

Use UEPFCRSP in combination with the type of record, shown in the file control portion of the log record addressed by parameter UEPBLOGR, to determine the specific problem. The storage area addressed by UEPBLOGR contains either the before-image of a “read-update” record or the new copy of a “write-add” to be deleted. The type-of-record field, FLJB_RECORD_TYPE, is defined in DSECT DFHFCLGD.

XBFERE

An error response has been returned from the file control file-request-handler program while processing a REWRITE request. This request is issued by file control backout to replace the existing copy of the record on the data set with the “before-image” held in the log record addressed by UEPBLOGR. Use parameter UEPFCRSP to determine which error occurred.

XBFEWR

An error response has been returned from the file control file-request-handler program while processing a WRITE request. This

request is issued by file control backout to add the "before-image" of a deleted record. Use parameter UEPFCRSP to determine which error occurred.

XBFEDL
An error response has been returned from the file control file-request-handler program while processing a REWRITE DELETE request. This request is issued by file control backout to delete a new record added to a VSAM data set. Use parameter UEPFCRSP to determine which error occurred.

XBFENO
The failure that occurred during file control backout was not as a result of an error response from the file control file-request-handler program. Use parameter UEPFCRSP to determine which error occurred.

Exit XFCBOUT, file control backout exit

XFCBOUT is invoked when a file control update is about to be backed out. The log record containing the before-image of the record being backed out is passed to the exit program.

XFCBOUT does not provide a return code to allow your exit program to bypass the backout of the update, because this would result in data corruption. However, the file name is in the log record, so your exit program can use an **EXEC CICS INQUIRE FILE** command to get information about the file.

When invoked
Invoked when an update (represented by a before-image log record) is being backed out by File Control.

Exit-specific parameters

UEPFLOGR
The address of the file control portion of the log record that is being presented for backout. This is mapped by the DSECT DFHFCLGD.

Return codes

UERCNORM
Continue processing.

A return code of UERCPURG is not allowed. There is no need to set a UERCPURG return code, because this exit is invoked during syncpoint phase 2, and therefore cannot get a purged response from any calls it makes.

XPI calls
All can be used, but subject to the same caution as for API and SPI calls.

API and SPI calls
Although this exit is allowed to issue API and SPI calls, you should be very careful about which commands you use because the exit is invoked during file backout, which is part of syncpoint phase 2.

It is recommended that you restrict EXEC CICS commands to inquiries, and avoid commands that update CICS resources, because the resources may themselves be in a state of recovery. In particular, the following restrictions apply:

1. Do not issue any recoverable operations.

2. Do not use operations that access systems or resource owners external to this CICS, even if the target resource is unrecoverable.
3. Do not disable or close files, because this could cause further error conditions.
4. It is possible for this exit to be invoked under a different transaction environment from that under which the updates that are being backed out were originally made. If your exit program wants to perform any actions (such as writing a message to the terminal) that require it to be running under the original transaction environment, it must first check the value returned in the RE_ATTACHED_TRANSACTION parameter of a transaction manager INQUIRE_TRANSACTION XPI call.

Because it is anticipated that XFCBOUT will be used for specific applications, no general-purpose sample exit program is provided. You could use any of the samples for the other file control recovery exits, DFH$FCBF, DFH$FCBV, or DFH$FCLD, as the basis for an XFCBOUT exit program.

Exit XFCBOVER, file control backout override exit

Exit XFCBOVER is part of the support CICS file control provides for *batch windows* in a VSAM RLS environment.

VSAM RLS locks individual records within a data set, and these locks are converted to retained locks for those UOWs that are not completed because of backout or indoubt failures, thus preserving data integrity. To avoid corruption of a data set by a non-RLS batch job, which is not aware of the retained record locks, a data set cannot normally be opened for update in non-RLS mode if it has any locked records.

Sample global user exit program for XFCBOVER

DFH$FCBV provides sample processing for the file control backout override global user exit XFCBOVER. For more information, see File control recovery sample exit program: DFH$FCBV.

Retained lock override for batch

There may be circumstances in which you want to override these locks and force the open of a data set for batch processing.

For example, when:

- There is insufficient time available, before running the batch job, in which to resolve the situation that caused the records to be locked, or
- It is known that the batch job cannot harm data integrity (because it does not update existing records in the data set, or it does not update any records that CICS may have updated).

To override the open restriction, VSAM RLS provides the SHCDS PERMITNONRLSUPDATE command, to allow a non-RLS batch job to open a sphere for update even when there are retained locks.

Effect of retained lock override on CICS

VSAM records the use of the option to override retained locks, so that it can notify a CICS region when the region next opens the data set. Because data could have been altered by the non-RLS batch job, the results of CICS performing any recovery (on UOWs that were in a backout-failed or indoubt-failed state at the time of the batch job) are unpredictable. In this situation, therefore, the default CICS

action is not to back out any updates that were outstanding at the time that locks were overridden, and to write diagnostic information about each backout ignored to the CSFL transient data queue.

The XFCBOVER global user exit is provided to enable you, for each UOW log record for which backout is being ignored, to:

- Write application-related diagnostics to supplement those provided by CICS
- To perform application-related recovery actions
- To reverse the default by requesting that the backout should be carried out after all. This option is required for the case where the batch job is known not to corrupt data integrity (for example, because it only inserts records).

When invoked

Whenever CICS is about to ignore a UOW log record that is due to be backed out, because the lock that protected the updated record could have been overridden by a non-RLS batch program.

Exit-specific parameters

UEPOLOGR

Address of the file control portion of a shunted log record that represents an update to a data set for which retained locks may have been overridden. The file control portion of the log record can be mapped using the DSECT DFHFCLGD.

UEPODSN

Address of a 44-byte area of storage containing the name of the data set whose locks were overridden.

Return codes

UERCNORM

Do not perform the backout of this log record. Any updates performed by the batch run should take precedence.

UERCBCKO

Perform the backout. It is known that the actions of the batch job could not have affected this update.

A return code of UERCPURG is not allowed. There is no need to set a UERCPURG return code, because this global user exit is invoked during syncpoint phase 2, and therefore cannot get a purged response from any calls that it makes.

XPI calls

All can be used, but subject to the same caution as for API and SPI calls.

API and SPI calls

Although this exit is allowed to issue API and SPI calls, you should be very careful about which commands you use because the exit is invoked during file backout, which is part of syncpoint phase 2.

It is recommended that you restrict EXEC CICS commands to inquiries, and avoid commands that update CICS resources, because the resources may themselves be in a state of recovery. In particular, the following restrictions apply:

1. Do not issue any recoverable operations.
2. Do not use operations that access systems or resource owners external to this CICS, even if the target resource is non-recoverable.

3. Do not disable or close files, because this could cause further error conditions.
4. It is possible for this exit to be invoked under a different transaction environment from that under which the updates that are being backed out were originally made. If your exit program wants to perform any actions (such as writing a message to the terminal) that require it to be running under the original transaction environment, it must first check the value returned in the RE_ATTACHED_TRANSACTION parameter of a transaction manager INQUIRE_TRANSACTION XPI call.

Exit XFCLDEL, file control logical delete exit

Exit XFCLDEL is invoked whenever a WRITE to a VSAM ESDS, or to a BDAM data set, is being backed out. Because these types of data set do not support deletion, you can use XFCLDEL to perform a logical delete by amending the record in some way that flags it as deleted.

Exit-specific parameters

UEPBLOGR
Address of the file control portion of the log record representing the update that is to be backed out by logical deletion. The log record can be mapped using the DSECT DFHFCLGD.

UEPTRANS
Address of the 4-byte transaction id under which the update that is being backed out was made.

UEPTRMNL
Address of the 4-byte terminal id for the terminal or principal facility from which the update that is being backed out was made.

UEPTASK
Address of the 4-byte (packed decimal) task number for the task under which the update that is being backed out was made.

UEPFDATA
Address of a variable-length field containing the data in the file control request. The exit program can amend the record data addressed by this field, marking it in some way that applications can recognize as representing a logically deleted record.

UEPFLEN
Address of a fullword containing the length of the data in the file control request.

Return codes

UERCFAIL
Do not perform the logical delete, and treat this as a backout failure. This is the default action taken if the exit is not enabled.

UERCLDEL
Perform the logical delete by reapplying the updated record.

A return code of UERCPURG is not allowed. There is no need to set a UERCPURG return code, because the conditions under which this exit is invoked should mean that “purged” cannot be returned by any XPI or API calls.

XPI calls
All can be used, but subject to the same caution as for API and SPI calls.

API and SPI calls

Although this exit is allowed to issue API and SPI calls, you should be very careful about which commands you use because the exit is invoked during file backout, which is part of syncpoint phase 2.

It is recommended that you restrict EXEC CICS commands to inquiries, and avoid commands that update CICS resources, because the resources may themselves be in a state of recovery. In particular, the following restrictions apply:

1. Do not issue any recoverable operations.
2. Do not use operations that access systems or resource owners external to this CICS, even if the target resource is non-recoverable.
3. Do not disable or close files, because this could cause further error conditions.
4. It is possible for this exit to be invoked under a different transaction environment from that under which the updates that are being backed out were originally made. If your exit program wants to perform any actions (such as writing a message to the terminal) that require it to be running under the original transaction environment, it must first check the value returned in the RE_ATTACHED_TRANSACTION parameter of a transaction manager INQUIRE_TRANSACTION XPI call.

The CICS file definition does not have to specify UPDATE=YES in order for CICS to update the record with the logical delete flag set by an XFCLDEL user exit like DFH$FCLD. When a backout is being done, checking the SERVREQS like UPDATE is bypassed.

Sample global user exit program for XFCLDEL

DFH$FCLD provides sample processing for the file control logical delete global user exit XFCLDEL. For more information, see File control recovery sample exit program: DFH$FCLD.

File control RLS coexistence program exit XFCRLSCO

The XFCRLSCO exit can be called during a request to open a file. Use this exit to allow an application to switch the mode between RLS and read-only non-RLS to access a particular data set.

CICS does not allow an open request to take place for a non-RLS file if an RLS file is already open against the same data set, or if the data set has outstanding RLS work in the system. Also, CICS does not allow an open request to take place for an RLS file if an existing non-RLS file is already open against the base data set. In these situations, if an open request occurs and the non-RLS file is open for read-only access, the XFCRLSCO exit is driven. You can use this exit to decide whether to allow the open request to proceed or to fail in the usual manner with message DFHFC0511 or DFHFC0512.

To switch the access mode, the application can open the data set using a new file with a different access mode. Do not keep the same data set open using both access methods simultaneously over an extended period of time, because CICS does not receive a consistent view of the data set when accessing it concurrently using both RLS and non-RLS files. In particular, CICS cannot get a consistent view if the data set is being updated by the RLS file at the same time as it is being read by the non-RLS file.

If VSAM upgrade set processing occurs while the data set is open using both RLS and non-RLS files, there is an increased risk that read errors might occur because the upgrade processing has not completed on either the base cluster or the associated alternate indexes.

Note:

1. The exit is not driven if the non-RLS file has any updatable SERVREQS set (that is, it allows updates, adds, or deletes).
2. The data set being opened must specify share options SHAREOPTION(2) on the VSAM base cluster. If lower share options are specified, VSAM fails the second open.
3. If static allocation is being used then be sure to specify DISP=SHR, otherwise VSAM fails the open.

When invoked

During a file OPEN request, before the open request is issued to VSAM.

Exit-specific parameters

UEPFILEN

Address of an 8-byte field containing the file name. If the file name is less than 8 characters in length, it is padded with blanks.

UEPDSNAME

Address of a 44-byte field containing the base data set name. If the data set name is less than 44 characters in length, it is padded with blanks.

UEPFSERV

Address of a 1-byte field containing the file SERVREQ indicator. Possible values are:

UEPFRDIM

Read valid indicator.

UEPFUPDIM

Update valid indicator.

UEPFADDIM

Add valid indicator.

UEPFDELIM

Delete valid indicator.

UEPFBRZIM

Browse valid indicator.

UEPFDSACC

Address of a 1-byte field containing the file access method flag. Possible values are:

UEPFVSAM

VSAM file indicator.

UEPFDTBL

Data table file indicator.

UEPFDTUM

User data table file indicator.

UEPFRLS

RLS file indicator.

UEPFCFDT
CFDT file indicator.

UEPRECUR
Address of the halfword recursion level.

Return codes

UERCNORM
Continue processing as normal. The open request fails.

UERCBYP
Allow the open to take place, and bypass the coexistence failure.

XPI calls
You can use XPI commands, but the commands must not result in any state changes to any files.

API and SPI calls
You can use EXEC CICS API and SPI commands, but the commands must not make any state changes to any files. For example, you can use **EXEC CICS INQUIRE FILE**, but not **EXEC CICS SET FILE**, or EXEC CICS API commands against file control which result in state changes to any files.

Good morning message program exit (XGMTEXT)

This exit is invoked before a "good morning" message is sent.

When invoked
Before the good morning message is transmitted.

Exit-specific parameters

UEPTCTTE
Address of the terminal control table terminal entry (TCTTE). The TCTTE can be mapped using the DSECT DFHTCTTE.

UEPTIOA
Address of the terminal input/output area (TIOA). The TIOA can be mapped using the DSECT DFHTIOA. However, fields TIOASAL and TIOASCA are **not** programming interfaces.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

HTTP client open and send exits: XWBAUTH, XWBOPEN and XWBSNDO

Exits XWBAUTH, XWBOPEN and XWBSNDO are invoked during processing of **EXEC CICS WEB CONVERSE**, **EXEC CICS WEB OPEN**, **EXEC CICS INVOKE SERVICE**, and **EXEC CICS WEB SEND** commands. They are used in making HTTP client requests from CICS as an HTTP client, which is a facility provided by CICS Web support.

HTTP client send exit XWBAUTH

With XWBAUTH, you can specify basic authentication credentials (user name and password) for a target server or service provider. XWBAUTH passes them to CICS on request, to create an Authorization header, which is forwarded using HTTP.

When you specify AUTHENTICATE(BASICAUTH) in the **EXEC CICS** WEB SEND (Client) or WEB CONVERSE command, the application can provide a user name and password. If they are not supplied, XWBAUTH is called, providing an alternative way of specifying these credentials. XWBAUTH is called when you specify AUTHENTICATE(BASIC) in a URIMAP resource definition for USAGE(CLIENT), unless the application provides a user name and password in the **EXEC CICS** WEB SEND (Client) or WEB CONVERSE command. XWBAUTH is also called with EC INVOKE SERVICE() URIMAP() when urimap specifies AUTHENTICATE(BASIC).

The user name and password are typically specific to the remote server environment, and might be longer than the standard eight characters used by RACF® systems. The user name and password fields can be up to 256 characters in length. The syntax of these fields is not validated.

The host is passed to the user exit program as the UEPHOST parameter, and the path is passed as the UEPPATH parameter. The realm is passed optionally as the UEPREALM parameter. In response, the user exit program returns the user name and password as the UEPUSNM and UEPPSWD parameters. When encoding the supplied userid and password CICS uses the EBCDIC code pages.

The following sample exit programs are shipped in the CICS sample library, SDFHSAMP:

- DFH$WBPI
- DFH$WBEX
- DFH$WBX1
- DFH$WBX2
- DFH$WBGA, a copybook to map the global work area used by the DFH$WBPI, DFH$WBX1, DFH$WBX2, and DFH$WBEX samples.

For more information about the client sample exit programs, see HTTP client sample exit programs (DFH$WB*). For more information about setting up your LDAP profile, see Configuring LDAP for CICS use

Exit XWBAUTH

When invoked

When the **EXEC CICS** WEB SEND or WEB CONVERSE command specifies AUTHENTICATE(BASICAUTH), but the USERNAME and PASSWORD are not specified.

Exit-specific parameters

UEPHOST (Input supplied by CICS)

The address of a field containing the address of the host name, IPv4, or IPv6 address specified in the HOST option of the WEB OPEN command for the connection. The host name is converted into lowercase characters when it is saved in this field. Your user exit program must take this conversion into account when matching the host name.

UEPHOSTL (Input supplied by CICS)
The address of a field containing the halfword length of the host name.

UEPPATH (Input supplied by CICS)
The address of a field containing the address of the path specified in the PATH option of the WEB SEND or WEB CONVERSE command. The path is mixed case, as it was specified.

UEPPATHL (Input supplied by CICS)
The address of a field containing the halfword length of the path.

UEPREALM (Input supplied by CICS)
The address of a field containing the address of the realm name associated with the target destination, if a realm name was returned in a previous HTTP 401 response from the server.

UEPREALML (Input supplied by CICS)
The address of a field containing the halfword length of the realm name.

UEPAUTHT (Input supplied by CICS)
The address of a 1-byte code that indicates the authentication type. This code is a binary 01, indicating Basic Authentication.

UEPUSNM (Output supplied by user exit)
The address of a fullword field, containing the address of the user name required to access the HTTP server. A predefined address and 64-byte area are created by CICS to store the user name. You can place your user name in this 64-byte area, leaving the address in UEPUSNM unchanged. Alternatively, you can place your user name in your own area and replace the address in UEPUSNM with your user name address. If you create your own user name area, the field can be up to 256 bytes in length.

UEPUSNML (Input supplied by CICS and output supplied by user exit)
The address of a halfword field, which initially contains the length of the buffer address supplied in UEPUSNM. Your user exit program must set the length of this buffer to the user name length, as supplied in UEPUSNM.

UEPPSWD (Output supplied by user exit)
The address of a fullword field, containing the address of the password required to access the HTTP server. A predefined address and 100-byte area are created by CICS to store the password or password phrase. You can place your password in this 100-byte area, leaving the address in UEPPSWD unchanged. Alternatively, you can place your password in your own area and replace the address in UEPPSWD with the address of your password. If you create your own password area, the field can be up to 256 bytes in length.

UEPPSWDL (Input supplied by CICS and output supplied by user exit)
The address of a halfword field, which initially contains the length of the buffer address supplied in UEPPSWD. Your user exit program must set the length of this buffer to the actual password length, as supplied in UEPPSWD.

UEPHOSTT (Input supplied by CICS)
The address of a 1-byte code that indicates the host type contained in the UEPHOST parameter.

Binary 01 indicates host name, binary 02 indicates an IPv4 address, and binary 03 indicates an IPv6 address.

Return codes

UERCNORM
The exit has successfully returned a user name and password.

UERCBYP
The exit cannot identify a user name and password. An Authorization header is not sent.

UERCERR
The exit cannot identify a user name and password. The WEB SEND (Client) or WEB CONVERSE command must be stopped.

XPI calls
All XPI calls can be used.

API and SPI commands
All API and SPI commands can be used, except for **EXEC CICS SHUTDOWN** and **EXEC CICS XCTL**.

Typical use of the LDAP XPI functions by XWBAUTH

The expected use of the DFHDDAPX functions (in association with the XWBAUTH global user exit) include opening and closing an LDAP session, browsing results for credentials, scanning and locating results, closing the browse, returning the correct value and closing the search.

BIND_LDAP
Establishes a session with an LDAP server. Used once on the first call to the global user exit XWBAUTH. The LDAP session token is stored in XWBAUTH's global work area (if one is provided) for use by subsequent calls to LDAP_SEARCH.

UNBIND_LDAP
Releases the connection with the LDAP server. This function is only required during CICS shutdown processing. This function can be used during the XSTERM (system termination) global user exit.

SEARCH_LDAP
Searches for credentials, specifying an LDAP distinguished name, that identifies the URL and realm of the required user information. Distinguished name is specified in the following format:

```
racfcid=uuuuuuuu, ibm-httprealm=rrrrrrrr, labeledURI=xxxxxxxx, cn=BasicAuth
```

where:

- `uuuuuuuu` is the current userid, obtained from the XWBAUTH parameter, UEPUSER.
- `rrrrrrrr` is the HTTP 401 realm, obtained from the XWBAUTH parameter, UEPREALM (if this exists).
- `xxxxxxxx` is the target URL, obtained by concatenating `http://` with the hostname from the XWBAUTH parameter, UEPHOST, and the path from the XWBAUTH parameter, UEPPATH.
- `cn=BasicAuth` is an arbitrary suffix that is configured into the LDAP server for storing Basic Authentication credentials.

START_BROWSE_RESULTS
Starts scanning the results returned by SEARCH_LDAP.

GET_NEXT_ENTRY
Locates the next result entry in a series of entries returned by SEARCH_LDAP. Typically, the URL specified in SEARCH_LDAP will locate a unique entry and the GET_NEXT_ENTRY function is not used.

GET_NEXT_ATTRIBUTE
Locates the next attribute in the current result entry. Typically, specific attributes will be selected and the GET_NEXT_ATTRIBUTE function is not used.

END_BROWSE_RESULTS
Ends the browse session started by SEARCH_LDAP.

GET_ATTRIBUTE_VALUE
Returns the values for various attributes of the target distinguished name. For XWBAUTH, these attributes values are the username and password, stored in the attributes `uid` and `userpassword`. XWBAUTH returns these attribute values as credentials.

FREE_SEARCH_RESULTS
Closes the search initiated by SEARCH_LDAP and releases associated storage.

HTTP client open exit XWBOPEN

With XWBOPEN, you can specify proxy servers that are used for HTTP requests by CICS as an HTTP client. You can also apply a security policy to the host name specified for those requests.

XWBOPEN is called during processing of an **EXEC CICS WEB OPEN** command, which is used by an application program to open a connection with a server. XWBOPEN is also called during processing of an **EXEC CICS INVOKE SERVICE** command.

CICS does not have any requirements concerning the use (or otherwise) of proxy servers for HTTP requests by CICS as an HTTP client, and CICS does not apply any security policy for those requests. You have to set up these facilities if they are required by your system or organization.

The **EXEC CICS** WEB OPEN command instructs the CICS web domain to open a connection with a server. XWBOPEN is called before the connection is opened. The host name for the connection (for example, `www.example.com`), which is specified by the HOST option on the **EXEC CICS** WEB OPEN command, is passed as the UEPHOST parameter to the user exit program for checking. At this point, you can use the user exit program for two purposes:

- To determine whether the HTTP request needs to use a proxy server, and to return the name of any proxy server that is required. If a proxy server is needed, return code UERCPROX is used, and the name of the proxy server is returned to the CICS web domain, in the buffer identified by UEPPROXY, and used to make the connection to the server. If no proxy server is needed, return code UERCNORM is used.
- To apply a security policy to the host name. Return code UERCBARR indicates that access to the host is not permitted and a NOTAUTH response is returned to the WEB OPEN command. The application programmer must stop trying to open that connection. If you want to apply a security policy for individual resources, as well as (or instead of) for the host, use the XWBSNDO user exit on the **EXEC CICS** WEB SEND and **EXEC CICS** WEB CONVERSE commands to apply a security policy to the path component of the URL.

The XWBOPEN user exit does not support the use of **EXEC CICS** commands.

The sample programs DFH$WBPI and DFH$WBEX, with the associated copybook DFH$WBGA, show you how to set up proxy server information or a security policy in a global work area. For example, if all the requests from your CICS system must use a single proxy server, you can specify the proxy server name as an initialization parameter. If you use a number of proxy servers or want to apply a security policy to different host names, you can load or build a table that matches host names to appropriate proxy servers or marks them as barred, which can then be used as a lookup table during processing of the **EXEC CICS** WEB OPEN command. The sample programs can be run during program list table post initialization (PLTPI) processing or at any point before you expect the **EXEC CICS** WEB OPEN command to be used.

Exit XWBOPEN

When invoked

During processing of an **EXEC CICS WEB OPEN** or **EXEC CICS INVOKE SERVICE** command.

Exit-specific parameters

UEPHOST (Input supplied by CICS)
The address of a field containing the host name, IPv4, or IPv6 address specified in the HOST option of the WEB OPEN command.

Note: The host name is converted into lowercase when it is saved in this field. Your user exit program must take into account this conversion when matching the host name.

UEPHOSTL (Input supplied by CICS)
The address of a field containing the halfword length of the host name.

UEPPROXY (Output supplied by user exit)
The address of a field containing the address that points to the proxy server name. The proxy server name must be in URL format. On input to the user exit program, the parameter is set to the address of a field containing the address of a 2046-byte area. You can place the proxy server name in this area and leave the address in UEPPROXY unchanged. Alternatively, you can place the proxy server name in your own area and replace the address in UEPPROXY with the address of a field containing the address of your own area.

UEPPROXYL (Output supplied by user exit)
The address of a field containing the halfword length of the proxy server name.

UEPHOSTT (Input supplied by CICS)
The address of a 1-byte code that indicates the host type contained in the UEPHOST parameter.

Note: Binary 01 indicates host name, binary 02 indicates an IPv4 address, and binary 03 indicates an IPv6 address.

Return codes

UERCNORM
A proxy server is not needed for this HTTP request, and the host name is not barred.

UERCPROX
A proxy server is needed for this HTTP request. UEPPROXY has been set to the name of the required proxy server, and UEPPROXYL has been set to the length of the proxy server name.

UERCBARR
The host name of the server is barred.

UERCERR
An error occurred in exit processing.

XPI calls
All XPI calls can be used.

API and SPI commands
No **EXEC CICS** commands can be used.

HTTP client send exit XWBSNDO

With XWBSNDO, you can specify a security policy for HTTP requests by CICS as an HTTP client. XWBSNDO is called during processing of an **EXEC CICS** WEB SEND or **EXEC CICS** WEB CONVERSE command. The host name and path information are passed to the exit, and a security policy can be applied to either or both of these components.

CICS does not apply any security policy for HTTP requests by CICS as an HTTP client; you must set up this facility if it is required by your system or organization.

You can use the XWBOPEN exit on the WEB OPEN command to bar access to a whole host. You use the XWBSNDO exit to do the same or to bar access to specific paths in a host. To bar access to a whole host, using the XWBOPEN exit saves time, because the application program cannot open the connection and so does not waste time creating the request that must be sent. The host name is provided to the XWBSNDO exit so that you can differentiate between identical paths used by different hosts.

If chunked transfer-coding is being used for the HTTP request, XWBSNDO is called only on the first WEB SEND command for the chunked message.

The XWBSNDO user exit does not support the use of **EXEC CICS** commands.

The host is passed to the user exit program as the UEPHOST parameter, and the path is passed as the UEPPATH parameter. Return code UERCNORM indicates that the path is permitted, and return code UERCBARR indicates that the path is not permitted. If the path is not permitted, a NOTAUTH response is returned to the WEB SEND or WEB CONVERSE command, and the application programmer handles this response by closing the connection with a WEB CLOSE command.

Exit XWBSNDO

When invoked
During processing of an **EXEC CICS** WEB SEND or **EXEC CICS** WEB CONVERSE command for an HTTP request by CICS as an HTTP client. A client request is indicated by the use of the SESSTOKEN parameter on the WEB SEND command.

Exit-specific parameters

UEPHOST
The address of a field containing the host name, IPv4, or IPv6 address specified in the HOST option of the WEB OPEN command for the connection.

Note: The host name is converted into lowercase when it is saved in this field. Your user exit program must take this conversion into account when matching the host name.

UEPHOSTL
The address of a field containing the halfword length of the host name.

UEPPATH
The address of a field containing the path specified in the PATH option of the WEB SEND command. The path is in mixed case, as it was specified.

UEPPATHL
The address of a field containing the halfword length of the path.

UEPHOSTT
The address of a 1-byte code that indicates the host type contained in the UEPHOST parameter.

Note: Binary 01 indicates host name, binary 02 indicates an IPv4 address, and binary 03 indicates an IPv6 address.

Return codes

UERCNORM
The path is permitted.

UERCBARR
The path is not permitted.

XPI calls
All XPI calls can be used.

API and SPI commands
No **EXEC CICS** commands can be used.

Intersystem communication program exits, XISCONA, XISLCLQ, and XISQLCL

The three exits in the intersystem communication program allow you to control the length of intersystem queues.

You can use several methods to control the length of intersystem queues. For a description of the available methods, see Intersystem session queue management.

The XISCONA exit

The purpose of XISCONA is to help you prevent the performance problems that can occur when function shipping or DPL requests awaiting free sessions for a non-IPIC connection are queued in the issuing region. The exit permits you to control the number of outstanding ALLOCATE requests by allowing you to reject any function shipping or DPL request that would otherwise be queued.

Important: Use the XZIQUE exit in the z/OS Communications Server working-set module to control the length of intersystem queues, rather than XISCONA.

XZIQUE provides more functions, and is of more general use than XISCONA (it is driven for function shipping, DPL, transaction routing, and distributed transaction processing requests, whereas XISCONA is driven only for function shipping and DPL). If you enable both exits, XZIQUE and XISCONA could both be driven for function shipping and DPL requests, which is not recommended.

If you already have an XISCONA exit program, you may be able to modify it for use at the XZIQUE exit point.

"Contention winner" is the terminology used for LU6.2 connections. The XISCONA exit applies also to MRO and LU6.1 connections: in these, the SEND sessions (defined in the session definitions) are used first for ALLOCATE requests; when all SEND sessions are in use, queuing starts.

Function shipping and DPL requests for a resource-owning region are queued by default if all bound contention winner sessions are busy, so that no sessions are immediately available. If the resource-owning region is unresponsive (for example, if it is a file-owning region, it may be waiting for a system journal to be archived), the queue can become so long that the performance of the issuing region is severely impaired. Further, if the issuing region is an application-owning region, its impaired performance can spread back to the terminal-owning region.

To control the queuing of function shipping and DPL requests, use the XISCONA exit to tell CICS, whenever a session cannot be allocated immediately, whether to queue the request, or to return 'SYSIDERR' to the application. The exit works like this:

1. If the XISCONA exit program is **not** active, CICS queues the request when necessary.
2. If the exit program is active, it is invoked *only if all bound contention winner sessions are in use*. For other failures (for example, 'Mode name not found' or 'Out of service'), CICS bypasses the exit and returns to the application.
3. If it is invoked, your exit program must decide whether or not to queue the request by analyzing the statistics provided through the user exit parameter list. Your exit program could:
   - Stipulate that queuing is never to be used. This is the simplest way to code the exit, and avoids complexities of tuning. It should be effective if you define enough contention winner sessions to handle the peak transaction load for the connection. If you suppress all queuing, you must specify AUTOCONNECT(YES) on the SESSIONS definition, because the queuing mechanism no longer binds sessions for you.

     With this approach, a danger arises if you base your estimate of required sessions on average conditions and the transaction load subsequently varies widely; when CICS cannot use queuing to cope with the variation, users may suffer transaction abends when there is no significant problem in the resource-owning region.
   - Examine the number of requests currently in the queue. The program could, for example, stop queuing when the number exceeds 120% of the maximum number of sessions. You could use this approach to cope with intermittent stoppages in the resource-owning region.

     You could use a table of thresholds for the connections in your system, with values determined from previous experience of queuing problems. Alternatively, you could use the EXEC CICS interface in a separate program to inquire about the state of the connection, and pass the information in a work area to the XISCONA exit program.

- Examine the type of request and the resource being accessed (which can be discovered by examining the request parameter list). The program could, for example, reject file read requests but queue file updates.

Note: Because a failure of the exit program could affect system availability, it is recommended that you make the logic of your program as simple as possible, thus reducing the possibility of errors.

There are some problems that XISCONA cannot solve. For example, if you have specified both a large number of sessions and a large value for MXT, CICS may develop the short-on-storage (SOS) condition *before* XISCONA is invoked because there are no further sessions available.

Sample global user exit program for XISCONA

DFHXIS is sample exit program that shows one way of limiting the queue of ALLOCATE requests, based on the information passed to the program. For more information, see Function-shipping/DPL queue control sample exit program: DFHXIS.

Exit

Exit XISCONA is invoked when a function shipping or DPL request is about to be queued because all bound contention winner sessions to the remote region are in use.

When invoked

When a function shipping or DPL request is about to be queued because all bound contention winner sessions to the remote region are in use.

Note: For DPL requests that are routed dynamically, the dynamic routing program is invoked before XISCONA. If there are no free sessions the routing program may choose not to queue a DPL request; in these circumstances, XISCONA is not invoked. For information about the dynamic routing of DPL requests, see Dynamically routing DPL requests.

Exit-specific parameters

UEPISPCA

Address of a parameter list containing the following fields. You can map the parameter list using the DSECT DFHXISDS.

UEPCONST

Address of the Connection statistics record.

Connection statistics records are of type STICONSR (STID value 52). Your exit program can map the record using the DSECT DFHA14DS. See notes.

UEPMODST

Address of the Mode Entry statistics record, or zero. A Mode Entry statistics record is built *only* if:

- The connection-type is LU6.2 (see field UEPCONTY).
- The profile DFHCICSF (which is always used for function shipping) defines a specific MODENAME to be used in the allocation of LU6.2 sessions.

Mode Entry statistics records are of type STICONMR (STID value 76). Your exit program can map the record (if present) using the DSECT DFHA20DS.

UEPEIPPL
Address of the request parameter list.

UEPCONTY
A 1-byte field indicating the connection-type. Possible values are:

UEPMRO (X'80')
Request for an MRO connection

UEPLU6 (X'40')
Request for an LU6.1 connection

UEPLUC (X'20')
Request for an LU6.2 connection.

UEPNETNM
An 8-character field containing the NETNAME for the connection; that is, the identifier (applid) of the remote CICS region or system.

Note:

1. The general format of statistics records is described in CICS statistics record format.
2. For a list of statistics record-types and their associated copy books, see CICS statistics data section.
3. For a description of the fields in Connection and Mode Entry statistics records, see CICS statistics in DSECTS and DFHSTUP report in Monitoring.

Return codes

UERCAQUE
Queue the request. This is the default.

UERCAPUR
Do not queue the request, unless local queuing is possible.

XPI calls
All can be used.

Important

There is no UERCNORM return code at this exit point, because the exit is invoked after a failure. The choice is whether or not to take the system default action of queuing the request.

The XISLCLQ exit

XISLCLQ is used for EXEC CICS START NOCHECK requests that are scheduled for a non-IPIC connection.

XISLCLQ is invoked, if enabled, under any of the following circumstances:

- The remote system is not in service.
- A connection to the remote system cannot be established.
- No sessions are immediately available, and your XISCONA exit program has specified that the request is not to be queued in the issuing region.

Note that this exit is invoked *only* if the request to be shipped is of type EXEC CICS START NOCHECK. For EXEC CICS requests other than those with the NOCHECK option (which is only available on START commands) the SYSIDERR condition is raised in the application program.

You can use the exit to specify whether the failed request is to be locally queued, to be run when the connection is reestablished.

Local queues are recovered when you perform a system restart.

Exit XISLCLQ

Exit XISLCLQ is invoked after a function shipping request of type EXEC CICS START NOCHECK has failed because the remote system is not in service, a connection to the remote system cannot be established, or no sessions are immediately available, and your XISCONA exit program has specified that the request is not to be queued in the issuing region.

When invoked

After a function shipping request of type EXEC CICS START NOCHECK has failed because the remote system is not in service, a connection to the remote system cannot be established, or no sessions are immediately available, and your XISCONA exit program has specified that the request is not to be queued in the issuing region.

Exit-specific parameters

UEPISPP

Address of a parameter list that contains:

UEPTCTSE

Address of the relevant terminal control table system entry. The TCT system entry can be mapped using the DSECT DFHTCTTE.

UEPXXTE

Address of the local transaction name, or 0 if SYSID was specified in the command.

Note: Your program can use the transaction manager XPI call INQUIRE_TRANDEF to obtain details of the local transaction (see The INQUIRE_TRANDEF call).

UEPPLIST

Address of the parameter list for the command.

Note: No DSECT is provided for this parameter list. You have to code your own DSECT to access the named fields.

Return codes

UERCSYS

Take the system action. This is determined by the value of the LOCALQ attribute in the local TRANSACTION definition for the remote transaction:

LOCALQ(YES)

The request is queued locally.

LOCALQ(NO)

'SYSIDERR' is returned to the application program.

UERCQUE
Queue the request locally (overriding the LOCALQ(NO) attribute, if specified).

UERCIGN
Override the LOCALQ(YES) attribute, if specified, and return with 'SYSIDERR'.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

Important

There is no 'UERCNORM' return code at this exit point, because the exit is invoked after a failure. The choice is whether to take the system default action or to handle the error in some other way.

The XISQLCL exit

You can use the XISQLCL exit for EXEC CICS START NOCHECK commands that are scheduled for an IPIC connection.

It is invoked, if enabled, under any of the following circumstances:

- The IPIC connection is not acquired.
- A session is not available and CICS does not queue the request for a new session.

XISQLCL allows you to decide whether to add the request to a local queue or to return with an error response.

Local queues are recovered when you perform a system restart.

Sample XISQLCL exit program

DFH£XISL is a sample XISQLCL exit program for controlling the queueing of START NOCHECK requests that are scheduled for an IPIC connection. For more information about DFH£XISL, see IPIC queue control sample exit program: DFH£XISL.

Exit XISQLCL

Exit XISQLCL is invoked after a function shipping request of a **START NOCHECK** or **START NOCHECK PROTECT** command over IPIC fails because the remote system is not in service, a connection to the remote system cannot be established, or no sessions are immediately available, and your XISQUE exit program specifies that the request is not queued in the issuing region.

Exit-specific parameters
The DSECT, DFHXILDS, is provided for this parameter list.

UEPISQPL
Address of a parameter list that contains the following fields:

UEPPLIST
The address of the parameter list for the command.

UEPQLEN
A halfword binary field containing the number of items currently on the queue.

UEPIPCNM
The eight-byte name of the IPCONN.

UEPTRID
The four-byte identifier of the local transaction name, or blanks if SYSID is specified in the command. Your program can use the transaction manager XPI call, INQUIRE_TRANDEF, to obtain details of the local transaction. See The INQUIRE_TRANDEF call.

Return codes

UERCSYS
Take the system action. This action is determined by the value of the LOCALQ attribute in the local TRANSACTION definition for the remote transaction:

LOCALQ(YES)
The request is queued locally.

LOCALQ(NO)
A SYSIDERR error message is returned to the application program.

UERCQUE
Queue the request locally, overriding the LOCALQ(NO) attribute, if specified.

UERCIGN
Override the LOCALQ(YES) attribute, if specified, and return with a SYSIDERR response.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

Important

There is no UERCNORM return code at this exit point, because the exit is invoked after a failure. You must choose whether to take the system default action or to handle the error in some other way.

The sample XISQLCL global user exit program, DFH$XISL, is provided.

Interval control program exits XICREQ, XICEXP, and XICTENF

You can use some XPI calls in exit programs invoked from the interval control program. However, when any of these exits are invoked for expiry analysis, any actions that delay the execution of the interval control program can have adverse effects on other transactions that are waiting for intervals to expire.

You can determine whether the exits have been invoked for expiry analysis by examining the type-of-request field, TCAICTR, a copy of which is pointed to by the UEPICRQ1 exit-specific parameter.

The XICREQ exit is invoked by internal requests made by CICS code, as well as by requests made by applications. DFHXRSP issues an interval control WAIT every 2 seconds; this means that any interval control exit programs are also invoked every 2 seconds.

Exit XICREQ

This exit is invoked at the beginning of the interval control program, before request analysis.

When invoked

At the beginning of the interval control program, before request analysis.

Exit-specific parameters

UEPICQID

Address of an 8-byte field containing the request ID parameter on request. See notes 1 and 2.

UEPICTID

Address of a 4-byte field containing the terminal ID, if any, specified on an EXEC CICS START command. See notes 1 and 2.

UEPICTI

Address of 4 bytes containing the transaction ID specified on an EXEC CICS START command. See notes 1 and 2.

UEPICRQ1

Address of a 1-byte field containing a copy of TCAICTR, the first request code field for requests to the interval control program.

UEPICRQ2

Address of a 1-byte field containing a copy of TCAICTR2, the second request code field for requests to the interval control program.

UEPICRT

Address of a 4-byte field containing the expiry time or interval, in packed decimal format. The value is in the form 0HHMMSSF, where H=hours, M=minutes, S=seconds, and F is a positive sign.

Note:

1. The contents of the fields addressed by UEPICQID and UEPICTID are unpredictable if the associated data items were not specified on the request. You must test the copy of TCAICTR to determine whether they contain meaningful values.
2. Your exit program can change the values of the fields addressed by UEPICQID, UEPICTID UEPICTI, and UEPICRT. Changing the values of the fields addressed by UEPICRQ1 or UEPICRQ2 has no effect.

Return codes

UERCNORM

Continue processing.

UERCPURG

Task purged during XPI call.

XPI calls

The following must not be used:

- ADD_SUSPEND

- DELETE_SUSPEND
- DEQUEUE
- ENQUEUE
- RESUME
- SUSPEND
- WAIT_MVS.

Exit XICEXP

This exit is invoked after an interval control time interval has expired.

When invoked

After an interval control time interval has expired.

Exit-specific parameters

UEPICE

Address of the interval control element (ICE) that has just expired. The ICE can be mapped using the DSECT DFHICEDS.

Return codes

UERCNORM

Continue processing.

UERCPURG

Task purged during XPI call.

XPI calls

The following must not be used:

- ADD_SUSPEND
- DELETE_SUSPEND
- DEQUEUE
- ENQUEUE
- RESUME
- SUSPEND
- WAIT_MVS.

Exit XICTENF

This exit is invoked from the interval control program.

When invoked

This exit relates to the 'terminal not known' condition. For more information see "'Terminal not known' condition exits XALTENF and XICTENF" on page 238.

Interval control EXEC interface program exits (XICEREQ, XICERES, and XICEREQC)

These exits are invoked when interactions with interval control programs occur.

XICEREQ

XICEREQ is invoked on entry to the interval control program before CICS processes an interval control request. Using XICEREQ, you can:

- Analyze the request to determine its type, the keywords specified, and their values.

- Modify any value specified by the request before the command is executed.
- Set return codes to specify that either:
  - CICS should continue with the request, modified or unmodified.
  - CICS should bypass the request. (Note that if you set this return code, you must also set up return codes for the EXEC interface block (EIB), as if you had processed the request yourself.)

Note: The XICEREQ exit is invoked by internal requests made by CICS code, as well as by requests made by applications.

XICERES

XICERES is invoked by the interval control program, before CICS processes a non-terminal-related **EXEC CICS START** request that has been dynamically routed to this region.

Note that XICERES is invoked:

- After exit XICEREQ and before XICEREQC (if these exits are enabled). This means that:
  - If an XICEREQ exit program chooses to bypass the request, XICERES is not invoked, even if it is enabled.
  - If an XICEREQ exit program modifies the request, XICERES must deal with the modified request.
- On the *target* region—that is, the region to which the START request has been routed.
- Only if the routing region—the region on which the routing program runs—supports the "resource unavailable" condition (RESUNAVAIL). To support the "resource unavailable" condition, the routing region must be a supported release of CICS TS.
- Only if it is enabled. It is strongly recommended that you enable this exit only in application-owning regions to which non-terminal-related **EXEC CICS START** requests may be dynamically routed.
- By internal requests made by CICS code, as well as by requests made by applications.

The XICERES exit is *not* invoked:

- For statically-routed requests.
- For terminal-related **EXEC CICS START** requests. (These always execute in the terminal-owning region and cannot be routed.)
- For dynamically-routed *transactions* - only dynamically-routed (non-terminal-related) START requests cause the exit to be invoked. Thus, a dynamically-routed transaction that was initiated by a terminal-related **EXEC CICS START** command does not cause the exit to be invoked.
- If it is disabled.
- If an XICEREQ exit program chooses to bypass the request.

You can use XICERES to check that all resources required by the transaction to be started are available on the target region. If, for example, the transaction is disabled, or a required file is missing, your exit program can give the distributed routing program the opportunity to route the request to a different region. To do this, set a return code of UERCRESU. This causes CICS to:

1. Set the DYRERROR field of the distributed routing program's communications area to 'F'—resource unavailable.
2. Reinvoke the routing program, on the routing region, for route selection failure.

3. Return a RESUNAVAIL condition on the **EXEC CICS START** command executed by the mirror on the target region. (This condition is not returned to the application program.)

CICS ignores any changes made by the exit program to the values of any of the exit parameters. Your exit program can set a return code, but not change any parameters.

For guidance information about dynamically routing non-terminal-related **EXEC CICS START** requests, see Non-terminal-related START commands. For information about writing a distributed routing program to route non-terminal-related **EXEC CICS START** requests, see Routing non-terminal-related START requests.

XICEREQC

XICEREQC is invoked after an interval control program request has completed. Using XICEREQC, you can:

- Analyze the request, to determine its type, the keywords specified, and their values.
- Set return codes for the EIB. When examining return codes, refer to the EIBRCODE value.

Note: The XICEREQC exit is invoked by internal requests made by CICS code, as well as by requests made by applications.

Parameters passed to each of the exits

CICS passes ten types of address parameter to the exits.

- Address of the command-level parameter structure (UEPCLPS)
- Address of a token (UEPICTOK) used to pass 4 bytes of data from XICEREQ to XICEREQC
- Addresses of copies of six return code segments, resource, date, and time information from the EIB
- Address of a token (UEPTSTOK) that is valid throughout the life of a task
- Address of an exit recursion count (UEPRECUR).

Exit XICEREQ

Using XICEREQ, you can analyze the request to determine its type, the keywords specified, and their values; modify any value specified by the request before the command is executed; and set return codes to specify whether CICS should continue with or bypass the request.

When the exit is invoked

On entry to the interval control program, before CICS processes an interval control API request.

Exit-specific parameters

UEPCLPS

Address of the command-level parameter structure. See "The UEPCLPS exit-specific parameter" on page 147.

UEPICTOK

Address of a 4-byte token to be passed to XICEREQC. This allows you, for example, to pass a work area to exit XICEREQC.

UEPRCODE
Address of a 6-byte hexadecimal copy of the EIB return code 'EIBRCODE'. For details of EIB return codes, refer to EIB fields.

UEPRESP
Address of a 4-byte binary copy of the EIB response code 'EIBRESP'.

UEPRESP2
Address of a 4-byte binary copy of the EIB response code 'EIBRESP2'.

UEPTSTOK
Address of a 4-byte token that is valid throughout the life of a task. See Using the task token UEPTSTOK.

UEPRECUR
Address of a halfword recursion counter. The counter is set to 0 when the exit is first invoked, and is incremented for each recursive call.

UEPRSRCE
Address of an 8-character copy of the EIB resource value, EIBRSRCE.

UEPDATE
Address of a fullword copy of the EIB date value, EIBDATE.

UEPTIME
Address of a fullword copy of the EIB time value, EIBTIME.

Return codes

UERCNORM
Continue processing.

UERCBYP
The interval control EXEC interface program should ignore this request.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used. You can also use EXEC CICS API commands at this user exit.

Although the exit permits the use of XPI GETMAIN and FREEMAIN calls, you are recommended to use the EXEC CICS GETMAIN and FREEMAIN commands instead.

API and SPI commands
All can be used, except for:

```
EXEC CICS SHUTDOWN
EXEC CICS XCTL
```

Note: Take care when issuing recursive commands not to cause a loop. For example, it is your responsibility to avoid entering a loop when an interval control request is issued from the XICEREQ exit. Use of the recursion counter UEPRECUR is recommended.

Exit XICERES

Exit XICERES is invoked by the interval control program. You can use XICERES to check that all resources required by the transaction to be started are available on the target region.

When the exit is invoked

Before processing of a non-terminal-related **EXEC CICS START** request that has been dynamically routed to this region where the routing region supports the "resource unavailable" condition (RESUNAVAIL).

Exit-specific parameters

Note: CICS ignores any changes made by the exit program to the values of any of the exit parameters. Your exit program can set a return code, but not change any parameters.

UEPCLPS
Address of the command-level parameter structure. See "The UEPCLPS exit-specific parameter" on page 147.

UEPICTOK
Address of a 4-byte token to be passed to XICEREQC.

UEPRCODE
Address of a 6-byte hexadecimal copy of the EIB return code 'EIBRCODE'. For details of EIB return codes, see EIB fields.

UEPRESP
Address of a 4-byte binary copy of the EIB response code 'EIBRESP'.

UEPRESP2
Address of a 4-byte binary copy of the EIB response code 'EIBRESP2'.

UEPTSTOK
Address of a 4-byte token that is valid throughout the life of a task. See Using the task token UEPTSTOK.

UEPRECUR
Address of a halfword recursion counter. Because the XICERES exit can never be called recursively in the same transaction, the value of this field is always 0.

UEPRSRCE
Address of an 8-character copy of the EIB resource value, EIBRSRCE.

UEPDATE
Address of a fullword copy of the EIB date value, EIBDATE.

UEPTIME
Address of a fullword copy of the EIB time value, EIBTIME.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

UERCRESU
A required resource is unavailable. Setting this value causes CICS to reject the routed request, and to return a value of 'F' (resource unavailable) in the DYRERROR field of the routing program's communications area.

XPI calls
All can be used. You can also use EXEC CICS API commands at this user exit.

Although the exit permits the use of XPI GETMAIN and FREEMAIN calls, you are recommended to use the EXEC CICS GETMAIN and FREEMAIN commands instead.

API and SPI commands
All except **EXEC CICS SHUTDOWN** and **EXEC CICS XCTL** can be used.

Exit XICEREQC

Using XICEREQC, you can analyze the request to determine its type, the keywords specified, and their values. You can also set return codes for the EIB.

When examining return codes, refer to the EIBRCODE value. For more information, see "EXEC interface block (EIB)" on page 156.

When the exit is invoked
After an interval control API request has completed, and before return from the interval control EXEC interface program.

Exit-specific parameters

UEPCLPS
Address of the command-level parameter structure. See "The UEPCLPS exit-specific parameter" on page 147.

UEPICTOK
Address of a 4-byte token passed from XICEREQ. This allows XICEREQ to, for example, pass a work area to XICEREQC.

UEPRCODE
Address of a 6-byte hexadecimal copy of the EIB return code 'EIBRCODE'. For details of EIB return codes, see EIB fields.

UEPRESP
Address of a 4-byte binary copy of the EIB response code 'EIBRESP'.

UEPRESP2
Address of a 4-byte binary copy of the EIB response code 'EIBRESP2'.

UEPTSTOK
Address of a 4-byte token that is valid throughout the life of a task. See Using the task token UEPTSTOK.

UEPRECUR
Address of a halfword recursion counter. The counter is set to 0 when the exit is first invoked and increments for each recursive call.

UEPRSRCE
Address of an 8-character copy of the EIB resource value, EIBRSRCE.

UEPDATE
Address of a fullword copy of the EIB date value, EIBDATE.

UEPTIME
Address of a fullword copy of the EIB time value, EIBTIME.

UEP_IC_REMOTE_SYSTEM
If the request is to be sent to a remote region, is the address of an area containing the 4-byte name of the remote region. (The remote region may have been specified by, for example, the SYSID option of the START command, workload management, or the REMOTESYSTEM option of the TRANSACTION definition.)

If the request is to be executed on the local region, this parameter is the address of a 4-byte area containing blanks.

UEP_IC_REMOTE_NAME
If the transaction is to be executed in a remote system, is the address of an area containing the name of the transaction, as it is known in the remote system.

The remote system may be another CICS region, or an IMS system. If UEP_IC_REMOTE_SYSTEM names a CICS region, the name is 1 through 4 characters long. If UEP_IC_REMOTE_SYSTEM names an IMS system, the name is 1 through 8 characters long. IMS uses 8-character names: if UEP_IC_REMOTE_NAME has fewer than 8 characters, IMS translates it into a usable format.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

Although the exit permits the use of XPI GETMAIN and FREEMAIN calls, you are recommended to use the EXEC CICS GETMAIN and FREEMAIN commands instead.

API and SPI commands
All can be used, except for:

```
EXEC CICS SHUTDOWN
EXEC CICS XCTL
```

Note: Take care when issuing recursive commands. For example, you must avoid entering a loop when issuing an interval control request from the XICEREQC exit. Use of the recursion counter UEPRECUR is recommended.

The command-level parameter structure

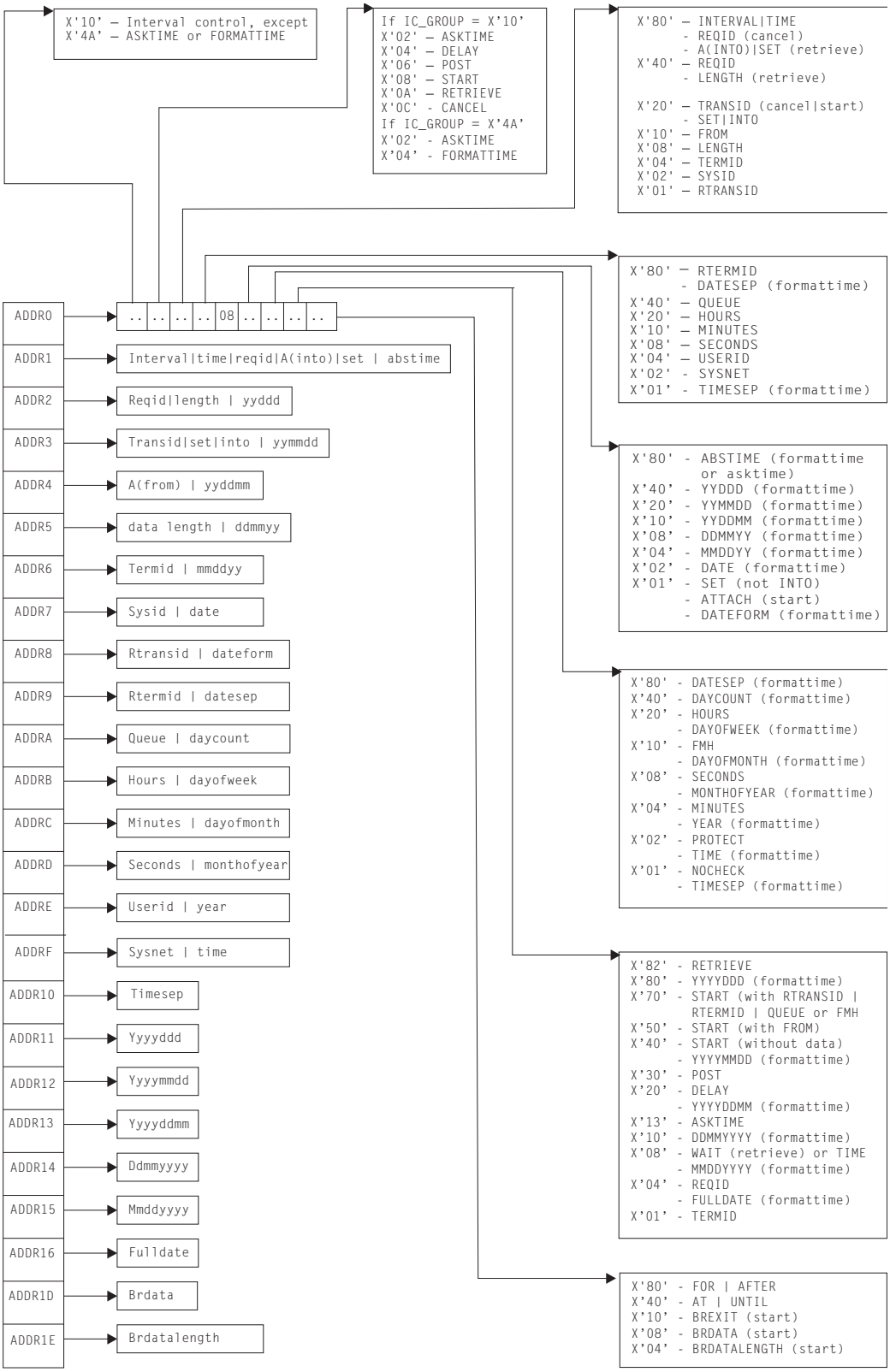


Figure 7. The command-level parameter structure for interval control

The command-level parameter structure consists of a series of addresses. The first address points to the EXEC interface descriptor (EID), which consists of a 9-byte area that describes the type of request and identifies each keyword specified with

the request. The remaining addresses point to pieces of data associated with the request. For example, the second address points to the interval for START requests.

You can examine the EID to determine the type of request and the keywords specified. You can examine the other parameters in the list to determine the values of the keywords. You can also modify values of keywords specified on the request. For example, you could change the SYSID specified in the request.

End of parameter list indicator

The high-order bit is set on in the last address set in the parameter list to indicate that it is the last one in the list. On return from your user exit program, CICS scans the parameter list for the high-order bit to find the last parameter. Therefore, if you modify the length of the parameter list, you must also reset the high-order bit to indicate which is the new last address.

For example, if the parameter list specifies only the first four addresses (IC_ADDR0, the address of the EID, to IC_ADDR3, the address of the name of the transaction named in a START request), the high-order bit is set on in IPC_ADDR3. If you extend the parameter list by setting the address of a SYSID in IC_ADDR7, you must unset the high-order bit in IC_ADDR3 and set it on in IC_ADDR7 instead.

The maximum size of parameter list is supplied to the exit, thus allowing your exit program to add any parameters not already specified without needing to first obtain more storage.

The original parameter list, as it was before XICEREQ was invoked, is restored after the completion of XICEREQC. It follows that the execution diagnostic facility (EDF) displays the original command before **and** after execution: **EDF does not display any changes made by the exit**.

The UEPCLPS exit-specific parameter

The UEPCLPS exit-specific parameter is included in both exit XICEREQ and exit XICEREQC. It is the address of the command-level parameter structure.

The command-level parameter structure contains 26 addresses, IC_ADDR0 through IC_ADDR1F. It is defined in the DSECT IC_ADDR_LIST, which you should copy into your exit program by including the statement COPY DFHICUED.

The command-level parameter list is made up as follows:

IC_ADDR0
is the address of a 9-byte area called the EXEC interface descriptor (EID), which is made up as follows:

- **IC_GROUP**
- **IC_FUNCT**
- **IC_BITS1**
- **IC_BITS2**
- **IC_BITS3**
- **IC_EIDOPT5**
- **IC_EIDOPT6**
- **IC_EIDOPT7**
- **IC_EIDOPT8**

IC_GROUP

X'10' This is an interval control request.

X'4A' This is an ASKTIME or FORMATTIME command.

IC_FUNCT

One byte that defines the type of request.

If IC_GROUP = X'10':

X'02' ASKTIME

X'04' DELAY

X'06' POST

X'08' START

X'0A' RETRIEVE

X'0C' CANCEL

If IC_GROUP = X'4A':

X'02' ASKTIME

X'04' FORMATTIME

IC_BITS1

Existence bits that define which arguments were specified. To obtain the argument associated with a keyword, you need to use the appropriate address from the command-level parameter structure. Before using this address, you must check the associated existence bit. If the existence bit is set off, the argument was not specified in the request and the address should not be used.

X'80' Set if the request contains INTERVAL or TIME arguments, or if a CANCEL request specifies REQID, or if a RETRIEVE request specifies SET or INTO. If set, **IC_ADDR1** is meaningful.

X'40' Set if the request other than CANCEL specifies REQID or if a RETRIEVE request specifies LENGTH. If set, **IC_ADDR2** is meaningful.

X'20' Set if the request specifies TRANSID or if a request other than RETRIEVE specifies SET or INTO. If set, **IC_ADDR3** is meaningful.

X'10' Set if the request specifies FROM. If set, **IC_ADDR4** is meaningful.

X'08' Set if a request other than RETRIEVE specifies LENGTH. If set, **IC_ADDR5** is meaningful.

X'04' Set if the request specifies TERMID. If set, **IC_ADDR6** is meaningful.

X'02' Set if the request specifies SYSID. If set, **IC_ADDR7** is meaningful.

X'01' Set if the request specifies RTRANSID. If set, **IC_ADDR8** is meaningful.

IC_BITS2

Further argument existence bits.

X'80' Set if the request specifies RTERMID, or if a FORMATTIME request specifies DATESEP. If set, **IC_ADDR9** is meaningful.

X'40' Set if the request specifies QUEUE. If set, **IC_ADDRA** is meaningful.

X'20' Set if the request specifies HOURS. If set, **IC_ADDRB** is meaningful.

X'10' Set if the request specifies MINUTES. If set, **IC_ADDRC** is meaningful.

X'08' Set if the request specifies SECONDS. If set, **IC_ADDRD** is meaningful.

X'04' Set if the request specifies USERID. If set, **IC_ADDRE** is meaningful.

X'02' Set if the request specifies SYSNET. If set, **IC_ADDRF** is meaningful.

X'01' Set if a FORMATTIME request specifies TIMESEP. If set, **IC_ADDR10** is meaningful.

IC_BITS3
One byte not used by interval control.

IC_EIDOPT5
Indicates whether certain keywords were specified on the request.

X'80' ABSTIME was specified on a FORMATTIME or ASKTIME command.

X'40' YYDDD was specified on a FORMATTIME command.

X'20' YYMMDD was specified on a FORMATTIME command.

X'10' YYDDMM was specified on a FORMATTIME command.

X'08' DDMMYY was specified on a FORMATTIME command.

X'04' MMDDYY was specified on a FORMATTIME command.

X'02' DATE was specified on a FORMATTIME command.

X'01' On a RETRIEVE command, SET (and not INTO) was specified. On a START command, ATTACH was specified. On a FORMATTIME command, DATEFORM was specified. You cannot modify this field in your user exit.

IC_EIDOPT6
Existence bits that indicate whether certain keywords were specified on the request.

X'80' DATESEP was specified on a FORMATTIME command.

X'40' DAYCOUNT was specified on a FORMATTIME command.

X'20' DAYOFWEEK was specified on a FORMATTIME command, or HOURS was specified.

X'10' DAYOFMONTH was specified on a FORMATTIME command, or FMH was specified.

X'08' MONTHOFYEAR was specified on a FORMATTIME command, or SECONDS was specified.

X'04' YEAR was specified on a FORMATTIME command, or MINUTES was specified.

X'02' TIME was specified on a FORMATTIME command, or PROTECT was specified.

X'01' TIMESEP was specified on a FORMATTIME command, or NOCHECK was specified.

IC_EIDOPT7
Indicates whether certain functions or keywords were specified on the request.

X'F0' CANCEL specified.

X'82' RETRIEVE specified.

X'80' YYYYDD specified on a FORMATTIME command.

X'40' YYYYMMDD specified on a FORMATTIME command, or START specified.

X'30' POST specified.

X'20' YYYYDDMM specified on a FORMATTIME command, or DELAY, RTRANSID, RTERMID, or QUEUE specified, and/or FMH.

X'13' ASKTIME specified.

X'10' DDMMYYYY specified on a FORMATTIME command, or FROM, RTRANSID, or RTERMID specified, and/or QUEUE.

X'08' MMDDYYYY specified on a FORMATTIME command, or TIME or WAIT specified.

X'04' FULLDATE specified on a FORMATTIME command, or REQID specified.

X'01' TERMID specified.

IC_EIDOPT8
Indicates whether certain keywords were specified on the request.

X'80' FOR or AFTER specified.

X'40' AT or UNTIL specified.

X'10' BREXIT specified.

X'08' BRDATA specified.

X'04' BRDATALENGTH specified.

X'02' CHANNEL specified on a START command.

IC_ADDR1
is the address of one of the following:

- An 8-byte area containing the value of the INTERVAL keyword (or TIME keyword if **IC_EIDOPT7** indicates that TIME is specified).
- An 8-byte area containing the value of REQID (if the request is CANCEL).
- An 8-byte area containing the value of the ABSTIME keyword.
- Data returned for INTO (if the request is RETRIEVE, and if **IC_EIDOPT5** indicates that this is not SET).
- A 4-byte address returned for SET (if the request is RETRIEVE and **IC_EIDOPT5** indicates that this is SET).

IC_ADDR2
is the address of one of the following:

- An 8-byte area containing the value of REQID (if the request is DELAY, POST or START).
- A halfword containing the value of LENGTH (if the request is RETRIEVE).

  Warning: For requests that specify INTO, do not change the value of LENGTH to a value greater than that specified by the application. To do so causes a storage overlay in the application.
- An area containing the value of YYDD.

IC_ADDR3
is the address of one of the following:

- An area containing the value of TRANSID (if the request is CANCEL or START).
- A 4-byte address returned for SET (if the request is START or POST and **IC_EIDOPT5** indicates that this is SET).
- An area containing the value of YYMMDD.

IC_ADDR4
is the address of one of the following:

- An area containing the data from FROM.
- An area containing the value of YYDDMM.

IC_ADDR5
is the address of one of the following:

- An area containing the halfword value of LENGTH.

  Warning: For requests that specify INTO, do not change the value of LENGTH to a value greater than that specified by the application. To do so causes a storage overlay in the application.
- An area containing the value of DDMMYY.

IC_ADDR6
is the address of one of the following:

- An area containing the value of TERMID.
- An area containing the value of MMDDYY.

IC_ADDR7
is the address of one of the following:

- An area containing the value of SYSID.
- An area containing the value of DATE.

IC_ADDR8
is the address of one of the following:

- An area containing the value of RTRANSID.
- An area containing the value of DATEFORM.

IC_ADDR9
is the address of one of the following:

- An area containing the value of RTERMID.
- An area containing the value of DATESEP.

IC_ADDRA
is the address of one of the following:

- An area containing the value of QUEUE.

- A fullword containing the value of DAYCOUNT.

IC_ADDRB
is the address of one of the following:
- An area containing the value of HOURS.
- A fullword containing the value of DAYOFWEEK.

IC_ADDRC
is the address of one of the following:
- An area containing the value of MINUTES.
- A fullword containing the value of DAYOFMONTH.

IC_ADDRD
is the address of one of the following:
- An area containing the value of SECONDS.
- A fullword containing the value of MONTHOFYEAR.

IC_ADDRE
is the address of one of the following:
- An area containing the value of USERID.
- A fullword containing the value of YEAR.

IC_ADDRF
is the address of one of the following:
- An 8–byte area containing the value of SYSNET.
- An area containing the value of TIME.

IC_ADDR10
is the address of a 1–byte area containing the value of TIMESEP.

IC_ADDR11
is the address of an area containing the value of YYYYDDD.

IC_ADDR12
is the address of an area containing the value of YYYYMMDD.

IC_ADDR13
is the address of an area containing the value of YYYYDDMM.

IC_ADDR14
is the address of an area containing the value of DDMMYYYY.

IC_ADDR15
is the address of an area containing the value of MMDDYYYY.

IC_ADDR16
is the address of an area containing the value of FULLDATE.

IC_ADDR1D
is the address of an area containing the value of BRDATA.

IC_ADDR1E
is the address of a fullword containing the value of BRDATALENGTH.

IC_ADDR1F
is the address of a 16-byte area containing the value of CHANNEL.

Modifying fields in the command-level parameter structure

Some fields that are passed to interval control are used as input to the request, some are used as output fields, and some are used for both input and output. The method your user exit program uses to modify a field depends on the usage of the field.

The following are always input fields:

- INTERVAL
- TIME
- REQID
- FROM
- TERMID
- SYSID
- HOURS
- MINUTES
- SECONDS
- USERID
- CHANNEL

The following are always output fields:

- DATE
- DATEFORM
- DAYCOUNT
- DAYOFMONTH
- DAYOFWEEK
- DDMMYY
- DDMMYYYY
- FULLDATE
- INTO
- MMDDYY
- MMDDYYYY
- MONTHOFYEAR
- SET
- TIME
- YEAR
- YYDDD
- YYDDMM
- YYMMDD
- YYYYDDD
- YYYYDDMM
- YYYYMMDD

The following are input fields on a START request and output fields on a RETRIEVE request:

- RTRANSID
- RTERMID
- QUEUE

LENGTH is an input field on a START request, an output field on a RETRIEVE with SET specified, and an input/output field on a RETRIEVE with INTO specified.

ABSTIME is an input field on a FORMATTIME request, and an output field on an ASKTIME request. DATESEP and TIMESEP can be input fields on a FORMATTIME request.

Modifying input fields

The correct method of modifying an input field is to create a new copy of it, and to change the address in the command-level parameter list to point to your new data.

Note: You must never modify an input field by altering the data that is pointed to by the command-level parameter list. To do so would corrupt storage belonging to the application program and would cause a failure when the program attempted to reuse the field.

Modifying output fields

You modify an output field by altering the data to which the command-level parameter list points.

The technique described in "Modifying input fields" is not suitable for modifying output fields because the results would be returned to the new area instead of the application's area, and would be invisible to the application.

In the case of an output field, you can modify the application's data in place because the application is expecting the field to be modified anyway.

Modifying the EID

It is not possible to modify the EID to make major changes to requests, such as changing a DELAY request to a START request. However, you can make minor changes to requests, such as turning on the existence bit for SYSID so that the request can be changed into one that is shipped to a remote system.

Some interval control commands use 2 bits in the EID to indicate a single keyword; the EXEC CICS START command, for example, uses 2 bits to indicate TERMID. The first bit, in IC_BITS1, indicates that ADDR6 in the command parameter list is valid (ADDR6 points to TERMID) and the second, in IC_EIDOPT7, is the keyword existence bit to show that the TERMID keyword was specified on the command.

Where this occurs you must ensure that both bit settings are changed (consistently) if you want to modify these commands from within a user exit program, or the results will be unpredictable.

The list that follows shows the bits in the EID that **can** be modified. Any attempt to modify any other part of the EID is ignored.

IC_BITS1

X'80' The existence bit for REQID (if the request is CANCEL)

X'40' The existence bit for LENGTH (if the request is RETRIEVE) or REQID

X'10' The existence bit for FROM

X'08' The existence bit for LENGTH

X'04' The existence bit for TERMID

X'02' The existence bit for SYSID

X'01' The existence bit for RTRANSID.

IC_BITS2

X'80' The existence bit for RTERMID

X'40' The existence bit for QUEUE

X'20' The existence bit for HOURS

X'10' The existence bit for MINUTES

X'08' The existence bit for SECONDS.

IC_EIDOPT6

X'20' The secondary existence bit for HOURS

X'10' The existence bit for FMH

X'08' The secondary existence bit for SECONDS

X'04' The secondary existence bit for MINUTES

X'02' The existence bit for PROTECT

X'01' The existence bit for NOCHECK.

IC_EIDOPT7

Bits in IC_EIDOPT7 should only be modified within the same functional group; that is, only those existence bits defined as valid for a START request should be set on a START request.

ASKTIME requests

X'13' ASKTIME request. This value is fixed for all ASKTIME requests, and should not be modified.

DELAY requests

X'20' DELAY request

X'08' TIME specified

X'04' REQID specified.

POST requests

X'30' POST request

X'08' TIME specified

X'04' REQID specified.

START requests

X'40' START request (without DATA)

X'50' START with DATA request

X'70' START with one or more of RTRANSID, RTERMID, QUEUE, or FMH specified.

X'08' TIME specified

X'04' REQID specified

X'01' TERMID specified.

RETRIEVE requests

X'82' RETRIEVE request.

CANCEL requests

X'F0' CANCEL request

X'04' REQID specified.

IC_EIDOPT8

X'20' Unused by CICS.

The EID is reset to its original value before return to the application program. That is, changes made to the EID are retained for the duration of the interval control request only.

Note: Your user exit program is prevented from making major changes to the EID. However, you must take great care when making the minor modifications that **are** permitted.

Using the interval control request token UEPICTOK

UEPICTOK provides the address of a 4-byte area that you can use to pass information between the XICEREQ and XICEREQC user exits for the same interval control request.

For example, the address of a piece of storage that is obtained by the XICEREQ user exit, which is to be freed by the XICEREQC exit, can be passed in the UEPICTOK field.

UEPICTOK is usable only for the duration of a single interval control request, because its contents might be destroyed at the end of the request. If you need to pass information between successive invocations of a global user exit, you can use task token UEPTSTOK to do so. For more information about UEPTSTOK, see Using the task token UEPTSTOK.

EXEC interface block (EIB)

Copies of EIBRSRCE, EIBRCODE, EIBRESP, and EIBRESP2 are passed to the exit so you can modify/set completion and resource information in XICEREQ and XICEREQC, or examine completion and resource information in XICEREQC.

You can update the copies of EIBRSRCE, EIBRCODE, EIBRESP, and EIBRESP2 that you are given in the parameter list. Interval control copies your values into the real EIB after the completion of XICEREQC; or if you specify a return code of 'bypass' in XICEREQ.

You must set valid interval control responses. You must set all three of EIBRCODE, EIBRESP, and EIBRESP2 to a consistent set of values, such as would be set by CICS interval control to describe a valid completion. **CICS does not police the consistency of EIBRCODE, EIBRESP, and EIBRESP2**. However, if EIBRCODE is set to a non-zero value and EIBRESP is set to zero, CICS overrides EIBRESP with a non-zero value. To aid you in setting the values of EIBRCODE, EIBRESP, and EIBRESP2, the values used by interval control are specified in DFHICUED.

If you want to examine the return codes by using the XICEREQC exit, refer to the EIBRCODE value.

Example of how XICEREQ and XICEREQC can be used

In this example, XICEREQ and XICEREQC are used to route START requests to a number of different CICS regions to provide a simple load balancing mechanism. The example shows only the capabilities of the exits; it is not intended to indicate an ideal way of achieving the function.

In XICEREQ:

1. Scan the global work area (GWA) to locate a suitable CICS region (for example, the region currently processing the least number of START requests).
2. Having decided which system to route the request to, increment the use count for this system.
3. Obtain a 4-byte area in which to store the SYSID for this request. This can be allocated from the GWA to avoid issuing a GETMAIN. If the area is obtained by issuing a GETMAIN, set UEPICTOK to the address of the storage obtained.
4. Set IC_ADDR7 to be the address of the 4-byte area so that XICEREQC can also use this area.
5. If setting IC_ADDR7 now makes it the last address, set the high-order bit in the address, and reset the high-order bit in what was previously the last address.
6. Set the X'02' existence bit on in IC_BITS1 to indicate that a SYSID is specified.
7. Return to CICS.

In XICEREQC:

1. Scan the global work area (GWA) and locate the entry for the CICS region specified in the SYSID parameter.
2. Decrement the use count for this system.
3. If a GETMAIN was issued in XICEREQ to obtain an area to hold the SYSID, issue a FREEMAIN for the address held in UEPICTOK.
4. Return to CICS.

Example and sample programs

CICS supplies two programs for use at the XICEREQ exit:

- DFH$XTSE, supplied as a softcopy listing only (not as a source code file), is an example program that shows how to modify fields in the command-level parameter structure passed to all the EXEC interface exits.
- DFH$ICCN is a sample program for use in a distributed routing environment, where you want to cancel a previously-issued interval control request but have no way of knowing to which region to direct the CANCEL. For examples of situations which DFH$ICCN is designed to cope with, see Canceling interval control requests.

Loader domain exits XLDLOAD and XLDELETE

There are two global user exits in the loader domain. XLDLOAD is invoked when a new instance of a program is loaded into storage, before the program is made available for use.

XLDELETE is invoked after an instance of a program is released by CICS and before the program is freed from storage.

For LPA-resident programs, the exits are still invoked when a program is acquired or released, even though the program is not physically loaded or freed.

These are both information-only exits. Any changes made to the exit parameters by the exit program are ignored by CICS, as is any return code which it sets.

Exit XLDLOAD

This exit is invoked for a program instance brought into storage, before the program becomes available.

When invoked

After an instance of a program is brought into storage, and before the program is made available for use.

Exit-specific parameters

UEPPROGN
Address of an 8-character field containing the name of the program that is being loaded.

UEPPROGL
Address of a 4-byte field containing the length, in bytes, of the program that is being loaded.

UEPLDPT
Address of a 4-byte field containing the address at which the program has been loaded.

UEPENTRY
Address of a 4-byte field containing the address of the program's entry point.

UEPTRANID
Zero, or the address of a 4-byte field containing the transaction ID which applied when the exit was invoked.

UEPUSER
Zero, or the address of an 8-byte field containing the userid in control at the time the exit was invoked.

UEPTERM
Zero, or the address of a 4-byte field containing the terminal name associated with the transaction under which the exit was invoked.

UEPPROG
Zero, or the address of an 8-character field containing the name of the program that was in control at the time the exit was invoked.

UEPLDCTXT
Zero, or the address of a 140-byte field containing the application context when a private program belonging to an application is loaded. The field contains:

1. The platform name, padded with spaces to 64 characters.
2. The application name, padded with spaces to 64 characters.
3. The major version number for the application, which is a fullword binary value.
4. The minor version number for the application, which is a fullword binary value.
5. The micro version number for the application, which is a fullword binary value.

CICS supplies a DSECT named **DFHUEACD** which maps this information. For more information about **DFHUEACD**, see UEACD - User exit application context in Data Areas.

Return codes

UERCNORM
Continue processing.

XPI calls
Must not be used.

API and SPI calls
Must not be used.

Exit XLDELETE

This exit is invoked when a program instance is released by CICS, before the program is freed from storage.

When invoked
After an instance of a program is released by CICS, and before the program is freed from storage.

Exit-specific parameters

UEPPROGN
Address of an 8-character field containing the name of the program that is being freed.

UEPPROGL
Address of a 4-byte field containing the length, in bytes, of the program that is being freed.

UEPLDPT
Address of a 4-byte field containing the address at which the program resides in storage.

UEPENTRY
Address of a 4-byte field containing the address of the program's entry point.

UEPTRANID
Zero, or the address of a 4-byte field containing the transaction ID which applied when the exit was invoked.

UEPUSER
Zero, or the address of an 8-byte field containing the userid in control at the time the exit was invoked.

UEPTERM
Zero, or the address of a 4-byte field containing the terminal name associated with the transaction under which the exit was invoked.

UEPPROG
Zero, or the address of an 8-character field containing the name of the program that was in control at the time the exit was invoked.

UEPLDCTXT
Zero, or the address of a 140-byte field containing the application context when a private program belonging to an application is deleted. The field contains:

1. The platform name, padded with spaces to 64 characters.

2. The application name, padded with spaces to 64 characters.
3. The major version number for the application, which is a fullword binary value.
4. The minor version number for the application, which is a fullword binary value.
5. The micro version number for the application, which is a fullword binary value.

CICS supplies a DSECT named **DFHUEACD** which maps this information. For more information about **DFHUEACD**, see UEACD - User exit application context in Data Areas.

Return codes

UERCNORM
Continue processing.

XPI calls
Must not be used.

API and SPI calls
Must not be used.

Log manager domain exit XLGSTRM

There is one exit point, XLGSTRM, in the log manager domain. You can use XLGSTRM to modify a request to MVS to create a new log stream. You can change the model log stream name and other parameters before they are passed to the MVS system logger.

If a log stream connection request from CICS to the MVS system logger fails because the log stream is not defined to MVS, CICS issues a request to the MVS system logger to create the log stream dynamically, using a model log stream definition.

The model log stream name that CICS passes to MVS depends on whether the journal name refers to the system log or a CICS general log, as follows:

CICS system logs
&sysname.LSN_last_qualifier.MODEL

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

If you do not provide a JOURNALMODEL resource definition for DFHLOG and DFHSHUNT, or if you use the CICS definitions supplied in group DFHLGMOD, the model log stream names default to *&sysname*.DFHLOG.MODEL and *&sysname*.DFHSHUNT.MODEL.

For example, if a CICS region issues a request to create a log stream for its primary system log, and CICS is running in an MVS image with a sysid of MV10 and using the default JOURNALMODEL definition, the MVS system logger expects to find a model log stream named MV10.DFHLOG.MODEL.

If the system name of the MVS image starts with a numeric character and is less than 8 characters long, CICS prefixes it with a "C", so that the model log stream name becomes *C&sysname.LSN_last_qualifier*. This is because the MVS system logger rejects log stream names that begin with a numeric. If the system name of the MVS image starts with a numeric but is

already 8 characters long (the maximum), CICS does not add the "C" prefix, which means that the MVS system logger will reject the default model log stream name. However, your global user exit program can change the model log stream name.

CICS general logs
LSN_qualifier_1.LSN_qualifier2.MODEL. The defaults for these two qualifiers are the CICS region userid and the CICS region APPLID, but they can be user-defined values specified in a JOURNALMODEL resource definition.

For example, if the CICS region userid is CICSHT## and the APPLID is CICSHTA1, the default model name is CICSHT##.CICSHTA1.MODEL.

The following information is passed to an XLGSTRM global user exit program:

- The name of the log stream to be defined
- The default model log stream name
- A system log flag
- The MVS system logger IXGINVNT parameter list.

Your exit program can amend the model stream name by updating the field pointed to by the UEPMLSN exit-specific parameter. Here is an example of how your exit program can change the model stream name:

```
         L     R3,UEPMLSN          R3 = address of stream name
         MVC   0(26,R3),=CL26'NEW.MODEL.NAME'
```

By updating the field pointed to by the UEPIXG parameter, your exit program can amend the IXGINVNT macro parameter list used by the MVS system logger to define the log stream. Use the IXGINVNT MF=M form of the macro, which allows the exit to specify the log stream attributes to be used. Here is an example of how your exit program can change the structure name:

```
         L     R9,UEPIXG
         IXGINVNT REQUEST=DEFINE,
               TYPE=LOGSTREAM,
               STRUCTNAME=NEW_STRUCTURE,
               MF=(M,(R9),NOCHECK)
    ...
NEW_STRUCTURE DC CL16'LOG_SYSTEST_009'
```

You do not need to code the list and execute forms of the IXGINVNT macro, or include the IXGCON or IXGANSAA macros in your exit—these are provided by the CICS code which issues the DEFINE request.

For information about the IXGINVNT service, see the *z/OS MVS Programming: Assembler Services Guide*.

An XLGSTRM global user exit program can set explicit attributes for the log stream definition, and can also set a return code that causes the log stream definition to be bypassed.

Note: If you want XLGSTRM to intercept the connection of the CICS system logs, you must enable your exit program in a first-phase PLT program.

Sample global user exit program for XLGSTRM

DFH$LGLS is a sample global user exit program for the XLGSTRM exit point. It shows you how to access and change some of the parameters that are passed to an

XLGSTRM exit program. For more information, see Log manager domain sample exit program: DFH$LGLS.

Exit XLGSTRM

Exit XLGSTRM is invoked after the CICS log manager detects that a log stream does not exist and before it calls the MVS system logger to define the log stream dynamically.

Exit-specific parameters

UEPTRANID
The address of the 4-byte transaction id.

UEPUSER
The address of the 8-byte userid associated with the transaction if the current task is a user task.

UEPTERM
The address of the 4-byte terminal id associated with the transaction, if any.

UEPPROG
The address of the 8-byte application program name for this transaction, if any.

UEPLSN
Address of a 26-character field containing the name of the log stream to be defined.

Your exit program should not modify the name of the logstream. On return from the exit, CICS ignores any changes to the contents of the field addressed by UEPLSN. JOURNALMODEL definitions are provided to cater for log stream name selection.

UEPMLSN
Address of a 26-character field specifying the name of the model log stream to be used to provide the attributes for the new log stream. This field is modifiable to allow the global user exit program to specify a different model log stream name from the one generated by CICS.

UEPIXG
Address of the IXGINVNT macro parameter list for use by the MVS system logger to define the log stream. Using the MF=M form of the IXGINVNT macro, the global user exit program can specify the log stream attributes to be used.

For details of the IXGINVNT macro, see the *z/OS MVS Programming: Authorized Assembler Services Guide*.

UEPLGTYP
Address of a 1-byte field indicating whether the log stream being created is for a system log or a general log. Valid values are:

UEPSYSLG
The log stream is for a CICS system log.

UEPGENLG
The log stream is for a general log (a forward recovery log, a user journal, or auto-journal).

Return codes

UERCNORM
CICS continues and attempts to define the log stream.

UERCBYP
CICS does not attempt to define the log stream. The process that was attempting to use the log stream may fail (for example, a data set open).

XPI calls
All can be used.

API and SPI commands
Must not be used.

Example of how to use the XLGSTRM exit

The XLGSTRM exit is used for selecting alternative model log streams.

Suppose that 200 CICS regions are running on 20 MVS images; To avoid having to define explicitly each log stream used by each CICS region, you decide to use model definitions. Log streams will be defined to MVS dynamically on their first usage, with an XLGSTRM exit program being used to select from alternative model log streams. This is how it might work:

1. On an initial start of a CICS region, the INITPARM system initialization parameter specifies:

   ```
   INITPARM=(Exit_enabler_pgmname=nnn)
   ```

   where:

   - `Exit_enabler_pgmname` is the name of the program that enables the XLGSTRM user exit program.
   - `nnn` is a number that identifies a group of CICS regions that share the same set of log stream models.
2. The program that enables the XLGSTRM user exit program issues an EXEC CICS ASSIGN INITPARM command to retrieve the value `nnn`, and places it in the exit program's global work area.
3. When the region tries to connect to its system log, because the log stream is not defined the XLGSTRM exit program is invoked. The exit program selects model CICS.DFHLOG.MODELnnn.

Message domain exit XMEOUT

You can use the XMEOUT exit to suppress or reroute CICS and CICSPlex SM messages that use the CICS message domain.

Your exit program has the following restrictions:

- It can suppress or reroute only messages sent to the system console or to transient data queues. It cannot suppress or reroute messages sent to terminal operators. XMEOUT is not invoked for messages sent to terminal operators.
- It can suppress or reroute only messages that use the message domain. You can determine which CICS messages this applies to from CICS messages. The description of each message that can invoke XMEOUT contains a list titled "XMEOUT parameters/Message inserts"; if no XMEOUT parameters are listed for a message, the message cannot invoke the exit. For example, message DFHDX8320 can invoke XMEOUT, but message DFHDU0205 does not.

For CICSPlex SM, XMEOUT is invoked only for messages that have a destination of EYULOG, because these are the messages that use the message domain. You can determine which messages this applies to from CICSPlex SM messages.

Note: CICSPlex SM messages that invoke the XMEOUT exit can be rerouted or suppressed only from the joblog or console, not from the EYULOG.

- It cannot reroute or suppress CICSPlex SM Web User interface messages.
- It cannot change the text of a message, or change the message inserts. If it attempts to do so, CICS ignores the changes.
- It cannot suppress or reroute messages issued during the early stages of CICS initialization (because the exit cannot be enabled then).
- It cannot reroute a message to transient data (TD) queues during CICS shutdown, unless the original message destination included one or more transient data queues. If it attempts to do so, the message is routed to its original destination, and message DFHME0120I is issued to the console. The user exit program cannot reroute message DFHME0120I, but it can suppress this message.

  This restriction is necessary because the message domain must handle messages during CICS shutdown even after the transient data queue function has ended.

  To discover whether CICS shutdown has started, your exit program can check for the first instance of message DFHME0120. It can stop rerouting messages to TD queues after DFHME0120 has been issued.

Note: If a message is being rerouted to a transient data queue and the transient data request fails, the message is lost. The MEME exception trace point ID X'0328' is written. The interpretation string of this trace entry explains why the transient data request failed.

Important

Because of the danger of recursion, your XMEOUT exit program must not try to reroute the following messages:

- Any DFHTDxxxx messages, which are produced by the transient data program.
- User domain messages in the range DFHUS0002 toDFHUS0006, plus message DFHUS0150.
- Transaction manager messages DFHXM0212, DFHXM0213, DFHXM0304, and DFHXM0308.
- Application messages DFHAP0001, DFHAP0002, DFHAP0004, DFHAP0601, DFHAP0602, and DFHAP0603.
- Any user domain (DFHUSxxxx) messages to an intrapartition queue defined with a TRIGLEV value of anything other than zero, if the messages are produced while the user domain is performing error recovery processing.

The message definition template contains an indicator called *noreroute*. This indicator is set on if the message that is being issued cannot be rerouted to a transient data queue by the XMEOUT exit program. The address of the indicator is passed to XMEOUT in the UEPNRTE exit-specific parameter. Your exit program can check the value of the indicator before deciding whether to reroute a particular message.

Note: If the exit program tries to reroute an ineligible message, the message domain inhibits the rerouting and issues the message to the console instead, along with message DFHME0137.

Each message that is affected by this restriction is identified by a note in CICS messages.

It is possible to pass APPLID (the application identifier) as an optional parameter in a message. However, the APPLID that is inserted in a message might not be the APPLID of the current CICS system. For example, when a CICSPlex SM MAS message is routed to a CMAS, the APPLID of the MAS system might be passed, so the message contains the APPLID of the MAS system and not the current system (the CMAS).

Your exit program can suppress or reroute messages by altering the values held in the addresses pointed to by the following fields of the parameter list. Your program cannot change any other sets of values.

- UEPMROU (route codes)
- UEPMNRC (number of route codes)
- UEPMTDQ (transient data queue names)
- UEPMNTD (number of TDQs)

Using XMEOUT to monitor DFHAP1900 message

Message DFHAP1900 is issued when a change is made to the CICS system configuration by certain system programming interface commands. These commands are **SET**, **PERFORM**, **ENABLE**, **DISABLE**, or **RESYNC**. The commands are written to the transient data queue CADS. The DFHAP1900 messages can provide auditing of dynamic configuration changes and also aid problem determination. For more information, see SPI commands that can be audited .

Note: Do not issue SPI commands in the exit which result in a DFHAP1900 message, because a recursion in the exit can occur.

Exit XMEOUT

This exit is invoked before a message domain sends a CICS message.

When invoked

Before the message domain sends a CICS message to its destination.

Exit-specific parameters

UEPTRANID

Address of the 4-byte transaction ID.

UEPUSER

Address of the 8-byte user ID.

UEPTERM

Address of the 4-byte ID of the terminal under which the current transaction is running. If the current transaction is not associated with a terminal, the addressed field contains hexadecimal zeroes.

UEPPROG

Address of the 8-byte application program name, or nulls if there is no current application.

UEPMNUM
Address of a 4-byte field containing the message number. For CICSPlex SM messages, this field contains binary zeros.

UEPMDOM
Address of a 2-byte field containing the domain identifier of the CICS message. For CICSPlex SM messages, this field contains binary zeros.

UEPMROU
Address of an array of up to 28 route codes. Route codes must be numbers in the range 1 through 28.

UEPMNRC
Address of a halfword containing the number of route codes in the route code array.

UEPMTDQ
Address of an array of up to 25 transient data queue names to which the message is to be sent. TD queue names must consist of 4 alphanumeric characters.

UEPMNTD
Address of a halfword containing the number of TDQs in the queues array.

UEPINSN
Address of a 2-byte field containing the number of message inserts.

UEPINSA
Address of an array, each element of which contains information about a single message insert. The size of the array depends on the number of inserts. Each array element has the following structure:

```
INSERT_FORMAT_P  DS  A  Address of the 1-byte insert
                        type-code, which has one of
                        the following hexadecimal values:
                        0    Not present
                        1    Character
                        2    Hexadecimal
                        3    Decimal
                        4    The insert is a number representing one item in a list of options.
                             (See the example below.)
                        7    Integer (no reformatting of numbers > 999)
 INSERT_P         DS  A  Address of the message insert
 INSERT_LENGTH_P  DS  A  Address of a fullword containing the length of the insert
 INSERT_TYPE_P    DS  A  Reserved.
```

You can find the order of the inserts in the array from the entry for the specific message in the CICS messages. For example:

DFHFC0531 *date time applid* Automatic journal *journal journalname*, opened for file *filename* is not of type MVS Module *module*

The XMEOUT inserts are *date, time, applid*, *journal*, *journalname*, *filename*, and *module*. The fourth insert (*journal*) is the number specified for JOURNAL on the file definition.

UEPNRTE
Address of 1-character flag indicating whether or not the message can be rerouted by XMEOUT. The possible values are:

C'0:' The message can be routed.

C'1:' The message cannot be routed.

UEPCPID
Address of a 3-byte product ID. The possible values are:

DFH CICS messages.

EYU CICSPlex SM messages.

UEPCPDOM
Address of a 2-byte field containing the domain identifier of the message.

UEPCPNUM
Address of a 4-byte field containing the message number.

UEPCPSEV
Address of the message severity code.

Return codes

UERCNORM
Continue processing.

UERCBYP
Suppress the message for all destinations.

CICSPlex SM messages cannot be suppressed. For these messages, a response of UERCBYP is treated as UERCNORM.

Specifying a return code of UERCBYP does not suppress the emission of any MESSAGE system events defined for the message.

XPI calls
WAIT_MVS can be used. **Do not use any other calls**.

Monitoring domain exit (XMNOUT)

This exit is invoked before monitoring records are written to SMF or to the record buffers.

XMNOUT is invoked at the following event points:

- Before an exception class monitoring record is passed to SMF
- Before a performance class monitoring record is written to the performance record buffer
- Before a transaction resource monitoring record is written to the transaction resource record buffer

Note: If performance class and transaction resource monitoring are both active in your CICS region, XMNOUT can be invoked twice for the same event. For example, if the event is end-of-task and CICS has both performance class data and transaction resource data to move to the appropriate buffer, XMNOUT is invoked once for each monitoring record type.
You can use this exit to examine the record, to suppress its output to SMF, or to change the data it contains. You must ensure that any changes you make do not conflict with the dictionary description of the data.

You can also add data to performance class data records. To do this you must define dummy user event-monitoring points (EMPs) in the monitoring control table (MCT) to reserve data fields of the required size and type.

Exit XMNOUT

This exit is invoked before monitoring records are written to SMF or buffered for subsequent writing to SMF.

When invoked

XMNOUT is invoked in these circumstances:

- Before an exception class monitoring record is written to SMF
- Before a performance class monitoring record is buffered for a later write to SMF
- Before a transaction resource monitoring record is buffered for a later write to SMF
- Before an identity class monitoring record is buffered for a later write to SMF

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID. This field is not available at task termination.

UEPUSER
Address of the 8-byte user ID. This field is not available at task termination.

UEPTERM
Address of the 4-byte terminal ID. This field is not available at task termination.

UEPPROG
Address of the 8-byte application program name. This field is not available at task termination.

UEPDICT
Address of the dictionary. The sequence of dictionary entries is mapped by the DSECT generated from the macro DFHMCTDR. This field has meaning only for performance class records. If the monitoring record type is exception class (type 4), transaction resource (type 5), or identity class (type 6), this field is set to 0. See parameter UEPMRTYP.

UEPDICTE
Address of the fullword number of dictionary entries. This field has meaning only for performance class records. If the monitoring record type is exception class (type 4), transaction resource (type 5), or identity class (type 6), this field is set to 0. See parameter UEPMRTYP.

UEPFCL
Address of the field connector list, containing a series of halfword connector values. This field has meaning only for performance class records. If the monitoring record type is exception class (type 4), transaction resource (type 5), or identity class (type 6), this field is set to 0. See parameter UEPMRTYP.

UEPFCLNO
Address of the fullword number of field connectors. This field has meaning only for performance class records. If the monitoring

record type is exception class (type 4), transaction resource (type 5), or identity class (type 6), this field is set to 0. See parameter UEPMRTYP.

UEPMRTYP
Address of the halfword monitoring record type. The monitoring record type value can be one of the numbers shown in the following table:

Table 7. Monitoring record type values and their meanings

Record type value	Meaning
3	Performance class monitoring record
4	Exception class monitoring record
5	Transaction resource monitoring record
6	Identity class monitoring record

UEPMRLEN
Address of the fullword monitoring record length.

UEPMREC
Address of the monitoring record. The length of UEPMREC is addressed by the parameter UEPMRLEN.

UEPSRCTK
Address of the z/OS Workload Manager service reporting class token for the current transaction. If CICS support for z/OS Workload Manager is not available, this token is null.

UEPMPREC
Address of the monitoring performance record. This field has meaning only for performance class records. If the monitoring record type is exception class (type 4), transaction resource (type 5), or identity class (type 6), this field is set to 0. See parameter UEPMRTYP. The performance record addressed by this parameter must be mapped using the DFHMNTDS DSECT, and must not be mapped using the UEPDICT and UEPDICTE dictionary parameters.

Return codes

UERCNORM
Continue processing.

UERCBYP
Suppress monitor record output.

UERCPURG
Task purged during XPI call.

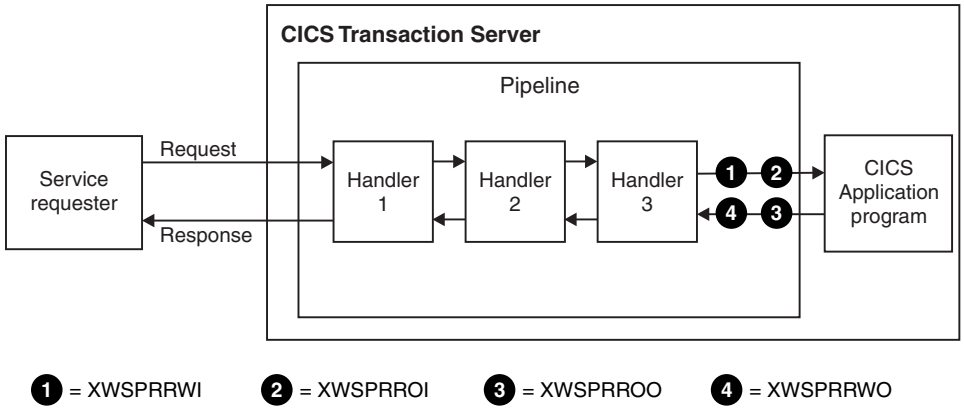
XPI calls
WAIT_MVS can be used. **Do not use any other calls**.

Pipeline domain exits

Use the pipeline domain exits to customize the processing that occurs for inbound and outbound web services in the pipeline. You can use the pipeline domain exits to access containers on a Web services provider pipeline, a web services requester pipeline, or a web services requester pipeline that contains a security message handler.

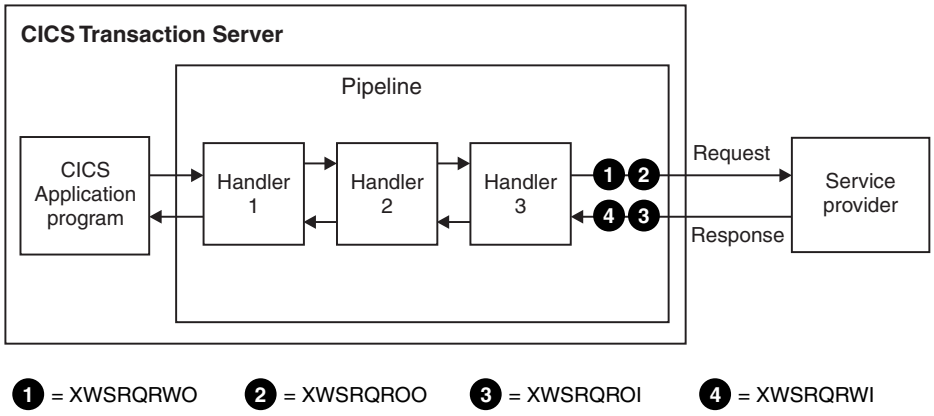
GLUE points in the provider pipeline

Global user exit (GLUE) points that you can use in a provider pipeline, or a secured provider pipeline, have a prefix of XWSPR. This diagram shows the order in which the GLUE points can be used:

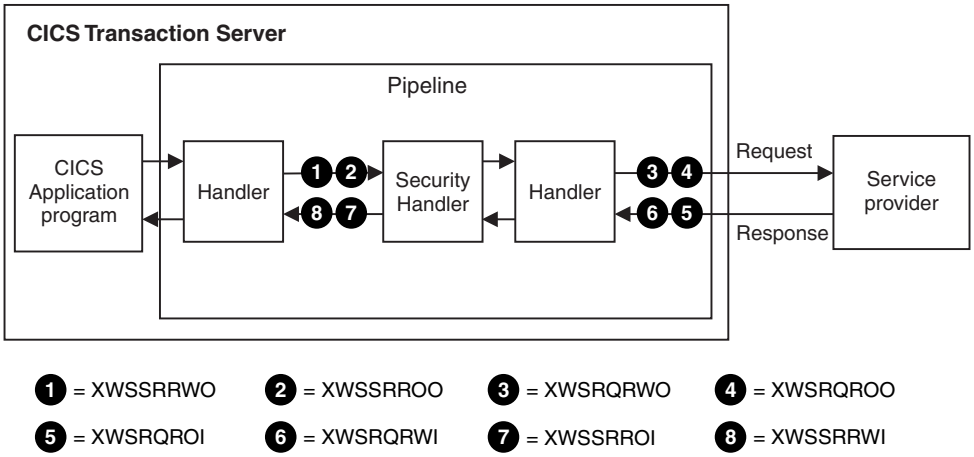


GLUE points in the requester pipeline

GLUE points that you can use in a requester pipeline have a prefix of XWSRQ. This diagram shows the order in which the GLUE points can be used:



GLUE points that you can use in a secured requester pipeline have a prefix of XWSSR. There are eight GLUE points that can be used in a pipeline containing a security handler; four of these can be used only in a secure requester pipeline and four can be used in any requester pipeline. This diagram shows the order in which the GLUE points can be used:



Exit XWSPRRWI

Use the XWSPRRWI exit to access containers on the current channel that are to be processed by the web services provider application, after CICS has converted the Web services request body into a language structure and before any instance of the XWSPRROI exit is invoked.

You can use this exit to issue API and SPI commands to examine and update any information in the containers and to issue a SOAP fault.

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID. The exit or task runs under this transaction ID.

UEPUSER
Address of the 8-byte user ID. The user ID is the one associated with the transaction ID.

UEPTERM
Address of the 4-byte terminal ID. The value is null for a Web service provider.

UEPPROG
Address of the 8-byte application program name. The application program name is that of the program that issued the **INVOKE SERVICE** or **INVOKE WEBSERVICE** command.

UEPCHANN
Address of a 16-byte field that contains the name of the current channel. You can identify the channel explicitly using this parameter.

UEPCONTR
Address of a 16-byte field that contains the name of the data container on the channel named by UEPCHANN. This container holds the application data structure.

Return codes

UERCNORM
Continue processing.

UERCRPIP
Do not continue on the pipeline.

XPI calls

No XPI interfaces are available.

API and SPI commands

The following commands are supported:

- **EXEC CICS DELETE CONTAINER**
- **EXEC CICS GET CONTAINER**
- **EXEC CICS INQUIRE WEBSERVICE**
- **EXEC CICS PUT CONTAINER**
- **EXEC CICS SOAPFAULT ADD**
- **EXEC CICS SOAPFAULT CREATE**
- **EXEC CICS SOAPFAULT DELETE**

Exit XWSPRROI

Use the XWSPRROI exit to access containers on the current channel before the containers are processed by a web services provider application, but after any instance of the XWSPRRWI exit is invoked.

You can use this exit to issue API and SPI commands to examine any information that is processed by the web services business application. You cannot issue a SOAP fault or update any of the information. CICS ignores any return code specified in register 15 after the global user exit program has finished.

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID. The exit or task runs under this transaction ID.

UEPUSER
Address of the 8-byte user ID. The user ID is the one associated with the transaction ID.

UEPTERM
Address of the 4-byte terminal ID. The value is null for a Web service provider.

UEPPROG
Address of the 8-byte application program name. The application program name is that of the program that issued the **INVOKE SERVICE** or **INVOKE WEBSERVICE** command.

UEPCHANN
Address of a 16-byte field that contains the name of the current channel. You can identify the channel explicitly using this parameter.

UEPCONTR
Address of a 16-byte field that contains the name of the data container on the channel named by UEPCHANN. This container holds the application data structure.

Return codes

UERCNORM
Continue processing.

XPI calls

No XPI interfaces are available.

API and SPI commands

You can use the following commands:

- **EXEC CICS GET CONTAINER**
- **EXEC CICS INQUIRE WEBSERVICE**

Exit XWSPRROO

Use the XWSPRROO exit to access containers on the current channel after the web services provider application issues the web service response message and before CICS creates the body of the response message.

You can use this exit to issue API and SPI commands to examine the containers on the current channel. You cannot issue a SOAP fault or update any of the containers. CICS ignores any return code specified in register 15 after the global user exit program has finished.

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID. The exit or task runs under this transaction ID.

UEPUSER
Address of the 8-byte user ID. The user ID is the one associated with the transaction ID.

UEPTERM
Address of the 4-byte terminal ID. The value is null for a Web service provider.

UEPPROG
Address of the 8-byte application program name. The application program name is that of the program that issued the **INVOKE SERVICE** or **INVOKE WEBSERVICE** command.

UEPCHANN
Address of a 16-byte field that contains the name of the current channel. You can identify the channel explicitly using this parameter.

UEPCONTR
Address of a 16-byte field that contains the name of the data container on the channel named by UEPCHANN. This container holds the application data structure.

UEPAPAB
A 1-byte field indicating whether the web service provider application completed its processing successfully. Valid values are as follows:

UEPAPABY (X'80')
The web service provider application abended.

UEPAPABN (X'40')
The web service provider application completed its processing successfully.

UEPAPSF
A 1-byte field indicating whether the web service provider application set a SOAP fault. Valid values are as follows:

UEPAPSFY (X'80')
The web service provider application is returning a SOAP fault.

UEPAPSFN (X'40')
The web service provider application is not returning a SOAP fault.

Return codes

UERCNORM
Continue processing.

XPI calls

No XPI interfaces are available.

API and SPI commands

The following commands are supported:

- `EXEC CICS GET CONTAINER`
- `EXEC CICS INQUIRE WEBSERVICE`

Exit XWSPRRWO

Use the XWSPRRWO exit to access containers on the current channel that have been processed by a Web services provider application after any instance of the XWSPRROO exit.

You can use this exit to issue API and SPI commands to examine and update any information in the containers and to issue a SOAP fault. For example, you can add additional SOAP headers to an outbound SOAP response. Any updates to the current channel container data are processed by CICS and returned to the requester. CICS ignores any return code specified in register 15 after the global user exit program has finished.

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID. The exit or task runs under this transaction ID.

UEPUSER
Address of the 8-byte user ID. The user ID is the one associated with the transaction ID.

UEPTERM
Address of the 4-byte terminal ID. The value is null for a Web service provider.

UEPPROG
Address of the 8-byte application program name. The application program name is that of the program that issued the **INVOKE SERVICE** or **INVOKE WEBSERVICE** command.

UEPCHANN
Address of a 16-byte field that contains the name of the current channel. You can identify the channel explicitly using this parameter.

UEPCONTR
Address of a 16-byte field that contains the name of the data container on the channel named by UEPCHANN. This container holds the application data structure.

UEPAPAB
A 1-byte field indicating whether the web service provider application completed its processing successfully. Valid values are as follows:

UEPAPABY (X'80')
The web service provider application abended.

UEPAPABN (X'40')
The web service provider application completed its processing successfully.

UEPAPSF
A 1-byte field indicating whether the web service provider application set a SOAP fault. Valid values are as follows:

UEPAPSFY (X'80')
The web service provider application is returning a SOAP fault.

UEPAPSFN (X'40')
The web service provider application is not returning a SOAP fault.

Return codes

UERCNORM
Continue processing.

XPI calls

No XPI interfaces are available.

API and SPI commands

You can use the following commands:

- **EXEC CICS DELETE CONTAINER**
- **EXEC CICS GET CONTAINER**
- **EXEC CICS INQUIRE WEBSERVICE**
- **EXEC CICS PUT CONTAINER**
- **EXEC CICS SOAPFAULT ADD**
- **EXEC CICS SOAPFAULT CREATE**
- **EXEC CICS SOAPFAULT DELETE**

Exit XWSRQRWO

Use the XWSRQRWO exit to access containers on the current channel before they are passed to the transport to be processed. This exit runs after CICS has converted the application's language structure into a web services request body and before CICS processes the optional XWSRQROO exit point.

You can use this exit to issue API and SPI commands to examine and update any information in the containers on the current channel. This information is available to any instance of the XWSRQRWO exit and also to the outbound web services provider. You cannot issue a SOAP fault. A return code can be put in register 15 to indicate that CICS does not continue the pipeline after the global user exit program finishes.

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID. The exit or task runs under this transaction ID.

UEPUSER
Address of the 8-byte user ID. The user ID is the one associated with the transaction ID.

UEPTERM
Address of the 4-byte terminal ID.

UEPPROG
Address of the 8-byte application program name. The application program name is that of the program that issued the **INVOKE SERVICE** or **INVOKE WEBSERVICE** command.

UEPPROG
Address of the 8-byte application program name. The application program name is that of the program which issued the **INVOKE SERVICE** or **INVOKE WEBSERVICE** command.

UEPCHANN
Address of a 16-byte field that contains the name of the current channel. You can identify the channel explicitly using this parameter.

UEPCONTR
Address of a 16-byte field that contains the name of the data container on the channel named by UEPCHANN. This container holds the application data structure.

Return codes

UERCNORM
Continue processing.

UERCRPIP
Do not continue on the pipeline.

XPI calls

No XPI interfaces are available.

API and SPI commands

You can use the following commands:

- **EXEC CICS DELETE CONTAINER**
- **EXEC CICS GET CONTAINER**
- **EXEC CICS INQUIRE WEBSERVICE**
- **EXEC CICS PUT CONTAINER**

Exit XWSRQROO

Use the XWSRQROO exit to access containers on the current channel before they are passed to the transport to be processed. This exit runs after any instance of the XWSRQRWO exit is processed and before the data flowing outbound on the Web services transport.

You can use this exit to issue API and SPI commands to examine any information in the containers. This information is processed by the outbound web services provider. You cannot issue a SOAP fault. CICS ignores any return code specified in register 15 after the global user exit program has finished.

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID. The exit or task runs under this transaction ID.

UEPUSER
Address of the 8-byte user ID. The user ID is the one associated with the transaction ID.

UEPTERM
Address of the 4-byte terminal ID.

UEPPROG
Address of the 8-byte application program name. The application program name is that of the program that issued the **INVOKE SERVICE** or **INVOKE WEBSERVICE** command.

UEPCHANN
Address of a 16-byte field that contains the name of the current channel. You can identify the channel explicitly using this parameter.

UEPCONTR
Address of a 16-byte field that contains the name of the data container on the channel named by UEPCHANN. This container holds the application data structure.

Return codes

UERCNORM
Continue processing.

UERCRPIP
Do not continue on the pipeline.

XPI calls

No XPI interfaces are available.

API and SPI commands

You can use the following commands:

- **EXEC CICS GET CONTAINER**
- **EXEC CICS INQUIRE WEBSERVICE**

Exit XWSRQROI

Use the XWSRQROI exit to access containers on the current channel after they are processed by the transport as a web services response. The XWSRQROI exit is invoked directly after CICS has processed the outbound web service provider. It can also be invoked before any instance of the XWSRQRWI exit.

You can use this exit to issue API and SPI commands to examine any information in the containers. This information is processed by the outbound web services provider. You cannot issue a SOAP fault. CICS ignores any return code specified in register 15 after the global user exit program has finished.

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID. The exit or task runs under this transaction ID.

UEPUSER
Address of the 8-byte user ID. The user ID is the one associated with the transaction ID.

UEPTERM
Address of the 4-byte terminal ID.

UEPPROG
Address of the 8-byte application program name. The application program name is that of the program that issued the **INVOKE SERVICE** or **INVOKE WEBSERVICE** command.

UEPCHANN
Address of a 16-byte field that contains the name of the current channel. You can identify the channel explicitly using this parameter.

UEPCONTR
Address of a 16-byte field that contains the name of the data container on the channel named by UEPCHANN. This container holds the application data structure.

Return codes

UERCNORM
Continue processing.

XPI calls

No XPI interfaces are available.

API and SPI commands

You can use the following commands:

- **EXEC CICS GET CONTAINER**
- **EXEC CICS INQUIRE WEBSERVICE**

Exit XWSRQRWI

Use the XWSRQRWI exit to access containers on the current channel after they have been processed by the transport as a web services response. The XWSRQRWI exit is invoked directly after CICS has processed the inbound web service response. It is also invoked after any instance of the XWSRQROI exit.

You can use this exit to issue API and SPI commands to examine and update any information in the containers. This information is processed by the outbound web services provider and is received by the associated web services requester application. You cannot issue a SOAP fault. CICS ignores any return code specified in register 15 after the global user exit program has finished.

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID. The exit or task runs under this transaction ID.

UEPUSER
Address of the 8-byte user ID. The user ID is the one associated with the transaction ID.

UEPTERM
Address of the 4-byte terminal ID.

UEPPROG
Address of the 8-byte application program name. The application program name is that of the program that issued the **INVOKE SERVICE** or **INVOKE WEBSERVICE** command.

UEPCHANN
Address of a 16-byte field that contains the name of the current channel. You can identify the channel explicitly using this parameter.

UEPCONTR
Address of a 16-byte field that contains the name of the data container on the channel named by UEPCHANN. This container holds the application data structure.

Return codes

UERCNORM
Continue processing.

XPI calls

No XPI interfaces are available.

API and SPI commands

You can use the following commands:

- **EXEC CICS DELETE CONTAINER**
- **EXEC CICS GET CONTAINER**
- **EXEC CICS INQUIRE WEBSERVICE**
- **EXEC CICS PUT CONTAINER**

Exit XWSSRRWO

Use the XWSSRRWO exit to access containers on the current channel, with CICS acting as a secured web services requester, before they are passed to the transport to be processed. This exit runs after CICS converts the application's language structure into a web services request body and before CICS processes the optional XWSSRROO exit point, and before being encrypted by the pipeline's security handler.

You can use this exit to issue API and SPI commands to examine and update any information in the containers. This information is available to any instance of the XWSSRRWO exit and also to the outbound web services provider. You cannot issue a SOAP fault. A return code can be specified in register 15 to indicate that CICS does not continue the pipeline after the global user exit program finishes.

If the pipeline does not contain a security handler, this exit is not driven. See the "Exit XWSRQRWO" on page 176 topic for instances of the pipeline not containing a security handler.

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID. The exit or task runs under this transaction ID.

UEPUSER
Address of the 8-byte user ID. The user ID is the one associated with the transaction ID.

UEPTERM
Address of the 4-byte terminal ID.

UEPPROG
Address of the 8-byte application program name. The application program name is that of the program that issued the **INVOKE SERVICE** or **INVOKE WEBSERVICE** command.

UEPCHANN
Address of a 16-byte field that contains the name of the current channel. You can identify the channel explicitly using this parameter.

UEPCONTR
Address of a 16-byte field that contains the name of the data container on the channel named by UEPCHANN. This container holds the application data structure.

Return codes

UERCNORM
Continue processing.

UERCRPIP
Do not continue on the pipeline.

XPI calls

No XPI interfaces are available.

API and SPI commands

You can use the following commands:

- **EXEC CICS DELETE CONTAINER**
- **EXEC CICS GET CONTAINER**
- **EXEC CICS INQUIRE WEBSERVICE**
- **EXEC CICS PUT CONTAINER**

Exit XWSSRROO

Use the XWSSRROO exit to access containers on the current channel, with CICS acting as a secured web services requester, before they are passed to the transport to be processed. This exit runs after any instance of the XWSSRRWO exit is processed and before the encryption of data flowing outbound on the web services transport.

If the pipeline does not contain a security handler, this exit is not driven. See the "Exit XWSRQROO" on page 177 topic for instances of the pipeline not containing a security handler

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID. The exit or task runs under this transaction ID.

UEPUSER
Address of the 8-byte user ID. The user ID is the one associated with the transaction ID.

UEPTERM
Address of the 4-byte terminal ID.

UEPPROG
Address of the 8-byte application program name. The application program name is that of the program that issued the **INVOKE SERVICE** or **INVOKE WEBSERVICE** command.

UEPCHANN
Address of a 16-byte field that contains the name of the current channel. You can identify the channel explicitly using this parameter.

UEPCONTR
Address of a 16-byte field that contains the name of the data container on the channel named by UEPCHANN. This container holds the application data structure.

Return codes

UERCNORM
Continue processing.

XPI calls

No XPI interfaces are available.

API and SPI commands

You can use the following commands:

- **EXEC CICS GET CONTAINER**
- **EXEC CICS INQUIRE WEBSERVICE**

Exit XWSSRROI

Use the XWSRQROI exit to access containers on the current channel, with CICS acting as a secured web services requester, after they are processed by the transport as a web services response. This exit runs after CICS processes the web service response and before any instance of the XWSSRRWI exit.

You can use this exit to issue API and SPI commands to examine any information in the containers. This information is processed by the outbound web services provider. You cannot issue a SOAP fault. CICS ignores any return code specified in register 15 after the global user exit program has finished.

If the pipeline does not contain a security handler, this exit is not driven. See the "Exit XWSRQROI" on page 178 topic for instances of the pipeline not containing a security handler.

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID. The exit or task runs under this transaction ID.

UEPUSER
Address of the 8-byte user ID. The user ID is the one associated with the transaction ID.

UEPTERM
Address of the 4-byte terminal ID.

UEPPROG
Address of the 8-byte application program name. The application program name is that of the program that issued the **INVOKE SERVICE** or **INVOKE WEBSERVICE** command.

UEPCHANN
Address of a 16-byte field that contains the name of the current channel. You can identify the channel explicitly using this parameter.

UEPCONTR
Address of a 16-byte field that contains the name of the data container on the channel named by UEPCHANN. This container holds the application data structure.

Return codes

UERCNORM
Continue processing.

XPI calls

No XPI interfaces are available.

API and SPI commands

You can use the following commands:

- **EXEC CICS GET CONTAINER**
- **EXEC CICS INQUIRE WEBSERVICE**

Exit XWSSRRWI

Use the XWSSRRWI exit to access containers on the current channel, with CICS acting as a secured web services requester, after they have been processed by the transport as a web services response. This exit runs after CICS processes the web service response and after any instance of the XWSSRROI exit.

You can use this exit to issue API and SPI commands to examine and update any information in the containers. This information is processed by the web services requester application. You cannot issue a SOAP fault. CICS ignores any return code specified in register 15 after the global user exit program has finished.

If the pipeline does not contain a security handler, this exit is not driven. See the "Exit XWSRQRWI" on page 179 topic for instances of the pipeline not containing a security handler.

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID. The exit or task runs under this transaction ID.

UEPUSER
Address of the 8-byte user ID. The user ID is the one associated with the transaction ID.

UEPTERM
Address of the 4-byte terminal ID.

UEPPROG
Address of the 8-byte application program name. The application program name is that of the program that issued the **INVOKE SERVICE** or **INVOKE WEBSERVICE** command.

UEPCHANN
Address of a 16-byte field that contains the name of the current channel. You can identify the channel explicitly using this parameter.

UEPCONTR
Address of a 16-byte field that contains the name of the data container on the channel named by UEPCHANN. This container holds the application data structure.

Return codes

UERCNORM
Continue processing.

XPI calls

No XPI interfaces are available.

API and SPI commands

You can use the following commands:

- **EXEC CICS DELETE CONTAINER**
- **EXEC CICS GET CONTAINER**
- **EXEC CICS INQUIRE WEBSERVICE**
- **EXEC CICS PUT CONTAINER**

Program control program exits (XPCREQ, XPCERES, XPCREQC, XPCFTCH, XPCHAIR, XPCTA, and XPCABND)

These exits are invoked before or after CICS program control operations, including program link requests, a program receiving control, and a transaction abend.

Program control exits XPCREQ, XPCERES, XPCREQC

These exits are called by the EXEC interface program before a link request is processed, before CICS processes dynamically routed link requests, or after a link request has completed.

XPCREQ

XPCREQ is called by the EXEC interface program before a link request is processed. If the request is a distributed program link, the XPCREQ exit is driven on both sides of the link; that is, in both the client and the server regions. The exit program is passed the address of the application's parameter list (in UEPCLPS), and can modify this list as required. For example, you can use this exit to modify the SYSID at the time of a distributed program link request. You can write an application program to manage a list of SYSIDs in a global work area (GWA). The global user exit program can obtain access to the GWA, and use the information stored there to redirect DPL requests.

Note:

1. The attributes of the local PROGRAM resource are not passed to the exit program. If the exit program requires the value of an attribute, it can issue an **EXEC CICS INQUIRE PROGRAM** command.
2. If you use XPCREQ to change the target SYSID, remember that:
   a. If SYSID specifies a remote region, no reference is made to the local PROGRAM resource. In the remote region the program runs under the TRANSID of the transaction in the client region, not under the TRANSID specified on the PROGRAM resource in the client region.
   b. If SYSID specifies the local region, CICS treats the link request as if SYSID was not specified. The local PROGRAM resource is honored.
   c. The XPCREQ exit is called by internal requests made by CICS code, in addition to requests made by applications.

XPCERES

XPCERES is called by the EXEC interface program before CICS processes either of the following kinds of dynamically routed link request:

- A distributed program link (DPL) call
- A Link3270 bridge request

XPCERES is called:

- After exit XPCREQ and before XPCREQC if these exits are enabled:
  - If an XPCREQ exit program chooses to bypass the request, XPCERES is not called.

– If an XPCREQ exit program modifies the command parameter list, XPCERES must deal with the modified request.

- On the target region to which the request has been routed.
- Only if it is enabled. Enable this exit only in application-owning regions when DPL and Link3270 bridge requests can be dynamically routed.
- By internal requests made by CICS code, in addition to requests made by applications.

The XPCERES exit is not called:

- For statically routed requests.
- If it is disabled.
- If an XPCREQ exit program chooses to bypass the request.

You can use XPCERES to check that all resources required by the linked-to program are available on the target region. If the program is disabled or a required file is missing, your exit program can give the dynamic routing program the opportunity to route the request to a different region. Set a return code of UERCRESU. CICS performs the following processing:

1. In the COMMAREA of the routing program, CICS sets the DYRERROR field to 'F' - resource unavailable.
2. CICS calls the routing program, on the routing region, for route selection failure.
3. CICS returns a RESUNAVAIL condition on the EXEC CICS LINK command that was run by the mirror on the target region. This condition is not returned to the application program.

CICS ignores any changes made by the exit program to the values of any of the exit parameters. Your exit program can set a return code, but not change any parameters.

For guidance information about dynamically routing DPL requests, see Dynamically routing DPL requests. For guidance information about dynamically routing Link3270 bridge requests, see Using Link3270 bridge load balancing. For programming information about writing a dynamic routing program to route DPL requests, see Routing DPL requests dynamically. For programming information about writing a dynamic routing program to route Link3270 bridge requests, see Routing bridge requests dynamically.

XPCREQC

XPCREQC is called after a link request has completed. You can use this exit to pass back a response to the application by using the EIBRESP or EIBRESP2 fields. Such responses might be used to keep status information about a link request up-to-date. For example, if a link request fails because a connection is unavailable, XPCREQC can set EIBRESP=500 (a response code not used by CICS) to indicate the failure, enabling the application, with the other exit XPCREQ, to determine a suitable course of action.

Note: The XPCREQC exit is called by internal requests made by CICS code, in addition to requests made by applications.

Exit XPCREQ

Exit XPCREQ is invoked by the EXEC interface program before a link request is processed.

When invoked

By the EXEC interface program before a link request is processed.

Exit-specific parameters

UEPCLPS
Address of the command parameter list.

UEPPCTOK
Address of a 4-byte token to be passed to XPCREQC. This allows you, for example, to pass a work area to exit XPCREQC.

UEPRCODE
Address of a 6-byte hexadecimal copy of EIBRCODE.

UEPRECUR
Address of a halfword recursion counter. The counter is set to 0 when the exit is first invoked, and is incremented for each recursive call.

UEPRESP
Address of a 4-byte copy of EIBRESP.

UEPRESP2
Address of a 4-byte copy of EIBRESP2.

UEPTSTOK
Address of a 4-byte token that is valid throughout the life of a task. See Using the task token UEPTSTOK.

UEPRSRCE
Address of an 8-character copy of the EIB resource value, EIBRSRCE.

UEP_PC_PBTOK
Address of a 4-byte field containing the z/OS Workload Manager (WLM) Performance Block Token. An exit program can use this token to access information (such as the service class token, SERVCLS) in the WLM Performance Block. To do so, it must use the WLM EXTRACT macro, IWMMEXTR, passing the Performance Block Token as the MONTKN input parameter. For more information about the IWMMEXTR macro, see z/OS MVS Programming: Workload Management Services.

An exit program must not attempt to modify the Performance Block: if it does so, the results are unpredictable.

Return codes

UERCBYP
Program control is to ignore the request.

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

Although the exit permits the use of XPI GETMAIN and FREEMAIN calls, we recommend that you use the EXEC CICS GETMAIN and FREEMAIN commands instead.

API and SPI calls
All can be used, except for:

EXEC CICS SHUTDOWN
EXEC CICS XCTL

Exit XPCERES

Exit XPCERES is invoked by the EXEC interface program, before processing of a program link or Link3270 bridge request that has been dynamically routed to this region, where the routing region supports the “resource unavailable” condition.

Exit-specific parameters

Note: CICS ignores any changes made by the exit program to the values of any of the exit parameters. Your exit program can set a return code, but not change any parameters.

UEPCLPS
Address of the command parameter list.

UEPPCTOK
Address of a 4-byte token to be passed to XPCREQC.

UEPRCODE
Address of a 6-byte hexadecimal copy of EIBRCODE.

UEPRECUR
Address of a halfword recursion counter. Because the XPCERES exit can never be called recursively in the same transaction, the value of this field is always 0.

UEPRESP
Address of a 4-byte copy of EIBRESP.

UEPRESP2
Address of a 4-byte copy of EIBRESP2.

UEPTSTOK
Address of a 4-byte token that is valid throughout the life of a task. See Using the task token UEPTSTOK.

UEPRSRCE
Address of an 8-character copy of the EIB resource value, EIBRSRCE.

UEP_PC_PBTOK
Address of a 4-byte field containing the z/OS Workload Manager (WLM) Performance Block Token. An exit program can use this token to access information (such as the service class token, SERVCLS) in the WLM Performance Block. To do so, it must use the WLM EXTRACT macro, IWMMEXTR, passing the Performance Block Token as the MONTKN input parameter. For more information about the IWMMEXTR macro, see z/OS MVS Programming: Workload Management Services.

An exit program must not attempt to modify the Performance Block: if it does so, the results are unpredictable.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

UERCRESU
A required resource is unavailable. Setting this value causes CICS to reject the routed request, and to return a value of 'F' (resource unavailable) in the DYRERROR field of the routing program's communications area.

XPI calls
All can be used.

Although the exit permits the use of XPI GETMAIN and FREEMAIN calls, we recommend that you use the EXEC CICS GETMAIN and FREEMAIN commands instead.

API and SPI calls
All except EXEC CICS SHUTDOWN and EXEC CICS XCTL can be used.

Exit XPCREQC

Exit XPCREQC is invoked on completion of a program control link request.

When invoked
On completion of a program control link request.

Exit-specific parameters

UEPCLPS
Address of the command parameter list.

UEPPCTOK
Address of a 4-byte token passed from XPCREQ. This allows XPCREQ to, for example, pass a work area to XPCREQC.

UEPRCODE
Address of a 6-byte hexadecimal copy of EIBRCODE.

UEPRECUR
Address of a halfword recursion counter. The counter is set to 0 when the exit is first invoked, and is incremented for each recursive call.

UEPRESP
Address of a 4-byte copy of EIBRESP.

UEPRESP2
Address of a 4-byte copy of EIBRESP2.

UEPTSTOK
Address of a 4-byte token that is valid throughout the life of a task. See Using the task token UEPTSTOK.

UEPRSRCE
Address of an 8-character copy of the EIB resource value, EIBRSRCE.

UEP_PC_REMOTE_SYSTEM
If the request is to be sent to a remote region, is the address of an area containing the 4-byte name of the remote region. (The remote region may have been specified by, for example, the SYSID option of the EXEC CICS LINK command, function shipping, work-load management, or the REMOTESYSTEM option of the PROGRAM definition.)

If the request is to be executed on the local region, this parameter is the address of a 4-byte area containing blanks.

UEP_PC_REMOTE_NAME
If the program is to be executed in a remote system, is the address of an area containing the name of the program, as it is known in the remote system.

UEP_PC_PBTOK
Address of a 4-byte field containing the z/OS Workload Manager (WLM) Performance Block Token. An exit program can use this token to access information (such as the service class token, SERVCLS) in the WLM Performance Block. To do so, it must use the WLM EXTRACT macro, IWMMEXTR, passing the Performance Block Token as the MONTKN input parameter. For more information about the IWMMEXTR macro, see z/OS MVS Programming: Workload Management Services.

An exit program must not attempt to modify the Performance Block: if it does so, the results are unpredictable.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

Although the exit permits the use of XPI GETMAIN and FREEMAIN calls, we recommend that you use the EXEC CICS GETMAIN and FREEMAIN commands instead.

API and SPI calls
All can be used, except for:
EXEC CICS SHUTDOWN
EXEC CICS XCTL

Note: Take care when issuing recursive commands not to cause a loop. For example, it is your responsibility to avoid entering a loop when a program control request is issued from the XPCREQ or XPCREQC exits.

Use of the recursion counter UEPRECUR is recommended.

The command parameter structure

The command parameter structure consists of a series of addresses. The first address points to the EXEC interface descriptor (EID), which consists of a bit string that describes the type of request and identifies each keyword specified with the request.

The remaining addresses point to pieces of data associated with the request; for instance, the second address always points to the program name. You can examine the parameters in the list to determine the values of the keywords. You can also modify values of parameters specified on the request. For example, you could change the name of the program involved in the request, or add the SYSID to route the link request to a remote system.

End of parameter list indicator

The high-order bit is set on in the last address set in the parameter list to indicate that it is the last one in the list. On return from your user exit program, CICS scans the parameter list for the high-order bit to find the last parameter. Therefore, if you modify the length of the parameter list, you must also reset the high-order bit to indicate which is the new last address.

For example, if the parameter list specifies only the first two addresses (PC_ADDR0, the address of the EID, and PC_ADDR1, the address of the name of the program named in the link request), the high-order bit is set on in PC_ADDR1. If you extend the parameter list by setting the address of a SYSID in PC_ADDR7, you must unset the high-order bit in PC_ADDR1 and set it on in PC_ADDR7 instead.

The original parameter list, as it was before XPCREQ was invoked, is restored after the completion of XPCREQC. It follows that EDF will display the original command before **and** after execution: **EDF will not display any changes made by the exit**.

The UEPCLPS exit-specific parameter:

The UEPCLPS exit-specific parameter is the address of the command-level parameter structure, and is included in exits XPCREQ and XPCREQC.

The command-level parameter structure contains 11 addresses, PC_ADDR0 through PC_ADDRA. It is defined in the DSECT PC_ADDR_LIST, which you should copy into your exit program by including the statement COPY DFHPCEDS.

The command-level parameter list is made up as follows:

PC_ADDR0
is the address of a 7-byte area called the EXEC interface descriptor (EID), which is made up as follows:

- **PC_GROUP**
- **PC_FUNCT**
- **PC_BITS1**
- **PC_BITS2**
- **PC_EIDOPT4**
- **PC_EIDOPT5**
- **PC_EIDOPT6**

PC_GROUP
Always X'0E', indicating that this is a program control request.

PC_FUNCT
One byte which defines the type of request, which for XPCREQ and XPCREQC is always X'02', indicating a LINK request.

PC_BITS1
Existence bits that define which keywords that contain values were specified. To obtain the value associated with a keyword, you need to use the appropriate address from the command-level parameter list. Before using this address you must check the associated existence bit to ensure that the address is valid. If the existence bit is set off, the

keyword was not specified in the request and the address should not be used. The symbolic and hexadecimal values of the existence bits are as follows:

PC_EXIST1 (X'80 ')
Set if the request contains the keyword PROGRAM. If set, **PC_ADDR1** is meaningful. (This bit should always be set for a LINK request.)

PC_EXIST2 (X'40 ')
Set if the request specifies the COMMAREA parameter. If set, **PC_ADDR2** is meaningful.

PC_EXIST3 (X'20 ')
Set if the request specifies the LENGTH parameter. If set, **PC_ADDR3** is meaningful.

PC_EXIST4 (X'10 ')
Set if the request specifies the INPUTMSG parameter. If set, **PC_ADDR4** is meaningful.

PC_EXIST5 (X'08 ')
Set if the request specifies the INPUTMSGLEN parameter. If set, **PC_ADDR5** is meaningful.

PC_EXIST6 (X'04 ')
Set if the request specifies the DATALENGTH parameter. If set, **PC_ADDR6** is meaningful.

PC_EXIST7 (X'02 ')
Set if the request specifies the SYSID parameter. If set, **PC_ADDR7** is meaningful.

PC_EXIST8 (X'01 ')
Set if the request specifies the TRANSID parameter. If set, **PC_ADDR8** is meaningful.

PC_BITS2
One byte containing one of the following values:

PC_EXIST9 (X'80')
Not used.

PC_EXISTA (X'40')
Set if the request specifies the CHANNEL parameter. If set, PC_ADDRA is meaningful.

PC_EIDOPT4
Not used by program control.

PC_EIDOPT5
Not used by program control.

PC_EIDOPT6
Indicates whether the request specifies the SYNCONRETURN option. If it does, X'80' is set.

PC_ADDR1
is the address of an 8-byte area containing the program name from the PROGRAM parameter.

PC_ADDR2
is the address of the COMMAREA data.

PC_ADDR3
is the address of a 2-byte area containing the length of the COMMAREA, as a half-word binary value.

PC_ADDR4
is the address of the INPUTMSG data.

PC_ADDR5
is the address of a 2-byte area containing the length of the INPUTMSG, as a half-word binary value.

PC_ADDR6
is the address of a 2-byte area containing the length specified on the DATALENGTH parameter, defining how much data is to be sent from the COMMAREA. The length is held as a half-word binary value.

PC_ADDR7
is the address of the 4-byte name of the remote system the LINK request is to be shipped to, as specified on the SYSID parameter.

PC_ADDR8
is the address of the 4-byte name of the mirror transaction to be attached in the remote system, as specified on the TRANSID parameter.

PC_ADDR9
is not used.

PC_ADDRA
is the address of the 16-byte channel name, as specified on the CHANNEL parameter.

Modifying fields in the command parameter structure

Some fields that are passed to program control are used as input to the request, some are used as output fields, and some are used for both input and output. The method your user exit program uses to modify a field depends on the usage of the field.

Modifying input fields:

The correct method of modifying an input field is to create a new copy of it, and to change the address in the command parameter list to point to your new data.

Note: You must never modify an input field by altering the data that is pointed to by the command parameter list. To do so would corrupt storage belonging to the application program and could cause a failure when the program attempted to reuse the field.

Modifying output fields: The technique described in "Modifying input fields" is not suitable for modifying output fields. (The results would be returned to the new area instead of the application's area, and would be invisible to the application.)

An output field is modified by altering the data that is pointed to by the command-level parameter list. In the case of an output field you can modify the application's data in place, because the application is expecting the field to be modified anyway.

Modifying the EID

It is not possible to modify the EID to make major changes to requests. It is not possible, for example, to change a LINK request to a different type of Program

Control request. However, you can make minor changes to requests, such as to turn on the existence bit for SYSID so that the request can be changed into one that is shipped to a remote system.

The list that follows shows the bits in the EID that can be modified. Any attempt to modify any other part of the EID is ignored.

PC_BITS1

X'40' The existence bit for the COMMAREA

X'20' The existence bit for LENGTH

X'10' The existence bit for INPUTMSG

X'08' The existence bit for INPUTMSGLEN

X'04' The existence bit for DATALENGTH

X'02' The existence bit for SYSID

X'01' The existence bit for TRANSID.

PC_BITS2

X'40' The existence bit for CHANNEL.

PC_EIDOPT5

Not used for a PC link request.

Bits in the EID should be modified in place. You should not modify the pointer to the EID. (Any attempt to do so is ignored by CICS.)

The EID is reset to its original value before return to the application program. That is, changes made to the EID are retained for the duration of the program control request only.

Your user exit program is prevented from making major changes to the EID.

Using the program control request token, UEPPCTOK

UEPPCTOK provides the address of a 4-byte area that you can use to pass information between the XPCREQ and XPCREQC user exits for the same program control request.

For example, the address of a piece of storage that is obtained by the XPCREQ user exit, which has to be freed by the XPCREQC user exit, can be passed in the UEPPCTOK field.

Do not use the area addressed by UEPPCTOK to store character data directly. The 4-byte area addressed by UEPPCTOK must always contain the address of a storage area, which contains the data that you want to pass and any eyecatcher to identify the storage. CICS IA uses UEPPCTOK to store the address of its own storage area. When the CICS IA global user exit runs, it checks the address in UEPPCTOK to identify any storage area that already exists for the exit.

UEPPCTOK is usable only for the duration of a single program control request, because its contents might be destroyed at the end of the request. If you need to pass information between successive invocations of a global user exit, you can use task token UEPTSTOK to do so. For more information about UEPTSTOK, see Using the task token UEPTSTOK.

The EIB

Copies of EIBRSRCE, EIBRCODE, EIBRESP, and EIBRESP2 are passed to the exit, so that you can:

- Modify or set completion or resource information in XPCREQ and XPCREQC.
- Examine completion information in XPCREQC.

You can update the copies of EIBRSRCE, EIBRCODE, EIBRESP and EIBRESP2 that you are given in the parameter list. Program Control copies your values into the real EIB after the completion of XPCREQC; or if you specify a return code of 'bypass' in XPCREQ.

You must set valid program control responses. You must set all three of EIBRCODE, EIBRESP and EIBRESP2 to a consistent set of values, such as would be set by Program Control to describe a valid completion. **Program Control does not police the consistency of EIBRCODE, EIBRESP, and EIBRESP2**. To aid you in setting the values of EIBRCODE, EIBRESP, and EIBRESP2, the values used by Program Control are specified in DFHPCEDS.

Example of how XPCREQ and XPCREQC can be used

In this example, XPCREQ and XPCREQC are used to route LINK requests to a number of different CICS regions to provide a simple load balancing mechanism.

The example shows only the capabilities of the exits; it is not intended to indicate an ideal way of achieving the load balancing function. For the purpose of this example, it is assumed that a global work area (GWA) already exists, and that it contains a list of available SYSIDs together with a count of the number of LINK requests currently being processed by each SYSID.

In XPCREQ:

1. Scan the global work area (GWA) to locate a suitable CICS region; for example, the region currently processing the least number of LINK requests.
2. Having decided which system to route the request to, increment the use count for this system.
3. Obtain a 4-byte area in which to store the SYSID for this request (this can be allocated from the GWA to avoid issuing a GETMAIN). If the area is obtained by issuing a GETMAIN, set UEPPCTOK to the address of the storage obtained.
4. Set PC_ADDR7 to the address of the 4-byte area.
5. If setting PC_ADDR7 now makes it the last address, set the high-order bit in the address, and unset the high-order bit in what was previously the last address.
6. Set the X'02' existence bit on in PC_BITS1 to indicate that a SYSID is specified.
7. Return to CICS.

In XPCREQC:

1. Scan the global work area (GWA) and locate the entry for the CICS region specified in the SYSID parameter.
2. Decrement the use count for this system.
3. If a GETMAIN was issued in XPCREQ to obtain an area to hold the SYSID, issue a FREEMAIN for the address held in UEPPCTOK.
4. Return to CICS.

Exit XPCFTCH

XPCFTCH is invoked before a program defined to CICS (including internal CICS modules) receives control, which could be because it is the first program in a transaction, or as a result of a LINK, XCTL, or HANDLE ABEND PROGRAM request.

You can use this exit to modify the entry address used when linking to the program. If the exit sets a return code of zero, or a modified address of zero, the entry address of the original application program is used.

You use this exit to pass control to an AMODE(64), AMODE(31), or AMODE(24) assembler application program or routine before the original program is invoked. After this assembler program finishes its processing, it should pass control back to the entry point of the original program by using a branch instruction. Do not use the exit to invoke any program other than the original program, because the results are unpredictable.

When XPCFTCH is invoked for a C or C++ program that is compiled with the XPLINK option, a flag is set to ignore any modification of the entry point address that the user exit might make.

If a modified entry address is supplied, the program that is invoked receives control in the execution key that the original application program would have received control in; that is, as specified on the EXECKEY option of the resource definition of the original program.

When invoked

Before an application program receives control.

Exit-specific parameters

UEPPCDS

Address of a storage area that contains program- and terminal-related information, and that can be mapped using the DSECT DFHPCUE. When XPCFTCH is invoked, the following DFHPCUE fields are significant:

PCUE_CONTROL_BITS

- 1-byte flag field. A setting of PCUECBTE indicates that the transaction is linked to a terminal.
- A setting of PCUENOTX (X'40') indicates that the program is not command level.
- A flag, PCUE_NO_MODIFY, in PCUE_CONTROL_BITS indicates that a modified entry address is not supported. When set, any return code of UERCMEA from XPCFTCH is ignored. CICS sets this flag before invoking XPCFTCH for C and C++ programs compiled with the XPLINK option.
- A setting of PCUE_REAL (X'20') indicates that a real entry point is set in PCUE_REAL_ENTRY.

PCUE_TASK_NUMBER

3-character packed decimal field that contains the task number.

PCUE_TRANSACTION_ID

4-character field that contains the ID of the original transaction. This ID might differ from the current transaction ID.

PCUE_TERMINAL_ID
4-character field that contains the terminal ID (if any).

PCUE_PROGRAM_NAME
8-character field that contains the name of the program that is to receive control.

PCUE_PROGRAM_LANGUAGE
3-character field that contains the language of the program that is to receive control.

PCUE_LOAD_POINT
The load point of the program.

PCUE_ENTRY_POINT
The entry point of the program.

PCUE_AMOD
The addressing mode of the program is AMODE(31). This field is provided for compatibility with existing exit programs.

PCUE_AMOD_31
The addressing mode of the program is AMODE(31). Use this field in preference to PCUE_AMOD.

PCUE_AMOD_64
The addressing mode of the program is AMODE(64).

PCUE_PROGRAM_SIZE
Fullword that contains the size of the program, in bytes.

PCUE_COMMAREA_ADDRESS
Address of the communication area of the program, if the program has one.

PCUE_COMMAREA_SIZE
Fullword that contains the length of the communication area of the program, if the program has one.

PCUE_LOGICAL_LEVEL
Fullword that contains the program logical level.

PCUE_BRANCH_ADDRESS
Fullword. Use this field to supply an alternative entry address. Set the top bit to specify that the alternative program is to run AMODE (31).

PCUE_REAL_ENTRY
From z/OS 1.7 onwards, this field provides the real entry point for Language Environment® conforming programs. Previously, only PCUE_ENTRY_POINT was available to you, but for Language Environment conforming programs, this field did not contain the entry point that you needed to know about.

Note: With z/OS 1.7, this field provides a solution to the problem raised by APAR PQ43992.

PCUE_CHANNEL_NAME
Address of a 16-byte field that contains the name of the channel with which the application program is to be invoked (that is, the current channel of the program). If there is no channel, this field is set to blanks.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

UERCMEA
Entry address has been modified.

XPI calls

All can be used.

Sample exit program

DFH$PCEX

Exit XPCHAIR

Exit XPCHAIR is invoked before a HANDLE ABEND LABEL routine is given control.

This situation occurs only when a program abend causes a branch to an internal abend routine. When the HANDLE ABEND request specifies PROGRAM, exit XPCFTCH is invoked (see "Exit XPCFTCH" on page 195). You can use the XPCHAIR exit to supply an alternative handle abend address. If the exit sets a return code of zero, or an alternative address of zero, CICS passes control to the specified internal routine of the application program

This exit is never invoked for a C or C++ program that is compiled with the XPLINK option, or for an AMODE(64) program.

If a modified entry address is supplied:

- The code that is invoked receives control in the execution key that the internal abend routine would have received control in; that is, the key in force when the EXEC CICS HANDLE ABEND LABEL command was issued.
- The resume address is placed in register 14 for a COBOL application or register 15 for an Assembler application. If your application runs in a mixed environment, your exit program might need to set up its own base register. For example, you could use the following code to set up addressability:

```
BASSM 15,0
USING *,15
```

When invoked
Before a HANDLE ABEND routine is given control.

Exit-specific parameters

UEPPCDS
Address of a storage area that contains program and terminal related information, and that can be mapped using the DSECT DFHPCUE. When XPCHAIR is invoked, the following DFHPCUE fields are significant:

PCUE_CONTROL_BITS
1-byte flag field. A setting of PCUECBTE indicates that the transaction is linked to a terminal.

PCUE_TASK_NUMBER
3-character packed decimal field that contains the task number.

PCUE_TRANSACTION_ID
4-character field that contains the transaction ID.

PCUE_TERMINAL_ID
4-character field that contains the terminal ID (if any).

PCUE_PROGRAM_NAME
8-character field that contains the name of the program that issued the HANDLE ABEND LABEL command.

PCUE_LOGICAL_LEVEL
Fullword that contains the program logical level.

PCUE_BRANCH_ADDRESS
Fullword. Use this field to supply the address of an alternate abend routine. Set the top bit to specify that the alternate abend routine is to run AMODE (31).

UEPTACB
Address of the transaction abend control block (TACB) for the abend. If the abend occurred because of a program check, the information in the TACB includes:

- The program status word (PSW).
- The registers at the time of the abend.
- Details of the subspace, access registers, and vector registers current at the time of the abend.
- The Breaking Event Address Register (BEAR).
- The Translation Exception Address (TEA)

You can map the TACB using the DFHTACB TYPE=DSECT macro.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

UERCMEA
The address of an alternate abend routine is supplied.

XPI calls
All can be used.

Exit XPCTA

Exit XPCTA is invoked immediately after a transaction abend, and before any processing that might modify the existing environment so that the task could not be resumed.

You can use the XPCTA exit to do the following:

- Set a resume address, instead of letting CICS process the abend
- Specify the subspace that control is passed in

If a resume address is passed back, registers 0 through 13 and 15 are restored to their values at the time of the abend. Register 14 is used to branch to the resume address. If the exit sets a return code of zero, or a resume address of zero, CICS processes the abend.

If the transaction abend occurs as a result of a program check or an operating system abend, the XDUREQ dump domain exit might be invoked before XPCTA (see "Exit XDUREQ" on page 36). Also, if a resume address is passed back, registers 0 through 15 are restored to their value at the time of the abend. The program status word (PSW) is used to branch to the resume address.

In some situations, CICS sets a flag to ignore the resume address that is usually obtained from the UERCMEA return code.

- CICS sets the PCUE_NO_RESUME flag to ignore any resume address that the exit supplies in the following situations:
  - XPCTA is invoked for a C or C++ program that is compiled with the XPLINK option.
  - The task control block (TCB) of the application is no longer available.
  - The transaction abend is an AK*xx* abend (relating to a kill request) other than AKKD or AKKE.
- For an AMODE(64) program, CICS sets the PCUE_NO_RESUME_AMODE64 flag to ignore any resume address that the exit supplies if the 64-bit registers are not available at the time of the abend.

When invoked

After an abend and before the environment is modified.

Exit-specific parameters

UEPPCDS

Address of a storage area that contains program- and terminal-related information, and that can be mapped using the DSECT DFHPCUE. When XPCTA is invoked, the following DFHPCUE fields are significant:

PCUE_CONTROL_BITS

1-byte flag field. A setting of PCUECBTE indicates that the transaction is linked to a terminal.

Flags PCUE_NO_RESUME and PCUE_NO_RESUME_AMODE64 in PCUE_CONTROL_BITS indicate that a resume address is not supported. When set, any return code of UERCMEA from XPCTA is ignored. CICS sets PCUE_NO_RESUME before invoking XPCTA for C and C++ programs compiled with the XPLINK option, when the TCB of the application is no longer available, and for AK*xx* abends other than AKKD or AKKE. CICS sets PCUE_NO_RESUME_AMODE64 for AMODE(64) programs when a resume address is not supported.

PCUE_TASK_NUMBER

3-character packed decimal field that contains the task number.

PCUE_TRANSACTION_ID

4-character field that contains the transaction ID.

PCUE_TERMINAL_ID
4-character field that contains the terminal ID (if any).

PCUE_PROGRAM_NAME
8-character field that contains the name of the failing program.

PCUE_LOGICAL_LEVEL
Fullword that contains the program logical level.

PCUE_BRANCH_ADDRESS
Fullword. You can use this field to supply a resume address. Set the top bit to specify that the resumed task is to run AMODE(31). Set the bottom bit to specify that the resumed task is to run AMODE(64).

PCUE_BRANCH_EXECKEY
If storage protection is active, you can use this 1-byte field to specify the execution key of the resumed task. The possible values are:

PCUE_BRANCH_USER
User key

PCUE_BRANCH_CICS
CICS key.

If storage protection is active and you do not specify a value, the resumed task executes in user key.

If storage protection is not active, the resumed task executes in CICS key.

UEPTACB
Address of the transaction abend control block (TACB) for the abend. If the abend occurred because of a program check, the information in the TACB includes:

- The program status word (PSW).
- The registers at the time of the abend.
- Details of the subspace, access registers and vector registers current at the time of the abend.
- The Breaking Event Address Register (BEAR).
- The Translation Exception Address (TEA).

You can map the TACB using the DFHTACB TYPE=DSECT macro.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

UERCMEA
A resume address is supplied.

XPI calls
All can be used.

Sample XPCTA exit program

DFH$PCTA is a sample global user exit program for the XPCTA exit point. It tests whether the abend was caused by a storage protection exception condition. For more information about DFH$PCTA, see Transaction abend sample exit program: DFH$PCTA.

Exit XPCABND

Exit XPCABND is invoked after a transaction abend and before a transaction dump call. You can use this exit to suppress the dump.

When invoked

After a transaction abend and before a transaction dump call is made.

Exit-specific parameters

UEPPCDS

Address of a storage area that contains program-related and terminal-related information. The storage area is mapped by the DSECT DFHPCUE.

When XPCABND is invoked, the following DFHPCUE fields are significant:

PCUE_CONTROL_BITS

A 1-byte flag field. A setting of PCUECBTE indicates that the transaction is linked to a terminal.

PCUE_TASK_NUMBER

A 3-character packed decimal field that contains the task number.

PCUE_TRANSACTION_ID

A 4-character field that contains the transaction ID.

PCUE_TERMINAL_ID

A 4-character field that contains the terminal ID (if any).

PCUE_PROGRAM_NAME

An 8-character field that contains the name of the program that is abending.

PCUE_LOGICAL_LEVEL

Fullword that contains the program logical level.

UEPTACB

Address of the transaction abend control block (TACB) for the abend. If the abend occurred because of a program check, the information in the TACB includes:

- The program status word (PSW).
- The registers at the time of the abend.
- Details of the subspace, access registers, and vector registers current at the time of the abend.
- The Breaking Event Address Register (BEAR).
- The translation Exception Address (TEA)

You can map the TACB using the DFHTACB TYPE=DSECT macro.

Return codes

UERCNORM
Continue processing and make the dump call.

UERCBYP
Suppress the dump call.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

Resource manager interface program exits (XRMIIN, XRMIOUT)

These exits are invoked when RMI API requests are processed.

Exit XRMIIN

Exit XRMIIN is invoked before a task-related user exit program is invoked when an application program issues an RMI API request.

Exit-specific parameters

UEPTRUEN
Address of the name of the task-related user exit program.

UEPTRUEP
Address of the parameter list to be passed to the task-related user exit program.

UEP_RM_PBTOK
Address of a 4-byte field containing the z/OS Workload Manager (WLM) Performance Block Token. An exit program can use this token to access information (such as the service class token, SERVCLS) in the WLM Performance Block. To do so, it must use the WLM EXTRACT macro, IWMMEXTR, passing the Performance Block Token as the MONTKN input parameter. For more information about the IWMMEXTR macro, see *z/OS MVS Programming: Workload Management Services*.

An exit program must not attempt to modify the Performance Block: if it does so, the results are unpredictable.

UEPRECUR
Address of a halfword recursion counter. The counter is set to 0 when the exit is first invoked, and is incremented for each recursive call.

Note: The task-related user exit program's parameter list is mapped by a DFHUEPAR DSECT that shares common field names with the global user exit program's DFHUEPAR parameter list. To include both DSECT definitions in your exit program, you must code:

```
DFHUEXIT TYPE=EP,ID=XRMIIN
DFHUEXIT TYPE,TYPE=RM
```

The statements must be coded in this order.

The two DFHUEPAR parameter lists, the global user exit's and the task-related user exit's, occupy separate areas of storage. The task-related user exit's parameter list is provided for information only; you should not amend it in any way.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

API and SPI commands
All except EXEC CICS SHUTDOWN and EXEC CICS XCTL can be used. However, CALLDLI, EXEC DLI, or EXEC SQL commands must **not** be used.

Exit XRMIOUT

Exit XRMIOUT is invoked after a task-related user exit program has returned from handling an RMI API request.

Exit-specific parameters

UEPTRUEN
Address of the name of the task-related user exit program.

UEPTRUEP
Address of the parameter list to be passed to the task-related user exit program.

The UEPHMSA parameter in this parameter list contains the register save area (RSA) of the caller. For an EXEC CPSM call, register 1 in this RSA contains the address of the parameter list for the CPSM command and can be used to identify the CPSM command and the specified resource. For details, see CICSPlex SM API command argument list.

UEP_RM_PBTOK
Address of a 4-byte field containing the z/OS Workload Manager (WLM) Performance Block Token. An exit program can use this token to access information (such as the service class token, SERVCLS) in the WLM Performance Block. To do so, it must use the WLM EXTRACT macro, IWMMEXTR, passing the Performance Block Token as the MONTKN input parameter. For more information about the IWMMEXTR macro, see z/OS MVS Programming: Workload Management Services.

An exit program must not attempt to modify the Performance Block: if it does so, the results are unpredictable.

UEPRECUR
Address of a halfword recursion counter. The counter is set to 0 when the exit is first invoked, and is incremented for each recursive call.

Note: The task-related user exit program's parameter list is mapped by a DFHUEPAR DSECT that shares common field names with the global user

exit program's DFHUEPAR parameter list. To include both DSECT definitions in your exit program, you must code:

```
DFHUEXIT TYPE=EP,ID=XRMIOUT
DFHUEXIT TYPE,TYPE=RM
```

The statements must be coded in this order.

The DFHUEPAR parameter list of the global user exit and the DFHUEPAR parameter list of the task-related user exit each occupy separate areas of storage. The parameter list of the task-related user exit is provided for information only; do not amend it in any way.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

API and SPI commands
All except EXEC CICS SHUTDOWN and EXEC CICS XCTL can be used. However, CALLDLI, EXEC DLI, or EXEC SQL commands must **not** be used.

Note: It is not advisable for your exit program to make calls to other external resource managers that use the RMI, because this causes recursion, and might result in a loop. It is the responsibility of your exit program to avoid entering a loop. You can use the recursion counter field UEPRECUR in the exit program to guard against this possibility.

Resource management installation and discard exit XRSINDI

The XRSINDI global user exit is driven, if it is enabled, immediately after CICS successfully installs or discards a resource definition.

The installation and discard activities that drive the exit are as follows:

- The installation function of the group list on an initial or cold start of CICS
- The **CEDA INSTALL** command
- The CICSPlex SM BAS **INSTALL** command
- All autoinstall operations, as follows:
  - The autoinstall of a terminal, connection, program, map set, partition set, or journal
  - The automatic discard of an unused terminal, controlled by the AILDELAY system initialization parameter and the SIGNOFF attribute on the TYPETERM resource definition.
- The connection to, and disconnection from, an MVS log stream
- A **CEMT DISCARD** and **EXEC CICS DISCARD** command
- An **EXEC CICS CREATE** command
- The front-end programming interface (FEPI) installation and discard operations: the **EXEC CICS FEPI INSTALL** command and **EXEC CICS FEPI DISCARD** command.

The parameter list is designed to pass the names of more than one resource, installed or discarded, in field UEPIDNAM. When designing your global user exit program, do not assume that the number of resource names passed is always one. Analyze the resources in a loop based on the value referenced by UEPIDNUM.

The names of modegroups are prefixed with the corresponding connection name. There is no separator between the two names: the first 4 characters form the connection name, followed by 8 characters for the modegroup. The parts of the concatenated name are fixed length—if connection names are defined with less than 4 characters, they are padded with blanks in the concatenated names. Similarly, the connection names for a front-end programming interface (FEPI) connection is a concatenation of a FEPI node name and a FEPI target name, each of which is 8 characters long (fixed length) with no separator.

The exit is driven once for each individual resource in a group list installed during a CICS initial or cold start. If you are concerned about the performance on an initial or cold start, do not enable the exit until after the group list is installed. To obtain the information about resources installed before enabling the exit, you can write a program to scan the tables of installed resources, by using the **EXEC CICS INQUIRE** *resource_name* browse function.

Exit XRSINDI

The XRSINDI global user exit is called when CICS installs or discards a resource definition.

Exit XRSINDI parameters and return codes

Abends in a program that is enabled at the XRSINDI exit point might cause CICS to shut down, because for some resources the exit is driven during sync point. If the exit returns code UERCPURG during the sync point for these resources, abend code AUEP is produced and CICS shuts down.

The parameters, return codes, and XPI information are as follows:

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID.

UEPUSER
Address of the 8-byte user ID.

UEPTERM
Address of the 4-byte terminal ID.

UEPPROG
Address of the 8-byte application program name.

UEPIDREQ
Address of the 1-byte install or discard identifier. The values are as follows:

UEIDINS
This request is for an install action, or, in the case of a log stream, it is a connection to a log stream.

UEIDDIS
This request is for a discard action, or, in the case of a log stream, it is a disconnection from a log stream.

UEPIDTYP

Address of the 1-byte type of resource. The values are as follows:

UEIDATOM
An ATOMSERVICE resource.

UEIDAITM
An autoinstall terminal model.

UEIDBNDL
A BUNDLE resource.

UEIDCONN
A connection.

UEIDDB2C
A DB2CONN resource definition for the connection between CICS and DB2®.

UEIDDB2E
A DB2ENTRY resource definition.

UEIDDB2T
A DB2TRAN resource definition.

UEIDDOCT
A DOCTEMPLATE.

UEIDEARB
An EAR file that is part of a CICS bundle.

UEIDEBAB
An EBA file that is part of a CICS bundle.

UEIDEPAD
An EPADAPTER resource.

UEIDEPAS
An EPADAPTERSET resource.

UEIDEVCS
An event capture resource.

UEIDEVNT
An EVENTBINDING resource.

UEIDFECO
A FEPI connection.

UEIDFENO
A FEPI node.

UEIDFEPO
A FEPI pool.

UEIDFEPS
A FEPI property set.

UEIDFETA
A FEPI target.

UEIDFILE
A file.

UEIDIPCO
An IPCONN resource.

UEIDJNMD
A journal model.

UEIDJNNM
A journal name.

UEIDJSRV
A JVM server resource.

UEIDLBRY
A LIBRARY resource.

UEIDMAP
A mapset.

UEIDMODE
A modegroup.

UEIDMPPP
A Policy resource.

UEIDMQCN
An MQCONN resource definition for the connection between CICS and WebSphere® MQ.

UEIDMQIN
An MQINI resource.

UEIDNQRN
An ENQMODEL.

UEIDOSGB
An OSGi bundle.

UEIDPART
A partner.

UEIDPIPE
A pipeline (PIPELINE).

UEIDPKST
A DB2 PACKAGESET resource.

UEIDPROF
A profile.

UEIDPROG
A program.

UEIDPRTY
A BTS process type.

UEIDPSET
A partition set.

UEIDSESS
A session.

UEIDSTRM
An MVS log stream.

UEIDTCLS
A transaction class.

UEIDTCPS
A TCP/IP service.

UEIDTDQU
A transient data queue.

UEIDTERM
A terminal.

UEIDTRAN
A transaction.

UEIDTSMD
A temporary storage queue model.

UEIDURIM
A URIMAP resource.

UEIDWARB
A WAR file that is part of a CICS bundle.

UEIDWEBS
A web service (WEBSERVICE).

UEIDXMLT
An XMLTRANSFORM resource.

UEPIDLEN
For public resources, this parameter is the address of the length of an individual resource name, as a fullword binary value.

For OSGi bundles, this parameter is the address of the length of the information that uniquely identifies an OSGi bundle in CICS in the **UEPIDNAM** parameter. The maximum length is 526 bytes.

UEPIDNUM
Address of the number of resources that are reported by this call, as a fullword binary value.

UEPIDNAM
Address of a variable-length list containing the names of the individual resources reported by this call.

For OSGi bundles, this parameter contains the information that uniquely identifies an OSGi bundle in CICS. The information is listed in the following order:

1. 8 bytes that contain the JVM server name.
2. A fullword containing the length of the OSGi bundle symbolic name.
3. A fullword containing the length of the OSGi bundle version.
4. A concatenation of the OSGi bundle symbolic name and version as a character string.

UEPIDREC
Address of a 1-byte identifier indicating whether resources are recovered at a warm or emergency restart. The values are as follows:

UEIDKEEP
The resources are recoverable at a warm or emergency restart.

UEIDLOSE
The resources are not recoverable.

Note: The exit is not driven during a CICS restart.

UEPDEFTM
The address of a variable-length list, which corresponds to the list in UEPIDNAM, containing the definition time of the individual resource as an 8-character STCK value.

Note: The parameters UEPDEFTM, UEPCHUSR, UEPCHAGT, UEPCHREL, UEPCHTIM, UEPDEFSRC, UEPINUSR, UEPINTIM, and UEPINAGT are valid for the following resources: ATOMSERVICE, BUNDLE, CONNECTION, DB2CONN, DB2ENTRY, DB2TRAN, DOCTEMPLATE, ENQMODEL, EPADAPTER, EPADAPTERSET, EVENTBINDING, FILE, IPCONN, JOURNALMODEL, JVMSERVER, LIBRARY, MQCONN, MQINI, OSGIBUNDLE, PIPELINE, PROFILE, PROCESSTYPE, PROGRAM, TCPIPSERVICE, TDQUEUE, TRANCLASS, TRANSACTION, TSMODEL, URIMAP, WEBSERVICE, and XMLTRANSFORM. The parameter value is zero for all other resources.

UEPCHUSR
Address of a variable-length list, which corresponds to the list in UEPIDNAM, containing the 8-character user ID that ran the agent that last changed the individual resource.

UEPCHAGT
Address of a variable-length list, which corresponds to the list in UEPIDNAM, of a 2-byte identifier representing the agent that last changed the individual resource. The possible values are as follows:

UEPUNKAGT
The resource was changed by an unknown agent.

UEPCSDAPI
The resource was changed using the CSD API or CEDA.

UEPCSDBAT
The resource was changed using the CSD batch program, DFHCSDUP.

UEPDRPAPI
The resource was changed using the CICSPlex SM BAS API.

UEPAUTOIN
The resource was changed using autoinstall.

UEPSYSTEM
The resource was changed by the running CICS region.

UEPDYNAMC
The resource was changed dynamically.

UEPTABLE
The resource was changed using a table.

UEPCHREL
Address of a variable-length list, which corresponds to the list in UEPIDNAM, containing the 4-character CICS release level that was running when the individual resource was last changed.

UEPCHTIM
Address of a variable-length list, which corresponds to the list in UEPIDNAM, containing the CSD record time stamp change for the individual resource as an 8-character STCK value.

UEPDEFSRC
Address of a variable-length list, which corresponds to the list in UEPIDNAM, containing the 8-character CSD group name or source corresponding to the individual resource.

UEPINUSR
Address of a variable-length list, which corresponds to the list in UEPIDNAM, containing the 8-character user ID that installed the individual resource.

UEPINTIM
Address of a variable-length list, which corresponds to the list in UEPIDNAM, containing the time that the domain was called for the installation of the individual resource as an 8-character STCK value.

UEPINAGT
Address of a variable-length list, which corresponds to the list in UEPIDNAM, of a 2-byte identifier representing the agent that installed the individual resource. The possible values are as follows:

UEPCSDAPI
The resource was installed using the CSD API or CEDA.

UEPCRESPI
The resource was installed using the EXEC CICS CREATE SPI commands.

UEPGRPLST
The resource was installed at startup using GRPLIST install.

UEPAUTOIN
The resource was autoinstalled.

UEPSYSTEM
The resource was installed by the running CICS system.

UEPDYNAMC
The resource was installed dynamically.

UEPBUNDLE
The resource was installed by a bundle deployment.

UEPTABLE
The resource was installed using a table.

UEPAPPTK
Address of a variable-length list, containing an 8-character token representing the application instance to which this resource belongs. For public resources, this address is zero.

UEPAPCTXT
For private resources for applications that are deployed on platforms, this parameter contains the address of a variable-length

list, which corresponds to the list in UEPIDNAM, containing the application context information for the resource. The information is listed in the following order:

1. The platform name, padded with spaces to 64 characters.
2. The application name, padded with spaces to 64 characters.
3. The major version number for the application, which is a fullword binary value.
4. The minor version number for the application, which is a fullword binary value.
5. The micro version number for the application, which is a fullword binary value.

CICS supplies a DSECT named **DFHUEACD** which maps this information. For more information about **DFHUEACD**, see UEACD - User exit application context in Data Areas.

UEPPLATTK
Address of a variable-length list, containing an 8-character token representing the platform instance to which this resource belongs. For public resources, this address is zero.

Return codes

UERCNORM
Continue processing. This code is the default.

UERCPURG
Task purged during XPI call.

XPI calls
You can use all XPI calls.

Signon and signoff exits XSNON, XSNOFF, and XSNEX

Exit XSNON is invoked after a terminal user signs on, and exit XSNOFF is invoked after a terminal user signs off (whether the signon or sign-off is successful or not). XSNON and XSNOFF do not make any security decisions; they are merely a means of tracking users logging on and off a CICS system.

The activities which drive the exits are:

- Invocation of an EXEC CICS SIGNON command for a terminal (when, for example, the terminal user enters the CICS-supplied CESN, or an equivalent, user-written, signon transaction)
- Invocation of an EXEC CICS SIGNON command for a surrogate terminal (that is, a terminal attached by the CRTE routing transaction, or by dynamic transaction routing)
- Invocation of an EXEC CICS SIGNOFF command for a terminal
- When a 'CANCEL' command is entered to terminate a CRTE routing session
- A timeout sign-off.

XSNEX is a special-purpose global user point, which is intended to be used only with the IBM-supplied global user exit program, DFH$SNEX.

Exit XSNON

When invoked
When a user signs on.

Exit-specific parameters

UEPUSRID
Address of the terminal userid.

UEPUSRLN
Address of the terminal userid length.

UEPGRPID
Address of the group ID. If the signon was successful, the group ID is that which the user is associated with in this signon session. If the signon was unsuccessful, it is that specified by the user when he or she tried to sign on.

UEPGRPLN
Address of the group ID length.

UEPNETN
Address of the terminal's netname.

UEPTRMID
Address of the terminal id.

UEPTCTUA
Address of the TCT user area.

UEPTCTUL
Address of the TCT user area length.

UEPTRMTY
Address of the terminal-type byte.

UEPSNFLG
Address of a 2-byte field containing flags:

Table 1. Flags set in the UEPSNFLG field of XSNON

Flag	Equivalent	Meaning
UEPSNOK	0	Signon was successful.
UEPSNFL	1	Signon failed.
UEPSNPSS	2	The persistent sessions signon succeeded.
UEPSNPSF	3	The persistent sessions signon failed.

UEPSGTYP
Address of signon type byte.

This parameter has two equates:

```
UEPSGUID   SIGNON USERID
UEPSGKER   SIGNON KERBEROS
```

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

Exit XSNOFF

When invoked
When a user signs off.

Exit-specific parameters

UEPUSRID
Address of the terminal userid.

UEPUSRLN
Address of the terminal userid length.

UEPGRPID
Address of the group ID.

UEPGRPLN
Address of the group ID length.

UEPNETN
Address of the terminal's netname.

UEPTRMID
Address of the terminal id.

UEPTCTUA
Address of the TCT user area.

UEPTCTUL
Address of the TCT user area length.

UEPTRMTY
Address of the terminal-type byte.

UEPSNFLG
Address of a 2-byte field containing flags:

UEPSNOK
Sign-off was successful

UEPSNFL
Sign-off failed

UEPSNNML
Normal sign-off

UEPSNTIM
Timeout sign-off.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

Exit XSNEX

The purpose of XSNEX, in conjunction with its supporting sample programs, is to provide a short-term aid for upgrading. It is designed to give you time to modify

those application programs that have a dependency on the way CICS handles **EXEC CICS SIGNON** and SIGNOFF before CICS TS 2.1, to enable them to work with the current behavior.

Note: XSNEX is for upgrade purposes only. Remove all application dependency on the old sign-on and sign-off behavior.

There are no exit-specific parameters for this global user exit, which is invoked whenever an application program issues an **EXEC CICS SIGNON** or an **EXEC CICS SIGNOFF** command. You are not intended to write your own global user exit program for this exit point. IBM® provides DFH$SNEX, the sole purpose of which is to make CICS handle **EXEC CICS SIGNON** and SIGNOFF commands in the same way as in CICS TS 1.3 and earlier.

The supplied programs are:

DFH$SNEX
: This user exit program is supplied in SDFHSAMP. The only function the program performs is to set return code UERCPREV, which causes the security domain to restore CICS behavior as in CICS TS 1.3 and earlier. You can enable this user exit program using DFH$SNPI.

DFH$SNPI
: This post-initialization program is supplied in SDFHSAMP. It issues an EXEC CICS ENABLE PROGRAM('DFH$SNEX') EXIT('XSNEX') command to enable the IBM-supplied user exit program, DFH$SNEX, in the final stages of CICS initialization.

  To use this program, add an entry to the first section of your PLTPI table (that is, before the DFHDELIM statement). For example:

```
DFHPLT TYPE=INITIAL,SUFFIX=SN
DFHPLT TYPE=ENTRY,PROGRAM=DFH$SNPI
DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
DFHPLT TYPE=FINAL
END
```

Statistics domain exit XSTOUT

On invocation, XSTOUT is passed the address of a buffer containing one or more statistics records. The buffer can contain records for various resource types; for example, connections and modenames. The buffer can also contain both specific and global information; for example, loader statistics for individual programs, and loader statistics for all programs.

Your exit program can identify the types of records in the buffer by their STID values. (STID values are described in CICS statistics data section.)

You can use XSTOUT to prevent the contents of the statistics data buffer being written to SMF. Note that you cannot use it to selectively suppress individual records within the buffer. Your exit program should not modify the values of any of the exit-specific parameters.

Some statistics records might be produced during very early during CICS initialization which will not be passed to XSTOUT. The earliest that a global user exit can be enabled is during PLT processing. Before this no exits can be invoked.

Exit XSTOUT

When invoked

Before a statistics record is written to SMF.

Exit-specific parameters

Fields UEPPROG, UEPTERM, UEPTRANID, and UEPUSER have meaning only for requested statistics (when using the CICS Explorer® **Regions** operations view, the **CEMT PERFORM STATISTICS RECORD** command, or the **EXEC CICS PERFORM STATISTICS RECORD** command).

UEPPROG

Address of the 8-byte application program name.

UEPSCLD

Address of an 8-byte character field containing the collection date (MMDDYYYY).

UEPSDATE

Address of a 6-byte character field containing the collection date (MMDDYY).

UEPSIVAL

Address of a 6-byte character field containing the interval time (HHMMSS). This field has meaning only for interval statistics.

UEPSIVN

Address of the 4-byte interval number. This field has meaning only for interval statistics.

UEPSRLEN

Address of the 4-byte hexadecimal length of the statistics record.

UEPSTATS

Address of a buffer containing one or more statistics records. For unsolicited statistics, the buffer always contains one record; for other types of statistics, it might contain several records. The length of the buffer is addressed by the UEPSRLEN parameter.

UEPSTIME

Address of a 6-byte character field containing the collection time (HHMMSS).

UEPSTYPE

Address of the 3-byte character field statistics type. The values of the types are as follows:

INT Interval statistics.

EOD End-of-day statistics.

REQ Requested statistics.

RRT Requested reset statistics.

USS Unsolicited statistics.

UEPTERM

Address of the 4-byte terminal ID.

UEPTRANID

Address of the 4-byte transaction ID.

UEPUSER

Address of the 8-byte user ID.

Return codes

UERCBYP
Suppress output of statistics data buffer to SMF.

UERCNORM
Continue processing.

XPI calls
WAIT_MVS can be used. Note, however, that the wait cannot be purged by using CEMT or SPI commands. Do not use any other calls.

System recovery program exit XSRAB

Exit XSRAB is invoked when the system recovery program (DFHSRP) finds a match in the system recovery table (SRT) for an operating system abend code.

When invoked
Exit XSRAB is invoked when the system recovery program (DFHSRP) finds a match in the SRT for an operating system abend code. For information about defining entries in the SRT, see System recovery table (SRT).

The SRT table is processed and the exit is driven, only when an MVS abend occurs under a CICS essential TCB; that is, one of QR, RO, CO, SZ, RP, or FO. For nonessential TCB types, such as L8, SL, SO, or S8, the exit is not driven.

Exit-specific parameters

UEPERROR
Address of the error data structure, SRP_ERROR_DATA, which contains the following fields:

SRP_ERROR_TYPE
The 4-character error type, which is always ASRB.

SRP_SYS_ABCODE
2 bytes that contain the system abend code XXX in binary format (for example, D37).

SRP_USER_ABCODE
2 bytes that contain the user abend code NNNN in binary format (for example, 0999).

SRP_ERROR_TRANID
4-character field that contains the ID of the transaction that abended.

SRP_ERROR_STACK_NAME
8-character field that contains the name of the current kernel stack entry for the transaction at the time of the abend.

SRP_ERROR_PPT_NAME
8-character field that contains the name of the current program for the transaction. This field contains a value only if flag SRP_PPT_ENTRY is set.

SRP_ERROR_OFFSET
Fullword that contains the offset into the program that abended, as follows:

- If flag SRP_PPT_ENTRY is set, gives the offset in SRP_ERROR_PPT_NAME

- Otherwise, gives the offset in SRP_ERROR_STACK_NAME.

This field contains a value only if flag SRP_VALID_OFFSET is set.

SRP_ERROR_FLAGS
1 byte that contains flags:

SRP_CICS_CODE
The abend occurred while running CICS code.

SRP_USER_CODE
The abend occurred while running user application code.

SRP_PPT_ENTRY
The abend occurred while running SRP_ERROR_PPT_NAME. If this flag is not set, the abend occurred while running SRP_ERROR_STACK_NAME.

SRP_VALID_OFFSET
A meaningful offset could be determined.

SRP_VALID_REASON
MVS has supplied a reason code for the abend.

SRP_NOT_CICS_RB
CICS RB was not in control at the time of the abend (that is, the abend occurred in a system service invoked by CICS).

SRP_CICS_ERROR_REASON
4-character field that contains the MVS abend reason code. It contains a value only if flag SRP_VALID_REASON is set.

SRP_CICS_ERROR_DATA
An area that describes the last thing that CICS did, before the abend. It contains the following:

SRP_CICS_EC_PSW
8-character field that contains the extended control (EC) mode program status word (PSW)

SRP_CICS_PSW16
16-character field that contains the 128-bit PSW

SRP_CICS_EC_INT
8-character field that contains the interrupt code and ILC

SRP_CICS_REGST
64-character field that contains the contents of the general-purpose (GP) registers

SRP_CICS_EXEC_KEY
1 byte that contains the PSW key, in the form X'0n'.

SRP_SYSTEM_ERROR_DATA
An area that describes the last thing the system did, before the abend. It contains the following:

SRP_SYSTEM_EC_PSW
8-character field that contains the EC mode PSW

SRP_SYSTEM_PSW16
16-character field that contains the 128-bit PSW

SRP_SYSTEM_EC_INT
8-character field that contains the interrupt code and ILC

SRP_SYSTEM_REGST
64-character field that contains the contents of the GP registers

SRP_SYSTEM_EXEC_KEY
1 byte that contains the PSW key, in the form X'0n'.

SRP_ERROR_FP_REGS
An area that describes the contents of the floating point registers at the time of the abend. It contains:

SRP_FP_REG_0
FP register 0

SRP_FP_REG_2
FP register 2

SRP_FP_REG_4
FP register 4

SRP_FP_REG_6
FP register 6

SRP_ADDITIONAL_REG_INFO
An area that contains additional register information.

SRP_ADDITIONAL_REGS_FLAG
1 byte that contains flags:

SRP_CICS_GPR64_AVAIL
The 64-bit CICS GP registers are available.

SRP_SYSTEM_GPR64_AVAIL
The 64-bit system GP registers are available.

SRP_ADDITIONAL_FPR_AVAIL
Additional FP registers are available.

SRP_CICS_GP64_REGS
128-byte area that contains the CICS 64-bit GP registers at the time of the abend.

SRP_SYSTEM_GP64_REGS
128-byte area that contains the system 64-bit GP registers at the time of the abend.

SRP_ADDITIONAL_FPR_REGS
132-byte area that contains additional FP registers at the time of the abend.

SRP_FP_REGS
128-byte area that contains all the FP registers at the time of the abend.

SRP_FPC_REG
4-byte field that contains the FPC register at the time of the abend.

SRP_VR_REGS
512-byte area that contains all the Vector registers at the time of the abend.

If flag SRP_NOT_CICS_RB is set, SRP_CICS_ERROR_DATA describes the last thing that CICS did before the abend and SRP_SYSTEM_ERROR_DATA describes the last thing that the system service (for example, z/OS Communications Server, VSAM, or MVS) did.

You can map the SRP_ERROR_DATA that is passed to the XSRAB exit by using the DFHSRED TYPE=DSECT macro. The format of SRP_ERROR_DATA is shown in SRED - System recovery error data.

Return codes

UERCNOCA
Abnormally terminate the task with abend code ASRB. Do not cancel any program-level abend exits that are associated with this task.

UERCCANC
Abnormally terminate the task with abend code ASRB. Cancel any program-level abend exits that are associated with this task.

UERCCICS
Abnormally terminate CICS.

XPI calls

Because CICS invokes the exit XSRAB in an error environment, you can use only a subset of the XPI calls.

Only TRACE_PUT is available for general use.

You can use WAIT_MVS, but only after the exit program determines (from the SRP_CICS_CODE and SRP_USER_CODE fields) that the abend occurred in user application code, and not in CICS code.

Important:

- Take care when coding a program to run at the XSRAB exit point. If your exit program causes the system recovery program to be reentered (for example, if a program check occurs), CICS terminates abnormally with a DFHSR06*xx* message.
- The default return code is UERCNOCA, which ensures that the task abends if the exit is in error.
- There is no UERCNORM return code at this exit point, because the exit is invoked after a failure.
- The exit should not set the return code UERCPURG.

System termination program exit XSTERM

The XSTERM exit could be used to output final statistics to your statistics SMF data sets, and to close them. Note that CICS VSAM and BDAM data sets have already been closed by CICS file control before the exit is invoked.

When invoked

During the second quiesce stage of a normal system shutdown,

immediately before the transient data and temporary storage buffers are cleared. The exit is not invoked during an IMMEDIATE shutdown.

Exit-specific parameters

None.

Return codes

UERCNORM
Continue processing.

XPI calls

All other XPI calls except WRITE_JOURNAL_DATA can be used. However, their use is not recommended, because they could cause the task to lose control, thus allowing another task to write more monitoring data.

Temporary storage domain exits (XTSQRIN, XTSQROUT, XTSPTIN, XTSPTOUT)

These exits are invoked when temporary storage has to be controlled or monitored.

You can change the temporary storage domain exits XTSQRIN, XTSQROUT, XTSPTIN, and XTSPTOUT to perform the following tasks:

- Specify, for a request that creates a queue, whether the queue is to be held in main or auxiliary storage, and its recoverability
- Monitor the use of temporary storage
- Control security for temporary storage queues

The UEPTERM parameter is a zero value for temporary storage requests that have been function shipped over an IPIC connection. To use IPIC connections for temporary storage requests, ensure that XTSQRIN, XTSQROUT, XTSPTIN, and XTSPTOUT check that the UEPTERM parameter is a non-zero value before trying to use it as an address.

XTSQRIN, XTSQROUT, XTSPTIN, and XTSPTOUT must be coded to threadsafe standards and declared threadsafe to get the benefits of being threadsafe using an IPIC connection.

The temporary storage domain has two main gates, TSQR and TSPT, that support the following functions:

TSQR Write, Rewrite, Read_into, Read_set, Read_next_into, Read_next_set, Delete

TSPT Put, Put_replace, Get, Get_set, Get_release, Get_release_set, Release

The TSQR functions correspond to those available through the EXEC CICS interface (or through DFHTS TYPE=PUTQ, GETQ, or PURGE). The TSPT functions are used by the interval control program in support of START and RETRIEVE functions (or DFHTS TYPE=PUT, GET, or RELEASE).

Exit XTSQRIN

Exit XTSQRIN is invoked before execution of a user temporary storage interface request for a user TS queue (for example, a WRITEQ TS, or READQ TS request).

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID.

UEPUSER
Address of the 8-byte user ID.

UEPTERM
Zero, or the address of the 4-byte terminal ID. If no address is returned, this could mean that this request has been function shipped over an IPIC connection.

UEPPROG
Address of the 8-byte application program name.

UEP_TS_FUNCTION
Address of a byte containing the function:

- UEP_TS_FUN_WRITE
- UEP_TS_FUN_REWRITE
- UEP_TS_FUN_READ_INTO
- UEP_TS_FUN_READ_SET
- UEP_TS_FUN_READ_NEXT_INTO
- UEP_TS_FUN_READ_NEXT_SET
- UEP_TS_FUN_DELETE

UEP_TS_QUEUE_NAME
Address of a 16-byte field containing the queue name.

UEP_TS_DATA_P
Address of a fullword containing the address of the data. (Write and rewrite requests).

UEP_TS_DATA_L
Address of a fullword containing the length of the data. (Write and rewrite requests).

UEP_TS_ITEM_NUMBER
Address of a fullword containing the item number. (Rewrite, read_into and read_set requests).

UEP_TS_STORAGE_TYPE
Address of a byte containing the storage type. (Write requests).

On input to the exit, the parameter will be set to either UEP_TS_STORAGE_TYPE_MAIN or UEP_TS_STORAGE_TYPE_AUX_TST. This parameter may be modified by the exit to any of the following values.

Note that if CICS® has been initialized with TS main-only support, setting this parameter has no effect. See the description of the TS system initialization parameter for more information.

UEP_TS_STORAGE_TYPE_MAIN
Main storage.

UEP_TS_STORAGE_TYPE_AUX_TST
Auxiliary storage (recoverability determined by the resource definition).

UEP_TS_STORAGE_TYPE_AUX_RECOV_YES
Auxiliary storage (recoverable).

UEP_TS_STORAGE_TYPE_AUX_RECOV_NO
Auxiliary storage (non-recoverable).

Return codes

UERCNORM
Normal.

UERCPURG
Purged.

XPI calls
All can be used.

API and SPI calls
None can be used.

Exit XTSQROUT

Exit XTSQROUT is invoked after execution of a user temporary storage interface request for a user TS queue (for example, a WRITEQ TS, or READQ TS request).

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID.

UEPUSER
Address of the 8-byte user ID.

UEPTERM
Zero, or the address of the 4-byte terminal ID. If no address is returned, this could mean that this request has been function shipped over an IPIC connection.

UEPPROG
Address of the 8-byte application program name.

UEP_TS_FUNCTION
Address of a byte containing the function:

- UEP_TS_FUN_WRITE
- UEP_TS_FUN_REWRITE
- UEP_TS_FUN_READ_INTO
- UEP_TS_FUN_READ_SET
- UEP_TS_FUN_READ_NEXT_INTO
- UEP_TS_FUN_READ_NEXT_SET
- UEP_TS_FUN_DELETE

UEP_TS_QUEUE_NAME
Address of a 16-byte field containing the queue name.

UEP_TS_DATA_P
Address of a fullword containing the address of the data. (All requests except delete).

UEP_TS_DATA_L
Address of a fullword containing the length of the data. (All requests except delete).

UEP_TS_ITEM_NUMBER
Address of a fullword containing the item number. (Rewrite, read_into and read_set requests).

UEP_TS_TOTAL_ITEMS
Address of a fullword containing the total number of items in the queue. (All requests except delete).

UEP_TS_RESPONSE
Address of a byte containing the response after a request has been completed.

- UEP_TS_RESPONSE_OK
- UEP_TS_RESPONSE_PURGED
- UEP_TS_RESPONSE_EXCEPTION
- UEP_TS_RESPONSE_DISASTER
- UEP_TS_RESPONSE_INVALID

Return codes

UERCNORM
Normal response.

UERCPURG
A purged response was received from an XPI request.

XPI calls
All can be used.

API and SPI calls
None can be used.

Exit XTSPTIN

Exit XTSPTIN is invoked before execution of a temporary storage interface request for a CICS internal queue (for example, for interval control or BMS queues).

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID.

UEPUSER
Address of the 8-byte user ID.

UEPTERM
Zero, or the address of the 4-byte terminal ID. If no address is returned, this could mean that this request has been function shipped over an IPIC connection.

UEPPROG
Address of the 8-byte application program name.

UEP_TS_FUNCTION
Address of a byte containing the function:

- UEP_TS_FUN_PUT
- UEP_TS_FUN_PUT_REPLACE
- UEP_TS_FUN_GET
- UEP_TS_FUN_GET_SET
- UEP_TS_FUN_GET_RELEASE
- UEP_TS_FUN_GET_RELEASE_SET
- UEP_TS_FUN_RELEASE

UEP_TS_QUEUE_NAME
Address of a 16-byte field containing the queue name.

UEP_TS_DATA_P
Address of a fullword containing the address of the data. (Put and put_replace).

UEP_TS_DATA_L
Address of a fullword containing the length of the data. (Put and put_replace).

UEP_TS_STORAGE_TYPE
Address of a byte containing the storage type. (Put requests).

On input to the exit, the parameter will be set to either UEP_TS_STORAGE_TYPE_MAIN or UEP_TS_STORAGE_TYPE_AUX_TST. This parameter may be modified by the exit to any of the following values.

Note that if CICS has been initialized with TS main-only support, setting this parameter has no effect.

UEP_TS_STORAGE_TYPE_MAIN
Main storage.

UEP_TS_STORAGE_TYPE_AUX_TST
Auxiliary storage (recoverability determined by the resource definition).

UEP_TS_STORAGE_TYPE_AUX_RECOV_YES
Auxiliary storage (recoverable).

UEP_TS_STORAGE_TYPE_AUX_RECOV_NO
Auxiliary storage (non-recoverable).

Return codes

UERCNORM
Normal.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

API and SPI calls
None can be used.

Exit XTSPTOUT

Exit XTSPTOUT is invoked after execution of a temporary storage interface request for a CICS internal queue (for example, for interval control or BMS queues). After execution of a TSPT request. No parameters may be modified.

Exit-specific parameters

UEPTRANID
Address of the 4-byte transaction ID.

UEPUSER
Address of the 8-byte user ID.

UEPTERM
Zero, or the address of the 4-byte terminal ID. If no address is returned, this could mean that this request has been function shipped over an IPIC connection.

UEPPROG
Address of the 8-byte application program name.

UEP_TS_FUNCTION
Address of a byte containing the function:
- UEP_TS_FUNCTION_PUT
- UEP_TS_FUN_PUT_REPLACE
- UEP_TS_FUN_GET
- UEP_TS_FUN_GET_SET
- UEP_TS_FUN_GET_RELEASE
- UEP_TS_FUN_GET_RELEASE_SET
- UEP_TS_FUN_RELEASE

UEP_TS_QUEUE_NAME
Address of a 16-byte field containing the queue name.

UEP_TS_DATA_P
Address of a fullword containing the address of the data. (All requests except release).

UEP_TS_DATA_L
Address of a fullword containing the length of the data. (All requests except release).

UEP_TS_RESPONSE
Address of a byte containing the response after a request has been completed.
- UEP_TS_RESPONSE_OK
- UEP_TS_RESPONSE_PURGED
- UEP_TS_RESPONSE_EXCEPTION
- UEP_TS_RESPONSE_DISASTER
- UEP_TS_RESPONSE_INVALID

Return codes

UERCNORM
Normal response.

UERCPURG
A purged response was received from an XPI request.

XPI calls
All can be used.

API and SPI calls
None can be used.

Temporary storage EXEC interface program exits XTSEREQ and XTSEREQC

The XTSEREQ exit allows you to intercept temporary storage API requests before any action has been taken on the request. The XTSEREQC exit allows you to intercept the response after a temporary storage API request has completed.

The API requests affected are:
- **EXEC CICS WRITEQ TS**
- **EXEC CICS READQ TS**
- **EXEC CICS DELETEQ TS.**

Using XTSEREQ, you can:

- Analyze the API parameter list (function, keywords, argument values, and responses)
- Modify any input parameter value before execution of a request
- Prevent execution of a request.

Using XTSEREQC, you can:

- Analyze the API parameter list
- Modify any output parameter value after request completion.

You can also:

- Pass data between your XTSEREQ and XTSEREQC exit programs when they are invoked for the same request
- Pass data between your temporary storage exit programs when they are invoked within the same task.

It is possible that programs invoked from the exits in the temporary storage domain (XTSQRIN, XTSQROUT, XTSPTIN, and XTSPTOUT) could modify situations set up by XTSEREQ; therefore you must consider the order in which the exits are invoked.

If all the temporary storage exits are enabled, the order of invocation is as follows:

1. XTSEREQ
2. XTSQRIN
3. XTSQROUT
4. XTSEREQC

Exit XTSEREQ

The XTSEREQ exit allows you to intercept temporary storage API requests before any action has been taken on the request.

When invoked

Before CICS processes a temporary storage API request.

Exit-specific parameters

UEPCLPS

Address of a copy of the command parameter list. See "The command-level parameter structure" on page 229.

UEPTQTOK

Address of a 4-byte area which can be used to pass information between XTSEREQ and XTSEREQC for a single temporary storage request.

UEPRCODE

Address of a 6-byte hexadecimal copy of the EIB return code EIBRCODE. For details of EIB return codes, see EIB fields.

UEPRESP

Address of a 4-byte binary copy of the EIB response code EIBRESP.

UEPRESP2

Address of a 4-byte binary copy of the EIB response code EIBRESP2.

UEPTSTOK
Address of a 4-byte token which can be used to pass information between successive temporary storage requests within the same task (for example, between successive invocations of the XTSEREQ exit).

UEPRECUR
Address of a halfword recursion counter. The counter is set to 0 when the exit is first invoked, and is incremented for each recursive call.

UEPRSRCE
Address of an 8-character copy of the EIB resource value, EIBRSRCE.

Return codes

UERCBYP
Bypass this request.

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

API and SPI commands
All can be used, except for:
EXEC CICS SHUTDOWN
EXEC CICS XCTL

Note: Take care when issuing recursive commands. For example, you must avoid entering a loop when issuing a temporary storage request from the XTSEREQ exit. Use of the recursion counter UEPRECUR is recommended.

Exit XTSEREQC

The XTSEREQC exit allows you to intercept the response after a temporary storage API request has completed.

When invoked
After CICS processes a temporary storage API request, before return from the temporary storage EXEC interface program.

Exit-specific parameters

UEPCLPS
Address of a copy of the command parameter list. See "The command-level parameter structure" on page 229.

UEPTQTOK
Address of a 4-byte area which can be used to pass information between XTSEREQ and XTSEREQC for a single temporary storage request.

UEPRCODE
Address of a 6-byte hexadecimal copy of the EIB return code EIBRCODE. For details of EIB return codes, see EIB fields.

UEPRESP
Address of a 4-byte binary copy of the EIB response code EIBRESP.

UEPRESP2
Address of a 4-byte binary copy of the EIB response code EIBRESP2.

UEPTSTOK
Address of a 4-byte token which can be used to pass information between successive temporary storage requests within the same task (for example, between successive invocations of the XTSEREQC exit).

UEPRECUR
Address of a halfword recursion counter. The counter is set to 0 when the exit is first invoked, and is incremented for each recursive call.

UEPRSRCE
Address of an 8-character copy of the EIB resource value, EIBRSRCE.

UEP_TS_REMOTE_SYSTEM
If the request is to be sent to a remote region, is the address of an area containing the 4-byte name of the remote region. The remote region might have been specified by, for example, the `SYSID` option of the command, function shipping, or workload management.

If the request is to be executed on the local region, this parameter is the address of a 4-byte area containing blanks.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

API and SPI commands
All can be used, except for:

```
EXEC CICS SHUTDOWN
EXEC CICS XCTL
```

You can update the copies of EIBRSRCE, EIBRCODE, EIBRESP, and EIBRESP2 that you are given in the parameter list. If you update the values, temporary storage copies the new values into the application program's EIB after the completion of XTSEREQC or if you specify a return code of UERCBYP in XTSEREQ.

You must set valid temporary storage responses. You must set all three of EIBRCODE, EIBRESP, and EIBRESP2 to a consistent set of values, such as would be set by temporary storage to describe a valid completion. CICS does not check the consistency of EIBRCODE, EIBRESP, and EIBRESP2. If EIBRCODE is set to a non-zero value and EIBRESP is set to zero, CICS will override EIBRESP with a non-zero value. To help you set values for EIBRCODE, EIBRESP, and EIBRESP2, the values used by temporary storage are specified in DSECT DFHTSUED.

Note: Take care when issuing recursive commands not to cause a loop. For example, it is your responsibility to avoid entering a loop when issuing a temporary storage request from the XTSEREQC exit. Use of the recursion counter UEPRECUR is recommended.

The command-level parameter structure

The command-level parameter structure consists of a series of addresses. The first address points to the EXEC interface descriptor (EID), which consists of a bit string that describes the type of request and identifies each keyword specified with the request. The remaining addresses point to pieces of data associated with the request.

End of parameter list indicator

You can examine the EID to determine the type of request and the keywords specified. You can examine the other parameters in the list to determine the values of the keywords. You can also modify values of keywords specified on the request.

The high-order bit is set on in the last address set in the parameter list to indicate that it is the last one in the list. On return from your user exit program, CICS scans the parameter list for the high-order bit to find the last parameter. Therefore, if you modify the length of the parameter list, you must also reset the high-order bit to indicate which is the new last address.

The UEPCLPS exit-specific parameter

The UEPCLPS exit-specific parameter is included in both exit XTSEREQ and exit XTSEREQC. It is the address of the command-level parameter structure.

The command-level parameter structure contains 8 addresses, TS_ADDR0 through TS_ADDR7. It is defined in the DSECT TS_ADDR_LIST, which you should copy into your exit program by including the statement COPY DFHTSUED.

The command-level parameter list is made up as follows.

Note: The relationship between arguments, keywords, data types, and input/output types is summarized for the temporary storage commands in the following tables:

Table 8. The relationship between arguments, keywords, data types, and input/output types for the temporary storage commands

Command	See
WRITEQ TS	Table 9 on page 233
READQ TS	Table 10 on page 233
DELETEQ TS	Table 11 on page 233

TS_ADDR0
is the address of a 9-byte area called the EID, which is made up as follows:

- **TS_GROUP**
- **TS_FUNCT**
- **TS_BITS1**
- **TS_BITS2**
- **TS_EIDOPT5**
- **TS_EIDOPT6**

- **TS_EIDOPT7**
- **TS_EIDOPT8**

TS_GROUP
Always X'0A', indicating that this is a temporary storage request.

TS_FUNCT
One byte that defines the type of request:

X'02' WRITEQ

X'04' READQ

X'06' DELETEQ

TS_BITS1
Existence bits that define which arguments were specified. To obtain the argument associated with a keyword, you need to use the appropriate address from the command-level parameter structure. Before using this address, you must check the associated existence bit. If the existence bit is set off, the argument was not specified in the request and the address should not be used.

X'80' Set if the request contains an argument for the QUEUE or QNAME keyword. If set, **TS_ADDR1** is meaningful.

X'40' Set if the request contains an argument for any of the FROM, INTO, or SET keywords. If set, **TS_ADDR2** is meaningful.

X'20' Set if the request contains an argument for the LENGTH keyword. If set, **TS_ADDR3** is meaningful.

X'10' Set if the request contains an argument for the NUMITEMS keyword. If set, **TS_ADDR4** is meaningful.

X'08' Set if the request contains an argument for the NUMITEMS or ITEM keyword. If set, **TS_ADDR5** is meaningful.

X'02' Set if the request contains an argument for the SYSID keyword. If set, **TS_ADDR7** is meaningful.

TS_BITS2
Two bytes not used by temporary storage.

TS_EIDOPT5
Indicates whether certain keywords were specified on the request.

X'80' QNAME was specified (otherwise QUEUE). You can modify this bit in your user exit if you want.

TS_EIDOPT6
One byte not used by temporary storage.

TS_EIDOPT7
Indicates whether certain functions and/or keywords were specified on the request.

X'10' WRITEQ NOSUSPEND specified.

X'80' WRITEQ MAIN or READQ ITEM specified.

X'04' WRITEQ REWRITE or READQ NUMITEMS specified.

TS_EIDOPT8
Indicates whether certain keywords were specified on the request.

X'80' ITEM was specified (otherwise NUMITEMS).

TS_ADDR1
is the address of area containing 8-byte name from QUEUE. or 16-byte name from QNAME. To determine which of these is applied, see the TS_BITS2 field.

TS_ADDR2
is the address of one of the following:

- A 4-byte address from SET (if the request is READQ and **TS_EIDOPT5** indicates that this is SET).
- Data from INTO (if the request is READQ and **TS_EIDOPT5** indicates that this is not SET).
- Data from FROM (if the request is WRITEQ).

TS_ADDR3
is the address of the halfword value of LENGTH (if the request is READQ or WRITEQ).

Warning: For requests that specify INTO, do not change the value of LENGTH to a value greater than that specified by the application. To do so causes a storage overlay in the application.

TS_ADDR4
is the address of the halfword value of NUMITEMS (if the request is READQ).

TS_ADDR5
is the address of one of the following:

- The halfword value of NUMITEMS (if the request is WRITEQ)
- The halfword value of ITEM (if the request is READQ or WRITEQ).

TS_ADDR6
is the address of a value intended for CICS internal use only. It must not be used.

TS_ADDR7
is the address of an area containing the value of SYSID.

Modifying fields in the command-level parameter structure

Some fields that are passed to temporary storage are used as input to the request, some are used as output fields, and some are used for both input and output. The method your user exit program uses to modify a field depends on the usage of the field.

The following are always input fields:

- QUEUE | QNAME
- FROM
- SYSID

The following are always output fields:

- INTO
- NUMITEMS
- SET

LENGTH is an input field on a WRITEQ request, and an output field on a READQ request that specifies SET. It is both an input and an output field on a READQ request that specifies INTO.

ITEM is an input field on a READQ request, and on a WRITEQ request that specifies REWRITE. It is both an input and an output field on a WRITEQ request that does not specify REWRITE.

Modifying input fields

The correct method of modifying an input field is to create a new copy of it, and to change the address in the command-level parameter list to point to your new data.

Do not modify an input field by altering the data that is pointed to by the command-level parameter list. To do so would corrupt storage belonging to the application program and would cause a failure when the program attempted to reuse the field.

Modifying output fields

The technique described in "Modifying input fields" is not suitable for modifying output fields. (The results would be returned to the new area instead of the application's area, and would be invisible to the application.)

An output field is modified by altering the data that is pointed to by the command-level parameter list. In the case of an output field, you can modify the application's data in place, because the application is expecting the field to be modified anyway.

Modifying fields used for both input and output

An example of a field that is used for both input and output is LENGTH on a READQ request that specifies INTO. You can treat such fields in the same way as output fields, and they are considered to be the same.

Modifying the EID

It is not possible to modify the EID to make major changes to requests. It is not possible, for example, to change a READQ request to a WRITEQ request. However, you can make minor changes to requests.

The list that follows shows the bits in the EID that can be modified. Any attempt to modify any other part of the EID is ignored.

TS_BITS1

X'02' The existence bit for SYSID.

TS_EIDOPT7

A user exit program at XTSEREQ can set the following on or off for all WRITEQ TS commands:

X'10' The existence bit for NOSUSPEND.

X'08' The existence bit for MAIN.

The EID is reset to its original value before return to the application program. That is, changes made to the EID are retained for the duration of the temporary storage request only.

Note: Your user exit program is prevented from making major changes to the EID. However, you must take great care when making the minor modifications that **are** permitted.

Use of the task token UEPTSTOK

The task token UEPTSTOK provides the address of a 4-byte area that you can use to pass information between successive temporary storage requests in the same task.

For example, you can use UEPTSTOK to pass information between successive invocations of the XTSEREQ exit. By contrast, UEPTQTOK is usable only for the duration of a single temporary storage request, because its contents may be destroyed at the end of the request.

Table 9. WRITEQ TS: User arguments and associated keywords, data types, and input/output types

Argument	Keyword	Data type	Input/output type
Arg1	QUEUE	CHAR(8)	input
Arg1	QNAME	CHAR(16)	input
Arg2	FROM	DATA-AREA	input
Arg3	LENGTH	BIN(15)	input
Arg4	*	*	*
Arg5	ITEM	BIN(15)	input/output
Arg5	NUMITEMS	BIN(15)	output
Arg6	*	*	*
Arg7	SYSID	CHAR(4)	input

Note: The different uses of Arg5 are shown, because Arg5 is used by the ITEM and NUMITEMS keywords which are alternatives and the argument to the ITEM keyword is an input field when REWRITE is specified.

Table 10. READQ TS: User arguments and associated keywords, data types, and input/output types

Argument	Keyword	Data type	Input/output type
Arg1	QUEUE	CHAR(8)	input
Arg1	QNAME	CHAR(16)	input
Arg2	SET	DATA-AREA, PTR	output
Arg2	INTO	DATA-AREA	output
Arg3	LENGTH	BIN(15)	input/output
Arg4	NUMITEMS	BIN(15)	output
Arg5	ITEM	BIN(15)	input
Arg6	*	*	
Arg7	SYSID	CHAR(4)	input

Table 11. DELETEQ TS: User arguments and associated keywords, data types, and input/output types

Argument	Keyword	Data type	Input/output type
Arg1	QUEUE	CHAR(8)	input
Arg1	QNAME	CHAR(16)	input
Arg2	*	*	*
Arg3	*	*	*

Table 11. DELETEQ TS: User arguments and associated keywords, data types, and input/output types (continued)

Argument	Keyword	Data type	Input/output type
Arg4	*	*	*
Arg5	*	*	*
Arg6	*	*	*
Arg7	SYSID	CHAR(4)	input

Modifying user arguments

User exit programs can modify user arguments, as follows:

For input arguments, the user exit program should obtain sufficient storage to hold the modified argument, set up that storage to the required value, and set the associated pointer in the parameter list to the address of the newly acquired area.

For output arguments, and for input/output arguments, the user exit program can update the argument in place, because the area of storage is represented by a variable in the application which is expected to receive a value from CICS.

Note:

1. CICS does not check changes to argument values, so any changes must be verified by the user exit program making the changes.
2. It is not advisable for XTSEREQ to modify output arguments or for XTSEREQC to modify input arguments.

Adding user arguments

Global user exit programs can add arguments associated with the SYSID keyword. You must ensure that the arguments you specify or modify in your exit programs are valid.

Assuming that the argument to be added does not already exist, the user exit program must:

1. Obtain storage for the argument to be added
2. Initialize the storage to the required value
3. Select and set up the appropriate pointer from the parameter list
4. Select and set up the appropriate argument existence bit in the EID
5. Modify the parameter list to reflect the new end of list indicator.

Removing user arguments

User exit programs can remove arguments (for which the program is totally responsible) associated with the SYSID keyword:

Assuming that the argument to be removed exists, the user exit program must:

1. Switch the corresponding argument existence bit to '0'b in the EID
2. Modify the parameter list to reflect the new end of list indicator.

Example program

CICS supplies—as a softcopy listing only (not as a source code file)—an example program, DFH$XTSE, that shows how temporary storage requests can be modified.
.

Terminal allocation program exit XALCAID

XALCAID is driven when an automatic initiation descriptor (AID) with data is canceled either by the CEMT transaction, running a SET TERMINAL or SET CONNECTION command, or during the reinstallation of a terminal or connection.

XALCAID is invoked only if there is data associated with the AID.

When invoked

Whenever an AID with data is canceled.

Note: It is not possible for the exit to prevent the request from being canceled.

Exit-specific parameters

UEPALTSD

Address of a 4-byte field containing the symbolic identifier of the transaction which was to be started by this request.

UEPALTRM

Address of a 4-byte field containing the identifier of the terminal or connection to which this request was directed.

UEPALDAT

Address of an area of storage containing the data specified in the FROM option; or hexadecimal zeros, in either of the following cases:

- The AID was created by a START request without a FROM option.
- The AID is associated with a channel (in which case the field pointed to by UEPALCHN will be set to a name other than blanks).

UEPALLEN

Address of a fullword binary field containing the length of the FROM data; or hexadecimal zeros, in either of the following cases:

- The AID was created by a START request without a FROM option.
- The AID is associated with a channel (in which case the field pointed to by UEPALCHN will be set to a name other than blanks).

UEPALRQD

Address of an 8-byte field containing the value of the REQID associated with the FROM data. The data was stored in a temporary storage queue with this name. This value was either specified explicitly using the REQID option on the START command, or created internally by CICS.

UEPALQUE

Address of an 8-byte field containing the value specified in the QUEUE option on the START command; or hexadecimal zeros if QUEUE was not specified.

UEPALRTE

Address of a 4-byte field containing the value specified in the RTERMID option on the START command, or hexadecimal zeros if RTERMID was not specified.

UEPALRTA
Address of a 4-byte field containing the value specified in the RTRANSID option on the START command, or hexadecimal zeros if RTRANSID was not specified.

UEPALFMH
Address of a 1-byte field containing the value X'FF' if the data contains FMHs, as specified by the FMH option on the associated START command; or hexadecimal zeros otherwise.

UEPALSTC
Address of a 2-byte field containing the start code. This is "SZ" for FEPI starts; otherwise it is "SD".

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.

Return codes

UERCNORM
No other return codes are supplied. The value of the return code is not inspected.

XPI calls
You can use:

- INQ_APPLICATION_DATA
- INQUIRE_SYSTEM

No other XPI calls should be used.

API and SPI commands
No EXEC CICS commands can be used.

Note: The XALTENF exit, used to handle the "terminal not known" condition, is also invoked from the terminal allocation program. XALTENF is described in "'Terminal not known' condition exits XALTENF and XICTENF" on page 238.

Terminal control program exits (XTCIN, XTCOUT, XTCATT)

These exits are invoked before I/O events for sequential devices or before task attaches.

Exit XTCIN

When invoked
After an input event for a sequential device.

Exit-specific parameters

UEPTCTTE
Address of the terminal control table terminal entry (TCTTE). The TCTTE can be mapped using the DSECT DFHTCTTE.

UEPTIOA
Address of the terminal input/output area (TIOA). Your exit program should not change the address. The TIOA can be mapped using the DSECT DFHTIOA. However, fields TIOASAL and TIOASCA are **not** programming interfaces.

UEPTCTLE
Address of the terminal control table line entry (TCTLE). The TCTLE can be mapped using the DSECT DFHTCTLE.

Return codes

UERCNORM
Continue processing.

XPI calls
All can be used. However, note that you cannot use a GETMAIN call to obtain terminal-class storage for use as a replacement TIOA.

Exit XTCOUT

When invoked
Before an output event for a sequential device.

Exit-specific parameters

UEPTCTTE
Address of the terminal control table terminal entry (TCTTE). The TCTTE can be mapped using the DSECT DFHTCTTE.

UEPTIOA
Address of the terminal input/output area (TIOA). Your exit program should not change the address. The TIOA can be mapped using the DSECT DFHTIOA. However, fields TIOASAL and TIOASCA are **not** programming interfaces.

UEPTCTLE
Address of the terminal control table line entry (TCTLE). The TCTLE can be mapped using the DSECT DFHTCTLE.

Return codes

UERCNORM
Continue processing.

XPI calls
All can be used. However, note that you cannot use a GETMAIN call to obtain terminal-class storage for use as a replacement TIOA.

Exit XTCATT

When invoked
Before task attach.

Exit-specific parameters

UEPTCTTE
Address of the terminal control table terminal entry (TCTTE). The TCTTE can be mapped using the DSECT DFHTCTTE.

UEPTIOA
Address of the terminal input/output area (TIOA). The TIOA can be mapped using the DSECT DFHTIOA. However, fields TIOASAL and TIOASCA are **not** programming interfaces.

UEPTCTLE
Address of the terminal control table line entry (TCTLE). The TCTLE can be mapped using the DSECT DFHTCTLE.

UEPTRAN
Address of the 4-byte transaction id.

Return codes

UERCNORM
Continue processing.

XPI calls
All can be used.

'Terminal not known' condition exits XALTENF and XICTENF

The 'terminal not known' condition can occur when intercommunicating CICS regions use both SHIPPABLE terminal definitions and automatic transaction initiation (ATI). The condition is especially likely to arise if autoinstall is used.

SHIPPABLE attribute
Terminals defined with the SHIPPABLE attribute in a terminal-owning region (TOR) do not need a definition in a connected application-owning region (AOR). If necessary to support transaction routing, CICS ships a copy of the definition from the TOR to the AOR. For further information, refer to Shipping terminal and connection definitions.

Automatic transaction initiation (ATI)
ATI occurs when an internally generated request leads to the initiation of a transaction. For example, when:

- An application issues an EXEC CICS START command, or
- The transient data trigger level is reached.

Two CICS modules handle ATI requests:

The **interval control program** processes a START command, checks that the terminal is known in the local system, and (when any START time interval elapses) calls the terminal allocation program.

The **terminal allocation program** is called by the interval control program or by the transient data triggering mechanism, and checks that the terminal is known in the local system. If the requested terminal is remote, the terminal allocation program ships an ATI request to the remote system, which initiates transaction routing back to the local system.

For guidance information about ATI, refer to Traditional routing of transactions started by ATI.

'Terminal not known' condition
The 'terminal not known' condition arises when an ATI request is made for a terminal not known in the region. An ATI request can occur in the AOR for a SHIPPABLE terminal before any transaction routing has taken place for the terminal, and so before the definition of the terminal can have been shipped from the TOR to the AOR.

If the 'terminal not known' condition occurs, both the interval control program and the terminal allocation program reject the transaction-initiation request as 'TERMIDERR'.

The exits

To deal with the 'terminal not known' condition, CICS provides global user exits in the interval control and terminal allocation programs:

XICTENF
In the interval control program

XALTENF
In the terminal allocation program.

CICS drives the XICTENF exit when the 'terminal not known' condition occurs after the interval control program has been invoked by an EXEC CICS START command. CICS drives the XALTENF exit when the 'terminal not known' condition occurs after the terminal allocation program has been invoked by the transient data trigger level or the interval control program. Note that an EXEC CICS START command could result in both exits being invoked.

The exit program must indicate whether the terminal exists on another system and, if so, on which one. CICS passes data to the exit program to help establish this information. You can use the same exit program at both exit points. CICS supplies a sample exit program, DFHXTENF, that can be used at both exits and that can deal unchanged with some typical situations.

The exits are designed to deal with 'terminal not known' conditions that occur in CICS regions other than the TOR. For a TOR/AOR pair, enable the exit program in the AOR. The exits cannot deal with a 'terminal not known' condition in the TOR and the exit program should not normally be enabled there. However, if more than one TOR exists, you may need to enable the exit program in each TOR to deal with requests for terminals owned by other TORs. In this case, the exit program must recognize terminals that should be owned by this system and reject the requests ('UERCTEUN'). Although the exit provides as much data as possible, the logic of your program depends entirely on your system design. A simple solution to the most complex case would be to make the name of each terminal reflect the netname or sysid of its owning region.

Data returned by exit

The exit program must set a return code in register 15 as follows:

UERCTEUN
Terminal does not exist

UERCNETN
Netname of TOR returned

UERCSYSI
Sysid of TOR returned.

For return codes UERCNETN and UERCSYSI, the exit program must place the netname or sysid of the terminal-owning region in fields UEPxxNTO or UEPxxSYO (where xx is AL or IC).

If the terminal-owning region is a member of a z/OS Communications Server generic resource, the exit program should place the netname of the terminal in field UEPxxNNO. For information about using ATI with z/OS Communications Server generic resources, see Using ATI with generic resources.

Exit XALTENF

Exit XALTENF is invoked by the terminal allocation program when the terminal that an ATI request from transient data or interval control requires is unknown in this system. The exit program is expected to give a return code indicating whether the terminal exists on another connected CICS system and, if so, on which one.

Exit-specific parameters

UEPALEVT
Address of 2 bytes containing the type of request. The equated values of the types are:

UEPALESD
START command with data

UEPALES
START command without data

UEPALETD
Transient data trigger level reached.

UEPALTR
Address of 1 byte containing an indication of whether the task issuing the START command was started by transaction routing. The equated values are:

UEPALTY
A START command was being processed and the task issuing the command was transaction routed to.

UEPALTN
A START command was not being processed **or** a START command was being processed but the task issuing the command was not transaction routed to.

UEPALFS
Address of 1 byte containing an indication of whether the START command was function shipped. The equated values are:

UEPALFY
A START command was being processed and the START was function shipped.

UEPALFN
A START command was not being processed **or** a START was being processed but it was not function shipped.

UEPALTRN
Address of 4 bytes containing the name of the transaction to be run.

UEPALRTR
Address of 4 bytes containing the name of the terminal on which the transaction should run. (If a transient data trigger level was reached and the transient data queue definition specified a system, then this would contain a system identifier.)

UEPALCTR
Address of 4 bytes containing, for START commands, the name of the current terminal if the command was transaction routed, or the name of the session if the command was function shipped.

For other START commands and for transient data trigger events, the field pointed to contains blanks.

UEPALNTI
Address of 8 bytes containing, for function-shipped START commands, the netname of the last system from which the request came.

For START commands issued in this system by transaction routing to a task, the netname of the last system from which the task was routed.

For other START command situations and for transient data trigger level events, the field pointed to contains blanks.

UEPALSYI
Address of 4 bytes containing, if UEPALNTI contains a netname, the corresponding sysid.

If UEPALNTI does not contain a netname, the field pointed to is blank.

UEPALNTO
Address of 8 bytes containing the contents of UEPALNTI.

If it sets a return code of 'UERCNETN', your exit program must place in this field the netname of the system to which the ATI request should be sent.

UEPALSYO
Address of 4 bytes containing the contents of UEPALSYI.

If it sets a return code of 'UERCSYSI', your exit program must place in this field the sysid of the system to which the ATI request should be sent.

UEPALNNI
Address of a 4-byte input field containing the netname of the terminal on which the transaction is to run, if this is known to CICS. If CICS does not know the netname, the addressed field contains blanks.

UEPALNNO
Address of a 4-byte input/output field containing, on invocation, the contents of UEPALNNI. Your exit program can use this field to supply the netname of the terminal on which the transaction is to run. It is important that your exit program supply a terminal netname if the TOR to which it directs the ATI request is a member of a z/OS Communications Server generic resource.

Return codes

UERCTEUN
Terminal unknown, reject request.

UERCNETN
Terminal known, netname returned in UEPALNTO.

UERCSYSI
Terminal known, sysid returned in UEPALSYO.

XPI calls

You can use:

- INQ_APPLICATION_DATA
- INQUIRE_SYSTEM.

No other XPI calls should be used.

Sample exit program

DFHXTENF is a sample program that can be used for the XALTENF and XICTENF exits. For more information about DFHXTENF, see Terminal-not-known sample exit program: DFHXTENF.

Exit XICTENF

Exit XICTENF is invoked by the interval control program when the terminal that an EXEC CICS START command requires is unknown in this system.

When invoked

By the interval control program when the terminal that an EXEC CICS START command requires is unknown in this system. The exit program is expected to give a return code indicating whether the terminal exists on another connected CICS system and, if so, on which one.

Exit-specific parameters

UEPICEVT

Address of 2 bytes containing the type of request. The equated values of the types are:

UEPICESD

START command with data

UEPICES

START command without data.

UEPICTR

Address of 1 byte containing an indication of whether the task issuing the START command was started by transaction routing. The equated values are:

UEPICTY

A START command was being processed and the task issuing the command was transaction routed to.

UEPICTN

A START command was not being processed **or** a START command was being processed but the task issuing the command was not transaction routed to.

UEPICFS

Address of 1 byte containing an indication of whether the START command was function shipped. The equated values are:

UEPICFY

A START command was being processed and the START was function shipped.

UEPICFN

A START command was not being processed **or** a START was being processed but it was not function shipped.

UEPICTRN

Address of 4 bytes containing the name of the transaction to be run.

UEPICRTR

Address of 4 bytes containing the name of the terminal on which the transaction should run.

UEPICCTR
Address of 4 bytes containing, for START commands, the name of the current terminal if the command was transaction routed, or the name of the session if the command was function shipped.

For other START commands, the field pointed to contains blanks.

UEPICNTI
Address of 8 bytes containing, for function-shipped START commands, the netname of the last system from which the request came.

For START commands issued in this system by transaction routing to a task, the netname of the last system from which the task was routed.

For other START command situations, the field pointed to contains blanks.

UEPICSYI
Address of 4 bytes containing, if UEPICNTI contains a netname, the corresponding SYSID.

If UEPICNTI does not contain a netname, the field pointed to is blank.

UEPICNTO
Address of 8 bytes containing the contents of UEPICNTI.

If it sets a return code of 'UERCNETN', your exit program must place in this field the netname of the system to which the ATI request should be sent.

UEPICSYO
Address of 4 bytes containing the contents of UEPICSYI.

If it sets a return code of 'UERCSYSI', your exit program must place in this field the sysid of the system to which the ATI request should be sent.

UEPICNNI
Address of a 4-byte input field containing the netname of the terminal on which the transaction is to run, if this is known to CICS. If CICS does not know the netname, the addressed field contains blanks.

UEPICNNO
Address of a 4-byte input/output field containing, on invocation, the contents of UEPICNNI. Your exit program can use this field to supply the netname of the terminal on which the transaction is to run. It is important that your exit program supply a terminal netname if the TOR to which it directs the ATI request is a member of a z/OS Communications Server generic resource.

Return codes

UERCTEUN
Terminal unknown, reject request.

UERCNETN
Terminal known, netname returned in UEPICNTO.

UERCSYSI
Terminal known, sysid returned in UEPICSYO.

UERCPURG
Task purged during XPI call.

XPI calls
The following must not be used:

- ADD_SUSPEND
- DELETE_SUSPEND
- DEQUEUE
- ENQUEUE
- RESUME
- SUSPEND
- WAIT_MVS.

Sample exit program

DFHXTENF is a sample program that can be used for the XALTENF and XICTENF exits. For more information about DFHXTENF, see Terminal-not-known sample exit program: DFHXTENF.

Transaction manager domain exit XXMATT

Exit XXMATT is invoked during transaction attach, and is able to change some of the attributes of the transaction that is being attached.

The exit can change the attach transaction ID of the transaction by changing the field addressed by UEPATPTI. You cannot use **EXEC CICS** commands from this exit.

Exit-specific parameters

UEPTRANID
The address of transaction ID (see Notes).

UEPUSER
The address of the user ID associated with the transaction if the current task is a user task (see Notes).

UEPTERM
The address of the terminal ID associated with the transaction, if any (see Notes).

UEPPROG
The address of the application program name for this transaction, if any (see Notes).

UEPATPTI
The address of a 4 byte field containing the primary transaction ID. You can change the primary transaction ID by modifying the addressed field.

UEPATOTI
The address of the 4 byte attach transaction ID. A transid of X'00000000' indicates that a transid was not supplied on the attach.

UEPATTPL
The address of an area containing the length of the attach TPName. A length of zero indicates that a TPName was not supplied on the attach.

UEPATTPA
The address of a fullword containing the address of the attach TPName. The attach TPName can be 1 through 64 bytes long, as defined by UEPTTPL.

UEPATLOC
The address of a 1 byte field indicating whether the transaction was found. Note that if the transaction was not found but system initialization parameters DTRTRAN and DTRPGM are specified, the transaction specified on DTRTRAN is attached, and CICS considers that the transaction has been found.

Equated values are:

UEATFND
The transaction was found.

UEATNFND
The transaction was not found.

UEPATTST
The address of a 1 byte transaction definition state. Equated values for the definition state are:

UEATENAB
The transaction is enabled.

UEATDISA
The transaction is disabled.

UEPATTTK
The address of a doubleword containing a transaction token. Note that some of the transaction manager XPI calls require this token to identify the transaction that is being attached.

Return codes

UERCNORM
Continue attach processing.

XPI calls

The user exit can inquire on the transaction being attached, using the UEPATTTK transaction token as input to the XMIQ INQUIRE_TRANSACTION XPI call.

The exit can also set the total priority and TCLASS, using the XMIQ SET_TRANSACTION XPI call.

Most of the XPI calls can be used, but with caution since typically this exit is invoked under the TCP task. Thus it is advisable not to issue any XPI calls that might cause the TCP task to wait.

Note:

1. The following XPI calls can be useful for obtaining information that could be used to modify the attach of a transaction:
   - INQUIRE_TRANSACTION
   - INQUIRE_MXT
   - INQUIRE_TCLASS
   - INQUIRE_TRANDEF
   - INQUIRE_SYSTEM

2. The fields UEPTRANID, UEPUSER, UEPTERM, and UEPPROG are common to many of the domain global user exit points, and normally return values associated with the current user task. In the case of XXMATT, however, the user task that is being attached is **not** the current task when the exit is invoked. Until task attach is complete, the current task is the CICS task that is performing the attach.

   When the task being attached is for a task started by an immediate START command; that is, a START without an interval, the current task is the task that issues the START command, and the fields contain values associated with that task.

Transient data program exits (XTDREQ, XTDIN, XTDOUT)

The XTDREQ exit intercepts a transient data request before request analysis. The XTDOUT and XTDIN exits are invoked before and after data is exchanged with QSAM or VSAM.

The CICS transient data facility is threadsafe, so CICS can process transient data requests on an open TCB. Transient data requests are also threadsafe when you function ship them to a remote region over an IPIC connection. To optimize TCB switching and gain the performance benefits of the open transaction environment, programs running at XTDREQ, XTDIN, and XTDOUT must be coded to threadsafe standards and defined to CICS as threadsafe.

Exit XTDREQ

Exit XTDREQ is invoked before request analysis.

Exit-specific parameters

UEPTDQUE
Address of 4-byte TD queue name.

UEPTDTYP
Address of 1-byte TD request type. Values are:

UEPTDPUT
PUT request

UEPTDGET
GET request

UEPTDPUR
PURGE request.

Return codes

UERCNORM
Continue TD processing.

UERCTDOK
Quit TD processing – returning 'NORMAL' to the caller.

UERCTDNA
Quit TD processing – returning 'NOTAUTH' to the caller.

UERCPURG
Task purged during XPI call.

XPI calls
You can use:

- INQ_APPLICATION_DATA

- INQUIRE_SYSTEM
- WAIT_MVS

Do not use any other calls.

Exit XTDIN

Exit XTDIN is invoked after CICS receives data from QSAM (for extrapartition) or VSAM (for intrapartition).

Exit-specific parameters

UEPTDQUE
Address of the 4-byte TD queue name.

UEPTDAUD
Address of the unmodified TD data.

UEPTDLUD
Address of the fullword length of the unmodified TD data.

UEPTDAMD
Address of the TD data modified by the exit program.

UEPTDLMD
Address of the fullword length of the TD data modified by the exit program.

Return codes

UERCNORM
Continue TD processing.

UERCPURG
Task purged during XPI call.

XPI calls

You can use:

- INQ_APPLICATION_DATA
- INQUIRE_SYSTEM
- WAIT_MVS

Do not use any other calls.

Exit XTDOUT

Exit XTDOUT is invoked before CICS passes data to a QSAM (for extrapartition) or VSAM (for intrapartition) user-defined transient data queue.

Exit-specific parameters

UEPTDQUE
Address of the 4-byte TD queue name.

UEPTDAUD
Address of the unmodified TD data.

UEPTDLUD
Address of the fullword length of the unmodified TD data.

UEPTDAMD
Address of the TD data modified by the exit program.

UEPTDLMD
Address of the fullword length of TD data modified by the exit program.

UEPTDNUM
Address of the fullword containing the number of items in the list.

UEPTDCUR
Address of the fullword containing the number of the current item.

Return codes

UERCNORM
Continue TD processing.

UERCTDOK
Quit TD processing – returning 'NORMAL' to the caller.

Note: If you return UERCTDOK to suppress the first line of a multiline message, the rest of the message is not presented to XTDOUT, but is also suppressed.

UERCPURG
Task purged during XPI call.

XPI calls

You can use:

- INQ_APPLICATION_DATA
- INQUIRE_SYSTEM
- WAIT_MVS

Do not use any other calls.

Transient data EXEC interface program exits XTDEREQ and XTDEREQC

The XTDEREQ exit intercepts a transient data request before any action has been taken on it by transient data. The XTDEREQC exit intercepts a transient data request after transient data has completed its processing.

You can change the XTDEREQ exit to perform the following tasks:

- Analyze the request to determine its type, the keywords specified, and their values.
- Modify any value specified by the request before the command is executed.
- Set return codes to specify either of the following instructions:
  - CICS should continue with the request, with any modifications that you made.
  - CICS should bypass the request. If you set this return code, you must also set up return codes for the EXEC interface block (EIB), as if you had processed the request yourself.

You can change the XTDEREQC exit to perform the following tasks:

- Analyze the request, to determine its type, the keywords specified, and their values.
- Set return codes for the EIB.

The CICS transient data facility is threadsafe, so CICS can process transient data requests on an open TCB. Transient data requests are also threadsafe when you function ship them to a remote region over an IPIC connection. To optimize TCB switching and gain the performance benefits of the open transaction environment, programs that run at XTDEREQ and XTDEREQC must be coded to threadsafe standards and defined to CICS as threadsafe.

Both exits are passed eight parameters as follows:

- The address of the command-level parameter structure.
- The address of a token (UEPTDTOK) used to pass 4 bytes of data from XTDEREQ to XTDEREQC.
- The addresses of copies of four pieces of return code and resource information from the EIB.
- The address of a token (UEPTSTOK) that is valid throughout the life of a task.
- The address of an exit recursion count (UEPRECUR).

Example program

CICS supplies, as a softcopy listing only and not as a source code file, an example program, DFH$XTSE, that shows how to modify fields in the command-level parameter structure passed to EXEC interface exits.

Exit XTDEREQ

Exit XTDEREQ is invoked before CICS processes a transient data API request.

Exit-specific parameters

UEPCLPS
Address of the command-level parameter structure. See "The UEPCLPS exit-specific parameter" on page 253.

UEPTDTOK
Address of the 4-byte token to be passed to XTDEREQC. UEPTDTOK allows you, for example, to pass a work area to exit XTDEREQC.

UEPRCODE
Address of a 6-byte hexadecimal copy of the EIB return code 'EIBRCODE'. For details of EIB return codes, refer to EIB fields.

UEPRESP
Address of a 4-byte binary copy of the EIB response code 'EIBRESP'.

UEPRESP2
Address of a 4-byte binary copy of the EIB response code 'EIBRESP2'.

UEPTSTOK
Address of a 4-byte token that is valid throughout the life of a task. See Using the task token UEPTSTOK.

UEPRECUR
Address of a halfword recursion counter. The counter is set to 0 when the exit is first invoked, and is incremented for each recursive call.

UEPRSRCE
Address of an 8-character copy of the EIB resource value, EIBRSRCE.

Return codes

UERCNORM
Continue processing.

UERCBYP
The transient data EXEC interface program ignores this request.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

Although the exit permits the use of XPI GETMAIN and FREEMAIN calls, we recommend that you use the EXEC CICS GETMAIN and FREEMAIN commands instead.

API and SPI commands
All can be used, except for:
EXEC CICS SHUTDOWN
EXEC CICS XCTL

Note: Take care when issuing recursive commands. For example, you must avoid entering a loop when issuing a transient data request from the XTDEREQ exit. Use of the recursion counter UEPRECUR is recommended.

Exit XTDEREQC

Exit XTDEREQC is invoked after a transient data API request has completed, and before return from the transient data EXEC interface program.

Exit-specific parameters

UEPCLPS
Address of the command-level parameter structure. See "The UEPCLPS exit-specific parameter" on page 253.

UEPTDTOK
Address of the 4 byte token to be passed to XTDEREQC. This allows you, for example, to pass a work area to exit XTDEREQC.

UEPRCODE
Address of a 6-byte hexadecimal copy of the EIB return code 'EIBRCODE'. For details of EIB return codes, refer to EIB fields.

UEPRESP
Address of a 4-byte binary copy of the EIB response code 'EIBRESP'.

UEPRESP2
Address of a 4-byte binary copy of the EIB response code 'EIBRESP2'.

UEPTSTOK
Address of a 4-byte token that is valid throughout the life of a task. See Using the task token UEPTSTOK.

UEPRECUR
Address of a halfword recursion counter. The counter is set to 0 when the exit is first invoked, and is incremented for each recursive call.

UEPRSRCE
Address of an 8-character copy of the EIB resource value, EIBRSRCE.

UEP_TD_REMOTE_SYSTEM
If the request is to be sent to a remote region, is the address of an area containing the 4-byte name of the remote region. (The remote region may have been specified by, for example, the SYSID option of the command, function shipping, or the REMOTESYSTEM option of the TDQUEUE definition.)

If the request is to be executed on the local region, this parameter is the address of a 4-byte area containing blanks.

UEP_TD_REMOTE_NAME
If the request is to be sent to a remote region, is the address of an area containing the 4-character name by which the queue is known in the remote region.

Return codes

UERCNORM
Continue processing.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

Although the exit permits the use of XPI GETMAIN and FREEMAIN calls, we recommend that you use the EXEC CICS GETMAIN and FREEMAIN commands instead.

API and SPI commands
All can be used, except for:
EXEC CICS SHUTDOWN
EXEC CICS XCTL

Note: Take care when issuing recursive commands. For example, you must avoid entering a loop when issuing a transient data request from the XTDEREQC exit. Use of the recursion counter UEPRECUR is recommended.

The command-level parameter structure

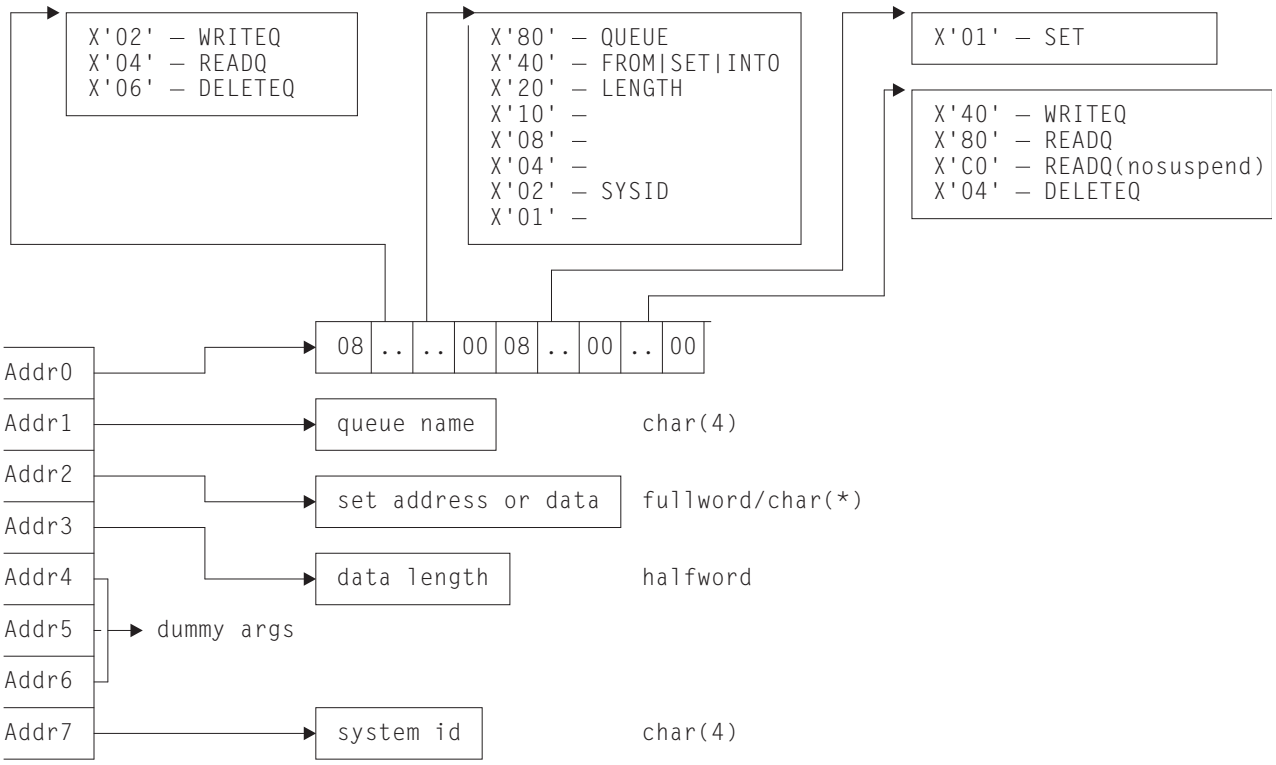


Figure 8. The command-level parameter structure for transient data

The command-level parameter structure consists of a series of addresses. The first address points to the EXEC interface descriptor (EID), which consists of an 8-byte area that describes the type of request and identifies each keyword specified with the request. The remaining addresses point to pieces of data associated with the request. (For example, the second address points to the queue name.)

You can examine the EID to determine the type of request and the keywords specified. You can examine the other parameters in the list to determine the values of the keywords. You can also modify values of keywords specified on the request. (For example, you could change the sysid specified in the request.)

End of parameter list indicator

The high-order bit is set on in the last address set in the parameter list to indicate that it is the last one in the list. On return from your user exit program, CICS scans the parameter list for the high-order bit to find the last parameter. Therefore, if you modify the length of the parameter list, you must also reset the high-order bit to indicate which is the new last address.

For example, if the parameter list specifies only the first two addresses (TD_ADDR0, the address of the EID, and TD_ADDR1, the address of the name of the queue named in a DELETEQ request), the high-order bit is set on in TD_ADDR1. If you extend the parameter list by setting the address of a SYSID in TD_ADDR7, you must reset the high-order bit in TD_ADDR1 and set it on in TD_ADDR7 instead.

The maximum size of parameter list is supplied to the exit, thus allowing your exit program to add any parameters not already specified without needing to first obtain more storage.

The original parameter list, as it was before XTDEREQ was invoked, is restored after the completion of XTDEREQC. It follows that the execution diagnostic facility (EDF) displays the original command before **and** after execution. **EDF does not display any changes made by the exit**.

The UEPCLPS exit-specific parameter

The UEPCLPS exit-specific parameter is included in both exit XTDEREQ and exit XTDEREQC. It contains the address of the command-level parameter structure.

The command-level parameter structure contains 8 addresses, TD_ADDR0 through TD_ADDR7. It is defined in the DSECT TD_ADDR_LIST, which you should copy into your exit program by including the statement COPY DFHTDUED.

The command-level parameter list is made up as follows:

TD_ADDR0
is the address of an 8-byte area called the EID, which is made up as follows:

- **TD_GROUP**
- **TD_FUNCT**
- **TD_BITS1**
- **TD_BITS2**
- **TD_EIDOPT5**
- **TD_EIDOPT6**
- **TD_EIDOPT7**

TD_GROUP
Always X'08', indicating that this is a transient data request.

TD_FUNCT
One byte that defines the type of request:

X'02' WRITEQ

X'04' READQ

X'06' DELETEQ.

TD_BITS1
Existence bits that define which arguments were specified. To obtain the argument associated with a keyword, you need to use the appropriate address from the command-level parameter structure. Before using this address, you must check the associated existence bit. If the existence bit is set off, the argument was not specified in the request and the address should not be used.

X'80' Set if the request contains an argument for the QUEUE keyword. If set, **TD_ADDR1** is meaningful.

X'40' Set if the request contains an argument for any of the INTO, SET, or FROM keywords. If set, **TD_ADDR2** is meaningful.

X'20' Set if the request contains an argument for the LENGTH keyword. If set, **TD_ADDR3** is meaningful.

X'02' Set if the request contains an argument for the SYSID keyword. If set, **TD_ADDR7** is meaningful.

TD_BITS2
Two bytes not used by transient data.

TD_EIDOPT5
Indicates whether certain keywords were specified on the request.

X'01' SET (and not INTO) was specified.

TD_EIDOPT6
One byte not used by transient data.

TD_EIDOPT7
Indicates whether certain functions, keywords or both were specified on the request:

X'40' WRITEQ specified

X'80' READQ specified

X'C0' READQ(nosuspend) specified

X'04' DELETEQ specified.

TD_ADDR1
is the address of a 4-byte area containing the name from QUEUE.

TD_ADDR2
is the address of one of the following:

- A 4-byte address from SET (if the request is READQ and **TD_EIDOPT5** indicates that this is SET).
- Data from INTO (if the request is READQ and **TD_EIDOPT5** indicates that this is not SET). You cannot modify this bit in your user exit.
- Data from FROM (if the request is WRITEQ).

TD_ADDR3
is the address of one of the following:

- The halfword value of LENGTH (if the request is READQ or WRITEQ). **Warning:** For requests that specify INTO, do not change the value of LENGTH to a value greater than that specified by the application. To do so causes a storage overlay in the application.

TD_ADDR4
is the address of a value intended for CICS internal use only. It must not be used.

TD_ADDR5
is the address of a value intended for CICS internal use only. It must not be used.

TD_ADDR6
is the address of a value intended for CICS internal use only. It must not be used.

TD_ADDR7
is the address of an area containing the value of SYSID.

TD_ADDR8
is the address of a value intended for CICS internal use only. It must not be used.

Modifying fields in the command-level parameter structure

Some fields that are passed to transient data are used as input to the request, some are used as output fields, and some are used for both input and output. The method your user exit program uses to modify a field depends on the usage of the field.

The following are always input fields:

- QUEUE
- FROM
- SYSID

The following are always output fields:

- INTO
- SET

LENGTH is an input field on a WRITEQ request, and an output field on a READQ request that specifies SET. It is both an input and an output field on a READQ request that specifies INTO.

Modifying input fields

The correct method of modifying an input field is to create a new copy of it, and to change the address in the command-level parameter list to point to your new data.

Note: You must never modify an input field by altering the data that is pointed to by the command-level parameter list. To do so would corrupt storage belonging to the application program and would cause a failure when the program attempted to reuse the field.

Modifying output fields

The technique described in "Modifying input fields" is not suitable for modifying output fields. (The results would be returned to the new area instead of the application's area, and would be invisible to the application.)

An output field is modified by altering the data that is pointed to by the command-level parameter list. In the case of an output field, you can modify the application's data in place, because the application is expecting the field to be modified.

Modifying fields used for both input and output

An example of a field that is used for both input and output is LENGTH on a READQ request that specifies INTO. You can treat such fields in the same way as output fields, and they are considered to be the same.

Modifying the EID

It is not possible to modify the EID to make major changes to requests, such as changing a READQ request to a WRITEQ request. However, you can make minor changes to requests, such as turning on the existence bit for SYSID so that the request can be changed into one that is shipped to a remote system.

The list that follows shows the bits in the EID that can be modified. Any attempt to modify any other part of the EID is ignored.

TD_BITS1

X'20' The existence bit for LENGTH.

X'02' The existence bit for SYSID.

TD_EIDOPT5

X'01' Existence bit for SET keyword. You cannot modify this bit from your user exit.

TD_EIDOPT7

Changes to TD_EIDOPT7 are limited to READQ requests. X'80'-READQ is interchangeable with X'C0'-READQ(nosuspend). No other changes may be made to this byte.

The EID is reset to its original value before return to the application program. That is, changes made to the EID are retained for the duration of the transient data request only.

Note: Your user exit program is prevented from making major changes to the EID. However, you must take great care when making the minor modifications that **are** permitted.

The EIB

Copies of EIBRSRCE, EIBRCODE, EIBRESP, and EIBRESP2 are passed to the exit, so that you can:

- Modify or set completion and resource information in XTDEREQ and XTDEREQC
- Examine completion and resource information in XTDEREQC.

You can update the copies of EIBRSRCE, EIBRCODE, EIBRESP, and EIBRESP2 that you are given in the parameter list. Transient data copies your values into the real EIB after the completion of XTDEREQC; or if you specify a return code of ' bypass' in XTDEREQ.

You must set valid transient data responses. You must set all three of EIBRCODE, EIBRESP, and EIBRESP2 to a consistent set of values, such as would be set by CICS transient data to describe a valid completion. **CICS does not police the consistency of EIBRCODE, EIBRESP, and EIBRESP2**. However, if EIBRCODE is set to a non-zero value and EIBRESP is set to zero then CICS will override EIBRESP with a non-zero value. To aid you in setting the values of EIBRCODE, EIBRESP, and EIBRESP2, the values used by transient data are specified in DFHTDUED.

User log record recovery program exits XRCINIT and XRCINPT

At warm and emergency restart, updates made to recoverable CICS resources that were not committed when the system terminated must be backed out. XRCINIT and XRCINPT are invoked from the user log record recovery program, which is used to back out, where necessary, user-written system log entries.

XRCINIT is invoked at warm and emergency restart:

- Before the first user recovery record is delivered to XRCINPT
- When all such records have been delivered to XRCINPT.

XRCINPT is invoked whenever a user log record is read from the system log.

You can use XRCINPT to change the default actions taken by CICS at emergency restart for particular user-journaled records. Records passed to XRCINPT are those in UOWs that:

- Appeared in the last complete activity keypoint
- Were in flight when CICS terminated

- Committed, backed out, or went indoubt after the start of the last complete activity keypoint. (However, this only applies to those records for which the leftmost bit of the JTYPEID specified in the WRITE JOURNALNAME(DFHLOG) request was a one.)

Records written by the activity keypoint exit XAKUSER are passed only if they appear in the last complete activity keypoint. They are passed after all other records. The order of presentation of records may therefore be different from their order in the reverse log stream sequence.

The format of records passed to the exit is:

Offset	Field contents
0	JTYPEID
2	Reserved
4	Length of prefix data (L). (Zero if no prefix)
8	Prefix data (if any)
8 + L	Log data

The record is mapped by the DSECT CL_USER_HEADER in copybook DFHLGGFD.

When using XRCINIT and XRCINPT, you should bear in mind that the exits may be invoked before recovery of temporary storage and transient data resources is complete.

Coding the exit programs

You can use CICS services in exit programs invoked from these exits using the XPI or EXEC CICS commands.

You need to consider the following:

- There is a restriction on using the XPI early during initialization: do not invoke exit programs that use the XPI functions TRANSACTION_DUMP, WRITE_JOURNAL_DATA, MONITOR and INQUIRE_MONITOR_DATA until the second phase of the PLTPI.
- There are also restrictions on the use of EXEC CICS commands in these exits:
  - You cannot use EXEC CICS commands to access terminal control services.
  - You are strongly advised not to use temporary storage, transient data, file control, journal control, or DL/I services, because the resources that you try to access may also be in a state of recovery and therefore "not open for business". Attempting to access resources in these circumstances causes, at best, serialization of the recovery tasks and, at worst, a deadlock.

    If you do issue file control requests in programs invoked from these exits, note that:

    - If an exit program acquires an area as a result of a file control request, it is the responsibility of the program to release that area.
    - An exit program must not attempt to make any file control requests to a file referring to a VSAM data set with a string number of 1, unless no action is specified for that file during the initialization exit.
  - Your exit program must not issue EXEC CICS commands if the recovery is as the result of an EXEC CICS SYNCPOINT ROLLBACK request.

  - Exit programs that issue EXEC CICS commands must first address the EIB. See Using CICS services.
  - Exit programs that issue EXEC CICS commands, and that use the DFHEIENT macro, should use the DFHEIRET macro to set a return code and return to CICS. See Global user exit XPI examples, showing the use of storage.
  - Exit programs invoked from these exits must be translated with the NOEDF option, if they issue EXEC CICS commands. See EDF and global user exits.
- Task-chained storage acquired in an exit program is released at the completion of emergency restart processing. However, the exit program should attempt to release the storage as soon as its contents are no longer needed.
- No exit program should reset either the absent or no-action indicators set by the file control backout program.
- Take care when issuing recursive commands not to cause a loop. For example, it is your responsibility to avoid entering a loop when an RC request is issued from these exits.

Enabling the exit programs

To enable these exits, you must do one of the following:

- Specify the system initialization parameter TBEXITS=(name1,name2,name3,name4,name5,name6), where name1 through name6 are the names of your user exit programs for XRCINIT, XRCINPT, XFCBFAIL, XFCLDEL, XFCBOVER, and XFCBOUT.
- Enable the exits during the first stage of initialization using a PLTPI program.

If you use the TBEXITS parameter to enable the exits, a global work area of 4 bytes is provided. If you use a PLTPI program, you can select the size of the global work area. You can also enable more than one exit program for use at each exit point; the TBEXITS parameter allows only one exit program at each exit point. PLTPI processing is described in Writing initialization and shutdown programs.

Exit XRCINIT

When invoked

At warm and emergency restart:

- Before the first user recovery record is delivered to XRCINPT
- When all such records have been delivered to XRCINPT.

Exit-specific parameters

UEPTREQ

Address of a 1-byte flag indicating the reason for the call. When UEPTREQ has a value of UEUSINIT, the exit has been invoked at the start of user recovery, and when UEPTREQ has a value of UEUSTERM, the exit has been invoked at the end of user recovery.

UEPRSTRT

Address of a 1-byte flag that indicates how CICS was restarted:

UEPRWARM

Warm start

UEPREMER

Emergency start.

Return codes

UERCNORM
Continue processing. No other return codes are supported.

XPI calls
All can be used. See "User log record recovery program exits XRCINIT and XRCINPT" on page 256 for restrictions.

Exit XRCINPT

When invoked
At warm and emergency restart, once for each user log record found in the system log.

Exit-specific parameters

UEPUOWST
Address of a 1-byte flag indicating the disposition of the unit of work. The possible values are:

UEPUOWAK
Activity keypoint record

UEPUOWCM
Unit of work committed

UEPUOWBO
Unit of work backed out

UEPUOWIF
Unit of work was in-flight

UEPUOWID
Unit of work was indoubt.

UEPLGREC
Address of the log record just read. The journal control record can be mapped using the information supplied in CICS logging and journaling.

UEPLGLEN
Address of a fullword containing the length of the log record.

UEPTAID
Address of a 4-byte field containing the task identifier.

UEPTRID
Address of a 4-byte field containing the transaction identifier.

UEPTEID
Address of a 4-byte field containing the terminal identifier.

Note: The values of the fields addressed by UEPTAID, UEPTRID, and UEPTEID are meaningless for activity keypoint records (that is, if the field addressed by UEPUOWST contains UEPUOWAK).

Return codes

UERCNORM
Continue processing.

UERCBYP
Bypass this record.

XPI calls
All can be used. See "User log record recovery program exits XRCINIT and XRCINPT" on page 256 for restrictions.

SNA LU management program exit (XZCATT)

This exit is invoked before a task attach for a LU terminal task.

When invoked
Before task attach for terminal tasks.

Exit-specific parameters

UEPTCTTE
Address of the terminal control table terminal entry (TCTTE). The TCTTE can be mapped using the DSECT DFHTCTTE.

UEPTIOA
Address of the terminal input/output area (TIOA). The TIOA can be mapped using the DSECT DFHTIOA. However, fields TIOASAL and TIOASCA are **not** programming interfaces.

UEPTPN
Address of the APPC transaction process name (TPN), or the LU6.1 process name (DPN), whose length is addressed by the parameter UEPTPNL.

UEPTPNL
Address of a 1-byte field containing the length of the TPN or DPN.

UEPTRAN
Address of the 4-byte transaction ID.

Note: The exit program must not change the TRANSID of tasks started by automatic transaction initiation (ATI). (This is because CICS needs to match the TRANSID in its program control table with the TRANSID in the automatic initiate descriptor (AID) that was created in the AOR.)

Return codes

UERCNORM
Continue processing.

XPI calls
All can be used.

Sample exit program
DFH$ZCAT

SNA working-set module exits (XZCIN, XZCOUT, XZCOUT1, and XZIQUE)

These exits are invoked after I/O events or before messages are disassembled into request units (RUs).

Exit XZCIN

This exit is invoked after an input event.

When invoked
After an input event.

Exit-specific parameters

UEPTCTTE
Address of the terminal control table terminal entry (TCTTE). The TCTTE can be mapped using the DSECT DFHTCTTE.

UEPTIOA
Address of the terminal input/output area (TIOA). Your exit program should not change the address. The TIOA can be mapped using the DSECT DFHTIOA. However, fields TIOASAL and TIOASCA are not programming interfaces.

Return codes

UERCNORM
Continue processing.

XPI calls
All can be used. However, do not use a GETMAIN call to obtain terminal-class storage for use as a replacement TIOA. This is because there are several internal pointers to the TIOA, and if any one of these is not updated the application might experience problems.

API and SPI commands
No **EXEC CICS** commands can be used.

Exit XZCOUT

This exit is invoked before an output event.

When invoked
Before an output event.

Exit-specific parameters

UEPTCTTE
Address of the terminal control table terminal entry (TCTTE). The TCTTE can be mapped using the DSECT DFHTCTTE.

UEPTIOA
Address of the terminal input/output area (TIOA). Your exit program should not change the address. The TIOA can be mapped using the DSECT DFHTIOA. However, fields TIOASAL and TIOASCA are **not** programming interfaces.

Note: In certain circumstances—for example, when XZCOUT is invoked before the send of a NULL RU—UEPTIOA contains zeroes.

Return codes

UERCNORM
Continue processing.

XPI calls
All can be used. However, we do not recommend that you use a GETMAIN call to obtain terminal-class storage for use as a replacement TIOA. This is because there are several internal pointers to the TIOA, and if any one of these is not updated the application may experience problems.

Exit XZCOUT1

This exit is invoked before a message is deconstructed into RUs.

When invoked

Before a message is broken into RUs.

Exit-specific parameters

UEPTCTTE

Address of the terminal control table terminal entry (TCTTE). The TCTTE can be mapped using the DSECT DFHTCTTE.

UEPTIOA

Address of the terminal input/output area (TIOA). Your exit program should not change the address. The TIOA can be mapped using the DSECT DFHTIOA. However, fields TIOASAL and TIOASCA are **not** programming interfaces.

Return codes

UERCNORM

Continue processing.

XPI calls

All can be used. However, we do not recommend that you use a GETMAIN call to obtain terminal-class storage for use as a replacement TIOA. This is because there are several internal pointers to the TIOA, and if any one of these is not updated the application may experience problems.

XZIQUE exit for managing MRO and APPC intersystem queues

You can use the XZIQUE exit to control the number of queued requests for sessions on MRO and APPC connections.

Note:

- Queued requests for sessions are known as "*allocate queues*".
- The equivalent global user exit to control the number of queued requests for sessions on IP interconnectivity (IPIC) connections is XISQUE: see "XISQUE exit for managing IPIC intersystem queues" on page 269.
- There are several methods that you can use to control the length of intersystem queues. For a description of the various methods, see Intersystem session queue management.

The XZIQUE exit enables you detect queuing problems (bottlenecks) early. It extends the function provided by the XISCONA global user exit, that is described in "Intersystem communication program exits, XISCONA, XISLCLQ, and XISQLCL" on page 131, which is invoked only for function shipping and DPL requests. XZIQUE is invoked for transaction routing, asynchronous processing, and distributed transaction processing requests, as well as for function shipping and DPL. Compared with XISCONA, it receives more detailed information on which to base its decisions.

XZIQUE enables allocate requests to be queued or rejected, depending on the length of the queue. It also allows a connection on which there is a bottleneck to be terminated and then re-established.

Interaction with the XISCONA exit

There is no interaction between the XZIQUE and XISCONA global user exits. If you enable both exits, XISCONA and XZIQUE could both be invoked for function shipping and DPL requests, although this is not recommended.

Therefore, you should ensure that only one of these exits is enabled. Because of it provides more function and greater flexibility, it is recommended that you use XZIQUE rather than XISCONA.

If you already have an XISCONA global user exit program, you could possibly modify it for use at the XZIQUE exit point.

When the XZIQUE exit is invoked

The XZIQUE global user exit is invoked, if it is enabled, at the following times:

- Whenever CICS tries to acquire a session with a remote system and there is no free session available. It is invoked whether or not you have specified the QUEUELIMIT option on the CONNECTION definition, and whether or not the limit has been exceeded. It is not invoked if the allocate request specifies NOQUEUE or NOSUSPEND.

  Requests for sessions can arise in a number of ways, such as explicit EXEC CICS ALLOCATE commands issued by DTP programs, or by transaction routing or function shipping requests.
- Whenever an allocate request succeeds in finding a free session, after the queue on the connection has been purged by a previous invocation of the exit program. In this case, your exit program can indicate that CICS is to continue processing normally, resuming queuing when necessary.

Using an XZIQUE global user exit program

When the exit is enabled, your XZIQUE global user exit program is able to check on the state of the allocate queue for a particular connection in the local system.

Information is passed to the exit program in a parameter list, that is structured to provide data about non-specific allocate requests, or requests for specific modegroups, depending on the session request. Non-specific allocate requests are for MRO, LU6.1, and APPC sessions that do not specify a modegroup.

Using the information passed in the parameter list, your global user exit program can decide (based on queue length, for example) whether CICS is to queue the allocate request. Your program communicates its decision to CICS by means of one of the return codes CICS provides. These are:

UERCAQUE
This return code indicates that CICS is to queue the allocate request.

The total number of allocate requests queued against the connection is provided in field A14ESTAQ of the system entry statistics (for all non-specific allocates) or A20ESTAQ of the mode entry statistics (for specific modegroup allocates). See DSECTs DFHA14DS or DFHA20DS for details. CICS passes to the exit program, in the exit specific parameter UEPQUELIM, the QUEUELIMIT parameter from the connection definition.

If the limit has not been reached, you can return control to CICS with return code UERCAQUE.

UERCAPUR
This return code indicates that CICS is to reject the allocate request and return SYSIDERR to the application program, but leave the existing queue unchanged.

If the number of queued allocate requests has reached the limit set on the QUEUELIMIT parameter for the connection, you can request that CICS rejects the request. However, you should first check whether the state of the link is satisfactory. This means checking that the rate of allocation of sessions is acceptable. Use the time the queue was started, the current time, and the total number of allocates processed since the queue began, to determine the rate at which CICS is processing requests. The relevant fields are: UEPSAQTS and UEPSACNT for non-specific allocate requests; and UEPMAQTS and UEPMACNT for specific modegroup requests.

To determine whether CICS is allocating requests for sessions on this connection at an acceptable rate, you can compare the calculated time with either of the following:

1. The parameter from the connection definition, MAXQTIME, which is passed in the exit specific parameter UEPEMXQT
2. Some other preset time value.

If the processing time using this kind of formula is acceptable, return control to CICS with return code UERCAPUR to purge only this request.

UERCAKLL or UERCAKLM

These return codes indicate that you want CICS to deal with the request as follows:

- UERCAKLL—reject this request, purge all other queued allocate requests on this connection, and send an information message to the operator console.
- UERCAKLM—reject this request, purge all other queued modegroup allocate requests on this connection, and send an information message to the operator console.

If the queue limit has been reached but the performance of allocate processing against the queue is below the acceptable limits defined in your user exit program, you can return control to CICS as follows:

- For non-specific allocate requests, use return code UERCAKLL. UERCAKLL also returns SYSIDERR to all application programs waiting on the purged allocate requests. CICS sets the UEPFLAG parameter to UEPRC8 on subsequent calls to your XZIQUE exit program to indicate that UERCAKLL was returned previously to purge the queue.
- For specific modegroup allocate requests, use return code UERCAKLM. UERCAKLM also returns SYSIDERR to all application programs waiting on the purged allocate requests. CICS sets the UEPFLAG parameter to UEPRC12 on subsequent calls to your XZIQUE exit program to indicate that UERCAKLM was returned previously to purge the queue.

Purging a queue that is causing congestion in the flow of tasks frees task slots that are needed to prevent the system becoming clogged. The more you allow a session queue to grow, the more likely you are to reach the task ceiling set by the MAXT parameter, and then cause a queue of incoming tasks in the local region that cannot be attached. Note that some internal CICS requests (such as those for the LU services model transactions CLS1, CLS2, and CLS3) are not purged by return codes UERCAKLL and UERCAKLM.

If a queue has been purged previously (with UERCAKLL or UERCAKLM) but there are no queued requests currently, check the number of successful allocates since the queue was last purged. For non-specific allocate requests, this number is in UEPSARC8, and for specific modegroup requests, this number is in UEPMAR12. If no requests of this type have been allocated on

this connection since the queue was last purged, the problem that caused the purge previously has not been resolved, and this request should be rejected with UERCAPUR.

If the UEPSARC8 or UEPMAR12 parameters show that allocates are being processed, you should use UERCAQUE to resume queuing of requests. If you return with UERCAQUE in this case, CICS issues an information message to the console to signal that queuing has been resumed.

Note: The address of the system entry statistics record, UEPCONST, is supplied for both non-specific and specific modegroup allocate requests.

The address of the modegroup statistics record, UEPMODST, is set to zeros for non-specific allocate requests. This address is supplied only if the request is for a specific modegroup.

If the exit is invoked after a successful allocate following the suppression of queuing, you can use the following return code:

UERCNORM
This return code indicates that CICS is to resume normal processing on the link, including queuing of requests.

Statistics fields in DFHA14DS and DFHA20DS

There are some statistics fields that your XZIQUE global user exit program can use to control queues.

A14EALRJ:

Each time an XZIQUE global user exit program returns with a request to reject a request, CICS increments field A14EALRJ in the system entry connection statistics

Field A14EALRJ (allocate rejected) is in DSECT DFHA14DS and is provided to help you to tune the queue limit. Normally, if the number of sessions and the queue limit defined for a link are correctly balanced, and there has been no abnormal congestion on the link, the A14EALRJ should be zero. If the rejected allocates field is non-zero it probably indicates that some action is needed.

A14EQPCT and A20EQPCT:

Each time an XZIQUE global user exit program returns with a request to purge a queue, CICS increments a new field in either the system entry connection statistics (field A14EQPCT) or mode entry connection statistics (field A20EQPCT).

A14EQPCT
The count of the number of times the queue has been purged for the connection as a whole.

A20EQPCT
The count of the number of times the mode group queue has been purged.

For detailed information about statistics fields, what they contain and how they are updated, see ISC/IRC system entry: Resource statistics.

Exit XZIQUE

Exit XZIQUE is invoked when an allocate request for a session is about to be queued, and when an allocate request succeeds following previous suppression of queuing.

When invoked

Whenever:

1. An allocate request for a session is about to be queued
2. An allocate request succeeds following previous suppression of queuing.

Exit-specific parameters

UEPZDATA
Address of the 70-byte area containing the information listed. This area is mapped by the DSECT in copybook DFHXZIDS.

Area addressed by UEPZDATA

UEPSYSID
The 4-byte SYSID of the connection.

UEPREQ
A 2-byte origin-of-request code, which can have the following values:

TR Transaction routing

FS Function shipping (includes distributed program link)

AL Other kinds of intercommunication (for example, distributed transaction processing (DTP) or CPI Communications).

UEPREQTR
The 4-byte identifier of the requesting transaction (applicable only when the origin-of-request code is FS or AL).

UEPTRAN
The 4-byte identifier of the transaction being routed (applicable only when origin of request is TR).

UEPFLAG
A 1-byte flag indicating whether a return code 8 or return code 12 was issued last time the exit was invoked.

UEPRC8
The exit program returned control to CICS on the previous invocation with return code 8.

UEPRC12
The exit program returned control to CICS on the previous invocation with return code 12.

UEPPAD
A 1-byte padding field.

UEPFSPL
Address of the 10-byte function shipping parameter list.

UEPCONST
Address of the 158-byte system entry statistics record (this can be mapped using DSECT DFHA14DS).

UEPMODST
Address of the 84-byte modegroup statistics record for the modegroup specified in the relevant CICS profile. This field applies only to APPC connections for a specific allocate. For LU61, IRC, or non-specific APPC allocates, it contains zero.

The statistics record can be mapped using DSECT DFHA20DS. The modegroup name field (A20MODE) may contain blanks. The record is followed by a fullword of X'FFFFFFFF'.

UEPSTEX
A 6-byte area containing additional current statistics for APPC that are not already in the modegroup statistics record (DFHA20DS). For specific allocates, the numbers refer to the specified modegroup only. For non-specific allocates, they refer to the whole connection—that is, they are the totals of each modegroup.

The 6-byte area contains:

UEPEBND
A halfword binary field containing the number of bound sessions

UEPEWWT
A halfword binary field containing the number of contention winners with tasks

UEPELWT
A halfword binary field containing the number of contention losers with tasks.

UEPEMXQT
A halfword binary field containing the maximum queuing time specified for the connection (MAXQTIME on the CONNECTION resource definition).

UEPMDGST
Address of a set of 84-byte modegroup statistics records—one for each user modegroup for the connection. This field applies only to APPC connections for a non-specific allocate. For LU61, IRC, and APPC specific allocates, it contains zero.

Each statistics record can be mapped using DSECT DFHA20DS. The modegroup name field (A20MODE) may contain blanks. The end of the set of records is indicated by a fullword of X'FFFFFFFF'.

Non-specific allocates data: The following three fields contain data relating to MRO, LU6.1, and non-specific APPC allocates:

UEPSAQTS
A double-word binary field containing the time stamp from the TCT system entry indicating the time the queue of non-specific requests was started.

UEPSACNT
A half-word binary field containing the number of all non-specific allocates processed since the queue was started (see UEPSAQTS for the start time).

UEPSARC8
A half-word binary field containing the number of sessions freed since the queue was last purged as a result of a UEPCAKLL return code to CICS.

Specific allocates data: The following three fields contain data relating to specific modegroup allocates. They are applicable only when UEPMODST is non-zero (that is, it contains the address of the relevant modegroup statistics).

UEPMAQTS
A double-word binary field containing the time stamp from the TCT mode entry indicating the time that the modegroup queue was started for this specific modegroup.

UEPMACNT
A half-word binary field containing the number of all specific allocates for this modegroup processed since the queue was started (see UEPMAQTS for the start time).

UEPMAR12
A half-word binary field containing the number of modegroup sessions freed since the queue was last purged as a result of a UEPCAKLL return code to CICS.

UEPQUELM
A half-word binary field containing the queue limit specified for this connection (QUEUELIMIT on the CONNECTION definition).

Return codes
In the case of an allocate that is about to be queued, use one of the following:

UERCAQUE
Queue the allocate request.

UERCAPUR
Reject the allocate request with SYSIDERR.

UERCAKLL
Reject this allocate request with SYSIDERR. Purge all other queued allocate requests and send an information message to the operator console. CICS also returns SYSIDERR to all application programs waiting on the purged allocate requests.

UERCAKLM
Reject this allocate request for the modegroup and return SYSIDERR. Purge all other queued allocate requests for the modegroup specified on this allocate request and send an information message to the operator console. Retry the modegroup after an interval.

UERCPURG
Task purged during XPI call.

In the case of a successful allocate following the use of UERCAKLL or UERCAKLM, on a previous invocation of the exit, use one of the following:

UERCNORM
Resume normal operation of the link or modegroup.

UERCAPUR
Reject the allocate request with SYSIDERR.

XPI calls
All can be used.

Designing an XZIQUE global user exit program

The functions of your XZIQUE exit should be designed:

1. To control of the number of tasks (and the amount of associated resource) that are waiting in a queue for a free intersystem session. Waiting tasks can degrade the performance of the local system.
2. To detect poor response from the receiving (remote) system and to notify the operator (or automatic operations program).
3. To cause CICS to issue a message when the link resumes normal operation.

The XZIQUE global user exit parameter list is designed to support these objectives.

Sample XZIQUE exit program

A sample XZIQUE exit program, DFH$XZIQ, is provided with CICS Transaction Server for z/OS, Version 5 Release 4 as a base for you to design your own global user exit program. The sample is supplied in the CICSTS54.CICS.SDFHSAMP library. The DSECT object DFHXZIDS, which is used by the sample program to map the area addressed by UEPZDATA is supplied in the CICSTS54.CICS.SDFHMAC library.

For more information about DFH$XZIQ, see MRO and APPC session queue management sample exit program: DFH$XZIQ.

Design considerations

The information passed at XZIQUE is designed to enable your XZIQUE global user exit program to:

- Avoid false diagnosis of problems on the connection by distinguishing poor response times from a complete bottleneck
- Ensure that a link resumes normal operation quickly and without operator intervention once any problem in a remote system is resolved.

Some guidance on the use of IRC/ISC statistics

CICS adds an entry for unsatisfied allocate requests to the *non-specific (generic) allocate queue*, and the *specific modegroup allocate queue*.

Non-specific (generic) allocate queue
: All non-specific allocate requests are queued in this single queue. CICS makes the total number of entries in this queue available in the system entry statistics field A14ESTAQ, to which your global user exit program has access by means of the address of the system entry statistics, which is passed in UEPCONST.

Specific modegroup allocate queues
: Specific allocate requests are queued in the appropriate modegroup queue—one queue for each specific modegroup name. CICS makes the total number of entries in all these queues available, as a single total, in the mode entry statistics field A20ESTAQ, to which your global user exit program has access by means of the address of the mode entry statistics, which is passed in UEPMODST.

XISQUE exit for managing IPIC intersystem queues

You can use the XISQUE exit to control queuing on IP interconnectivity (IPIC) connections.

The XISQUE exit controls these requests or commands that are queued on the IPIC connection:

- Distributed program link (DPL) requests for sessions
- Transaction routing requests

- Function shipping requests
- START or CANCEL commands.

Use the XISQUE exit to detect queuing problems (bottlenecks) early.

XISQUE enables allocate requests to be queued or rejected, depending on the length of the queue. It also allows an IPCONN on which there is a bottleneck to be ended and then reestablished.

Exit XISQUE

The XISQUE global user exit is called, if it is enabled, when CICS attempts to acquire a session and no free session is available, or when an allocate request finds a free session after queuing has previously been suppressed. Exit-specific parameters, return codes, and XPI call information are explained.

When called

The XISQUE exit is called under these circumstances:

1. CICS tries to acquire a session on an IPIC connection to a remote system and no free session is available. It is called whether or not you have specified the QUEUELIMIT option on the IPCONN definition and whether or not the limit has been exceeded.

   Requests for IPIC sessions occur when one of the following requests or commands is used across an IPIC connection:

   - A distributed program link (DPL) request
   - A START or CANCEL command
   - A transaction routing request
   - A function shipping file control, transient data, or temporary storage request
2. An IPIC allocate request succeeds in finding a free session, after the queue on the IPIC connection has been purged by a previous call of the exit program. In this case, your exit program can indicate that CICS is to continue processing normally, resuming queuing when necessary.

Exit-specific parameters

UEPISDATA

Address of the 78-byte area. This area is mapped by the DSECT in copybook DFHXIQDS.

Area addressed by UEPISDATA:

UEPREQ

A 2-byte origin-of-request code, which can have the following value:

AL Other kinds of intercommunication (for example, STARTs).

FS Function shipping and distributed program link

TR Transaction routing

UEPIPCNM

The 8-byte name of the IPCONN.

UEPREQTR

The 4-byte identifier of the requesting transaction.

UEPFLAG
A 1-byte flag indicating whether a return code 8 was issued the last time the exit was called.

UEPRC8
The exit program returned control to CICS on the previous call with return code 8.

UEPFSPL
Address of the 10-byte parameter list for the DPL request.

UEPCONST
Address of the 504-byte IPCONN statistics record. This record can be mapped using DSECT DFHISRDS.

UEPEMXQT
A halfword binary field containing the maximum queuing time, MAXQTIME, specified in the IPCONN resource definition.

UEPSAQTS
A double-word binary field containing the time stamp from the installed IPCONN resource definition, indicating the time that the queue of allocate requests was started.

UEPSACNT
A half-word binary field containing the number of allocate requests processed since the queue was started. See UEPSAQTS for the start time.

UEPSARC8
A half-word binary field containing the number of sessions freed since the queue was last purged as the result of an UEPCAKLL return code.

UEPQUELM
A half-word binary field containing the queue limit, QUEUELIMIT, specified in the IPCONN resource definition.

Return codes
In the case of an allocate that is about to be queued, use one of the following return codes:

UERCAQUE
Queue the allocate request.

UERCAKLL
Reject this allocate request with SYSIDERR. Purge all other queued allocate requests and send an information message to the operator console. CICS also returns SYSIDERR to all application programs waiting on the purged allocate requests.

UERCAPUR
Reject the allocate request with SYSIDERR.

UERCPURG
Task purged during XPI call.

In the case of a successful allocate following the use of UERCAKLL on a previous call of the exit, you can use a normal return code (UERCNORM) or use the SYSIDERR return code (UERCAPUR):

UERCNORM
Resume normal operation of the IPCONN.

UERCAPUR
Reject the allocate request with SYSIDERR.

XPI calls
All can be used.

Using an XISQUE global user exit program

When the exit is enabled, your XISQUE global user exit program is able to check on the state of allocate queues for IPCONNs in the local system. The parameter list passed to the exit program on invocation provides data about a specific allocate request and IPCONN.

Using the information passed in the parameter list, your global user exit program can decide (based on queue length, for example) whether CICS is to queue the allocate request. Your program communicates its decision to CICS by setting one of the following return codes:

UERCAQUE
CICS is to queue the allocate request.

The total number of allocate requests currently queued against the connection is provided in field ISR_CURRENT_QUEUED_ALLOCATES of the IPCONN statistics record, which is addressed by the UEPCONST exit-specific parameter. See DSECT DFHISRDS for details.

CICS also passes to the exit program, in field UEPQUELM, the value of the QUEUELIMIT option of the IPCONN resource definition. If the queue limit has not been reached, you can return control to CICS with return code UERCAQUE.

UERCAPUR
CICS is to reject the allocate request, return SYSIDERR to the application program, and leave the existing queue unchanged.

If the number of queued allocate requests has reached the limit set on the QUEUELIMIT option of the IPCONN definition, you can request that CICS rejects the request. However, you should first check whether the state of the link is satisfactory. This means checking that the rate of allocation of sessions is acceptable. Use the time the queue was started, the current time, and the total number of allocates processed since the queue began, to determine the rate at which CICS is processing requests. The relevant fields are: UEPSAQTS and UEPSACNT.

To determine whether CICS is allocating requests for sessions on this IPCONN at an acceptable rate, you can compare the calculated time with either of the following:

1. The value of the MAXQTIME option of the IPCONN resource definition, which is passed in the UEPEMXQT exit-specific parameter.
2. Some other preset time value.

If, using this kind of formula, you find the processing time to be acceptable, return control to CICS with return code UERCAPUR, which purges only this request.

UERCAKLL
Reject this request, purge all other allocate requests queued on this IPCONN, and send an information message to the operator console.

If the queue limit has been reached and the performance of allocate processing is below the acceptable limits defined in your user exit program, you can purge all queued allocate requests by specifying return code UERCAKLL.

UERCAKLL also causes CICS to:

- Return SYSIDERR to all application programs waiting on the purged allocate requests.
- On subsequent calls to your XISQUE exit program, set the UEPFLAG parameter to UEPRC8 to indicate that UERCAKLL was returned previously to purge the queue.

Purging a queue that is causing congestion in the flow of tasks frees task slots that are needed to prevent the system becoming clogged. The more you allow a session queue to grow, the more likely you are to reach the task ceiling set by the MAXT parameter, and then cause a queue of incoming tasks in the local region that cannot be attached.

If a queue has been purged previously (with UERCAKLL) but there are no queued requests currently, check the number of sessions freed since the queue was last purged. This number is in UEPSARC8. If no sessions have been freed on this IPCONNsince the queue was last purged, the problem that caused the purge previously has not been resolved, and this request should be rejected with UERCAPUR.

If the UEPSARC8 parameter shows that sessions are being freed, you should use UERCAQUE to resume queuing of requests. If you return with UERCAQUE in this case, CICS issues an information message to the console to signal that queuing has been resumed.

UERCNORM
CICS is to resume normal processing on the connection, including queuing of requests.

Use UERCNORM when the exit is invoked after a successful allocate following the suppression of queuing.

Statistics fields in DFHISRDS

The following fields in IPCONN statistics can help your XISQUE global user exit program to control allocate queues efficiently:

ISR_XISQUE_ALLOC_REJECTS
Each time an XISQUE global user exit program returns with a request to reject a request, CICS increments this field, which is provided to help you tune the queue limit. Normally, if the number of sessions and the queue limit specified on the IPCONN definition are correctly balanced, and there has been no abnormal congestion on the link, ISR_XISQUE_ALLOC_REJECTS should be zero. If the rejected allocates field is non-zero it indicates that action is probably needed.

ISR_XISQUE_ALLOC_QPURGES
Each time an XISQUE global user exit program returns with a request to purge a queue, CICS increments this field.

For detailed information about the fields in IPCONN statistics, see IPCONN report. IPCONN statistics are mapped by DSECT DFHISRDS.

Designing an XISQUE global user exit program

Your XISQUE exit program should be designed to:

1. Control of the number of tasks (and the amount of associated resource) that are waiting in a queue for a free IPIC session. Waiting tasks can degrade the performance of the local system.
2. Detect poor response from the remote system and notify the operator (or automatic operations program).
3. Cause CICS to issue a message when the IPCONN resumes normal operation.

The XISQUE parameter list is designed to support these objectives. The information it contains enables your exit program to:

- Avoid false diagnosis of connection problems by distinguishing poor response times from a complete bottleneck
- Ensure that a link resumes normal operation quickly and without operator intervention after a problem in a remote system is resolved

Using IPCONN statistics

In reaching its decisions about which requests to reject, which to queue, and which queues to purge, your exit program will probably take into account the number of allocate requests currently queued against the connection. All allocate requests for a particular IPCONN are queued in a single queue that is specific to that IPCONN. CICS makes the total number of entries in this queue available in the IPCONN statistics field ISR_CURRENT_QUEUED_ALLOCATES. Your exit program can access this field by means of the address of the IPCONN statistics, which is passed in the UEPCONST exit-specific parameter.

Sample XISQUE exit program

CICS provides a sample XISQUE exit program, DFH$XISQ, that you can use as the basis for your own program. It is supplied in the CICSTS54.CICS.SDFHSAMP library. The DSECT used by the sample program to map the area addressed by UEPISDATA is called DFHXIQDS, and is supplied in the CICSTS54.CICS.SDFHMAC library.

For more information about DFH$XISQ, see Session queue management sample exit program for IPIC connections: DFH$XISQ.

XRF request-processing program exit XXRSTAT

This exit is invoked if a z/OS Communications Server failure or a z/OS Communications Server predatory takeover occurs.

XXRSTAT enables you to decide whether to terminate CICS when either of the following occurs:

- CICS is notified of a z/OS Communications Server failure by the TPEND exit.
- A predatory takeover has occurred. Predatory takeover can occur if you are using Release 3.4.0 or later, and a z/OS Communications Server application with the same APPLID as that of the executing CICS system assumes control of all the sessions of the executing CICS system.

  XXRSTAT gives you the choice of allowing the system which has suffered the takeover to continue or to terminate.

  To avoid potential integrity exposures, CICS default action after a predatory takeover is to terminate without a dump. If you want CICS to terminate with a dump, your exit program should return UERCABDU. CICS terminates with the abend code specified by your exit program.

If you want CICS to continue after a predatory takeover, your exit program must return UERCCOIG. Message DFHZC0101 is issued and CICS continues processing without z/OS Communications Server support. The predatory application assumes control of all z/OS Communications Server sessions.

Note: Allowing CICS to continue after a predatory takeover could cause integrity problems and is not recommended. Use RACF to protect your CICS APPLIDs.

Exit XXRSTAT

This exit is invoked if a z/OS Communications Server failure or predatory takeover occurs.

When invoked

After either of the following:

- CICS is notified of a z/OS Communications Server failure by the TPEND exit.
- A predatory takeover.

Exit-specific parameters

UEPERRA

Address of parameter list containing:

UEPGAPLD

Address of the 8-byte generic applid

UEPSAPLD

Address of the 8-byte specific applid

UEPDOMID

Address of the 4-byte domain ID

UEPERRID

Address of the 4-byte error ID.

Note:

1. No DSECT is provided for this parameter list. You need to code your own DSECT to access the named fields.
2. When z/OS Communications Server has failed, the domain ID is 'ZC ' (uppercase Z, uppercase C, and two blanks), and the error ID is the character string '3443'.

Return codes

UERCNORM

Take the system action. The system action depends on the reason why the exit was invoked:

- For XRF, in the event of a z/OS Communications Server failure: CICS continues processing as if the exit program had not been invoked.
- For z/OS Communications Server persistent sessions, in the event of a predatory takeover: CICS abends without a dump.

UERCCOIG

Ignore.

UERCABNO

Abend CICS without a dump.

UERCABDU
Abend CICS with a dump.

UERCPURG
Task purged during XPI call.

XPI calls
All can be used.

Notices

This information was developed for products and services offered in the U.S.A. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property rights may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

© Copyright IBM Corp. 1977, 2017

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming interface information

CICS supplies some documentation that can be considered to be Programming Interfaces, and some documentation that cannot be considered to be a Programming Interface.

Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 4 are included in the following sections of the online product documentation:

- Developing applications
- Developing system programs
- RACF security overview
- Developing for external interfaces
- Reference: application developmenth
- Reference: system programming
- Reference: connectivity

Information that is NOT intended to be used as a Programming Interface of CICS Transaction Server for z/OS, Version 5 Release 4 , but that might be misconstrued as Programming Interfaces, is included in the following sections of the online product documentation:

- Troubleshooting and support
- Reference: diagnostics

If you access the CICS documentation in manuals in PDF format, Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 4 are included in the following manuals:

- Application Programming Guide and Application Programming Reference
- Business Transaction Services
- Customization Guide
- C++ OO Class Libraries
- Debugging Tools Interfaces Reference
- Distributed Transaction Programming Guide
- External Interfaces Guide
- Front End Programming Interface Guide
- IMS Database Control Guide
- Installation Guide
- Security Guide
- Supplied Transactions
- CICSPlex SM Managing Workloads
- CICSPlex SM Managing Resource Usage
- CICSPlex SM Application Programming Guide and Application Programming Reference
- Java Applications in CICS

If you access the CICS documentation in manuals in PDF format, information that is NOT intended to be used as a Programming Interface of CICS Transaction Server for z/OS, Version 5 Release 4 , but that might be misconstrued as Programming Interfaces, is included in the following manuals:

- Data Areas
- Diagnosis Reference
- Problem Determination Guide
- CICSPlex SM Problem Determination Guide

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability
These terms and conditions are in addition to any terms of use for the IBM website.

Personal use
You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use
You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM online privacy statement

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below:

For the CICSPlex SM Web User Interface (main interface):
Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user’s user name and other personally identifiable information for purposes of session management, authentication, enhanced user usability, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface (data interface):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's user name and other personally identifiable information for purposes of session management, authentication, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface ("hello world" page):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect no personally identifiable information. These cookies cannot be disabled.

For CICS Explorer:
Depending upon the configurations deployed, this Software Offering may use session and persistent preferences that collect each user’s user name and password, for purposes of session management, authentication, and single sign-on configuration. These preferences cannot be disabled, although storing a user's password on disk in encrypted form can only be enabled by the user's explicit action to check a check box during sign-on.

If the configurations deployed for this Software Offering provide you, as customer, the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM Privacy Policy and IBM Online Privacy Statement, the section entitled “Cookies, Web Beacons and Other Technologies” and the IBM Software Products and Software-as-a-Service Privacy Statement.

Index

A

AILDELAY, system initialization parameter 204
allocate queues
  controlling the length of
    using the XISCONA global user exit 131
    using the XISQUE global user exit 269
    using the XZIQUE global user exit 262
Alphabetical list of GLUEs 1
APPC connections
  intersystem queues 262
  XZIQUE global user exit
    for controlling intersystem queues 262

B

Basic Mapping Support (BMS)
  global user exit points 11

C

CICS web support
  User exits XWBAUTH, XWBOPEN, XWBSNDO 125
  User exits XWBOPEN, XWBSNDO 128, 130
coexistence of local DL/I and DBCTL
  XDLIPRE to change PSB to be scheduled 30

D

data tables 17
DFH$ICCN, sample global user exit program 157
DFH$WBGA, sample global user exit program 125

E

Exit XTSPTOUT 224
Exit XTSQRIN 220

F

Functional list of GLUEs 1

G

global user exits
  example programs
    for modifying TS requests 234
    for XFCREQ 81, 84
    for XFCREQC 81, 84
global user exits *(continued)*
  example programs *(continued)*
    for XICEREQ 157
    for XICEREQC 157
    for XPCREQ 194
    for XPCREQC 194
    for XTDEREQ 248
    for XTDEREQC 248
    for XTSEREQ 234
    for XTSEREQC 234
  exit points
    bridge facility creation 15
    bridge facility deletion 15
    for 'terminal not known' condition 238
    in activity keypoint program 9
    in application association data 10
    in BMS 11
    in CICS Web support 125
    in data tables management 17
    in data tables programs 17
    in DBCTL interface control program 22
    in DBCTL tracking program 23
    in dispatcher domain 24
    in DL/I interface program 26
    in dump domain 36
    in enqueue EXEC interface program 43
    in EXEC interface program 50
    in file control domain 59
    in file control EXEC interface program 72, 84
    in file control file state program 98
    in file control open/close program 108
    in file control quiesce receive program 110
    in file control quiesce send program 112
    in file control recovery program 113
    in file control RLS coexistence 122
    in Front End Programming Interface 55
    in good-morning message program 124
    in intersystem communication program 131
    in interval control EXEC interface program 139
    in interval control program 137
    in loader domain 157
    in log manager domain 160
    in message domain 163
    in monitoring domain 167
    in pipeline domain 170
    in program control program 184
global user exits *(continued)*
  exit points *(continued)*
    in resource management modules 204
    in resource manager interface program 202
    in security manager domain 211
    in SNA LU management program 260
    in SNA working-set module 260
    in statistics domain 214
    in system recovery program 216
    in system termination program 219
    in temporary storage domain 220
    in temporary storage EXEC interface program 225
    in terminal allocation program 235
    in terminal control program 236
    in transaction manager domain 244
    in transient data EXEC interface program 248
    in transient data program 246
    in user log record recovery program 256
    in XRF request-processing program 274
  sample programs
    DFH$ICCN 157
    DFH$WBGA 125
  XDLIPRE
    example 30
  XSZARQ
    exit-specific parameters 57
    overview 57
    UEPSZACN parameter 58
  XSZBRQ
    overview 56
    UEPSZACT parameter 58
GLUEs, alphabetical list 1
GLUEs, functional list 1

H

HTTP client open exit XWBOPEN 128

I

intersystem queues
  controlling the length of
    using the XISCONA global user exit 131
    using the XZIQUE global user exit 262
IPCONNs
  intersystem queues 269

© Copyright IBM Corp. 1977, 2017

IPCONNs *(continued)*
XISQUE global user exit
for controlling intersystem queues 269
ISC over TCP/IP intersystem queues
controlling the length of
using the XISQUE global user exit 269
ISC over TCP/IPCONNs
intersystem queues 269
XISQUE global user exit
for controlling intersystem queues 269

M

MRO connections
intersystem queues 262
XZIQUE global user exit
for controlling intersystem queues 262

P

PSB (program specification block)
XDLIPRE to change PSB to be scheduled 30

Q

queues for intersystem sessions
controlling the length of
using the XISCONA global user exit 131
using the XISQUE global user exit 269
using the XZIQUE global user exit 262

S

sample programs
for global user exits
DFH$WBGA, for the XWBOPEN exit 125
system initialization parameters
AILDELAY 204
TBEXITS 114, 258

T

TBEXITS, system initialization parameter 114, 258

U

UEPSZACN, exit-specific parameter for XSZARQ 58
UEPSZACT, exit-specific parameter for XSZBRQ 58

X

XALCAID, global user exit 235
XALTENF, global user exit 240
XAPADMGR, global user exit 10
XBMIN, global user exit 12
XBMOUT, global user exit 12
XDLIPOST, global user exit 28
XDLIPRE, global user exit 26
to change PSB to be scheduled 30
XDSAWT, global user exit 25
XDSBWT, global user exit 25
XDTAD, global user exit 20
XDTLC, global user exit 21
XDTRD, global user exit 18
XDUCLSE, global user exit 42
XDUOUT, global user exit 42
XDUREQ, global user exit 36
XDUREQC, global user exit 39
XEIIN, global user exit 52
XEIOUT, global user exit 53
XEISPIN, global user exit 53
XEISPOUT, global user exit 54
XEPCAP, global user exit 50
XFAINTU, global user exit 15
XFCAREQ, global user exit
description 84
parameter list and return codes 85
XFCAREQC, global user exit
description 84
parameter list and return codes 86
XFCBFAIL, global user exit 114
XFCBOUT, global user exit 118
XFCBOVER, global user exit 119, 120
XFCFRIN, global user exit
parameter list and return codes 60
XFCFROUT, global user exit
parameter list and return codes 66
XFCLDEL, global user exit 121
XFCNREC, global user exit
description 108
parameter list and return codes 108
XFCQUIS, global user exit
description 112
XFCREQ, global user exit
command parameter structure 73
description 72
example of use 81
parameter list and return codes 81
UEPCLPS parameter 74
XFCREQC, global user exit
command parameter structure 73
description 72
example of use 81
parameter list and return codes 83
UEPCLPS parameter 74
XFCRLSCO, global user exit
description 122
XFCSREQ, global user exit 100
XFCSREQC, global user exit 103
XFCVSDS, global user exit
description 110
XGMTEXT, global user exit 124
XICEREQ, global user exit
command parameter structure 146
example of use 157
parameter list and return codes 141
UEPCLPS parameter 147
XICEREQC, global user exit
command parameter structure 146
example of use 157
parameter list and return codes 144
UEPCLPS parameter 147
XICERES, global user exit
parameter list and return codes 143
XICEXP, global user exit 139
XICREQ, global user exit 138
XICTENF, global user exit 242
XISCONA, global user exit 131, 133
XISLCLQ, global user exit 131, 135
XISQLCL, global user exit 131, 136
XISQUE, global user exit 269, 270
designing the exit program 273
how to use 272
overview 269
XLDELETE, global user exit 159
XLDLOAD, global user exit 158
XLGSTRM, global user exit 160
XMEOUT 163
XMEOUT, global user exit 165
XMNOUT, global user exit 167
XNQEREQ, global user exit 44
command parameter structure 46
UEPCLPS parameter 46
XNQEREQC, global user exit 45
command parameter structure 46
UEPCLPS parameter 46
XPCABND, global user exit 201
XPCERES, global user exit
description 184
parameter list and return codes 187
XPCFTCH, global user exit 195
XPCHAIR, global user exit 197
XPCREQ, global user exit
command parameter structure 189
description 184
example of use 194
parameter list and return codes 185
UEPCLPS parameter 190
XPCREQC, global user exit
command parameter structure 189
description 185
example of use 194
parameter list and return codes 188
UEPCLPS parameter 190
XPCTA, global user exit 198
XRCINIT, global user exit 258
XRCINPT, global user exit 259
XRMIOUT, global user exit 203
XRMMI, global user exit 202
XRSINDI 205
XRSINDI, global user exit 205
XSNEX, global user exit 214
XSNOFF, global user exit 213
XSNON, global user exit 211
XSRAB, global user exit 216
XSTERM, global user exit 219
XSTOUT, global user exit 214
XSZARQ, global user exit 55
overview 57
UEPSZACN parameter 58
XSZBRQ, global user exit 55
overview 56
UEPSZACT parameter 58
XTCATT, global user exit 237

XTCIN, global user exit 236
XTCOUT, global user exit 237
XTDEREQ 248
XTDEREQ, global user exit
  command parameter structure 252
  parameter list and return codes 249
  UEPCLPS parameter 253
XTDEREQC 248
XTDEREQC, global user exit
  command parameter structure 252
  parameter list and return codes 250
  UEPCLPS parameter 253
XTDIN 246
XTDIN, global user exit 247
XTDOUT 246
XTDOUT, global user exit 247
XTDREQ 246
XTDREQ, global user exit 246
XTSEREQ, global user exit 226
  command parameter structure 229
  example program 234
  UEPCLPS parameter 229
XTSEREQC, global user exit 227
  command parameter structure 229
  example program 234
  UEPCLPS parameter 229
XTSPTIN global user exit 223
XTSQROUT, global user exit 222
XWBAUTH user exit 125
XWBAUTH, global user exit 125
XWBOPEN user exit 128
XWBOPEN, global user exit 125
XWBSNDO user exit 130
XWBSNDO, global user exit 125
XWSPRROI, global user exit 172
XWSPRROO, global user exit 173
XWSPRRWI, global user exit 171
XWSPRRWO, global user exit 174
XWSRQROI, global user exit 178
XWSRQROO, global user exit 177
XWSRQRWI, global user exit 179
XWSRQRWO, global user exit 176
XWSSRROI, global user exit 182
XWSSRROO, global user exit 181
XWSSRRWI, global user exit 183
XWSSRRWO, global user exit 180
XXDFA, global user exit 22
XXDFB, global user exit 23
XXDTO, global user exit 24
XXMATT, global user exit 244
XXRSTAT, global user exit 275
XZCATT, global user exit 260
XZCIN, global user exit 260
XZCOUT, global user exit 261
XZCOUT1, global user exit 262
XZIQUE, global user exit 262, 266
  designing your exit program 269
  how to use 263
  interaction with XISCONA 263
  overview 262
  using IRC/ISC statistics 269
  when invoked 263

