

CICS Transaction Server for z/OS
Version 4 Release 1



Front End Programming Interface User's Guide

CICS Transaction Server for z/OS
Version 4 Release 1



Front End Programming Interface User's Guide

Note

Before using this information and the product it supports, read the information in "Notices" on page 265.

This edition applies to Version 4 Release 1 of CICS Transaction Server for z/OS (product number 5655-S97) and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright IBM Corporation 1992, 2011.**

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

Contents

Preface	vii
What this manual is about	vii
Who this manual is for	vii
What you need to know to understand this manual	vii
Notes on terminology	vii
CICS syntax notation	viii

Changes in CICS Transaction Server for z/OS, Version 4 Release 1 xi

Part 1. FEPI concepts and facilities 1

Chapter 1. Introducing FEPI	3
Problems FEPI can solve	3
How FEPI fits into your system	4

Chapter 2. Planning to use the Front End Programming Interface	7
Hardware and software requirements	7
System integrity	7
Storage	7
Installation	8

Chapter 3. Configuring your system for FEPI	9
--	----------

Chapter 4. FEPI functions and services 11	
Introducing FEPI functions	11
Samples	12
FEPI programming commands	12
High-level FEPI commands	12
Data-stream-level commands	13
Specialized-level commands	13
List of commands	13
Setup and resources	14
FEPI resources	14
CICS FEPI application programs	15
Terminals supported	16
FEPI Security	16
Signon security	16
Command security	17
Problem determination, customization, and performance	17

Part 2. FEPI installation and administration 19

Chapter 5. Planning for FEPI	21
Analysis and planning.	21
Back-end applications and systems	21
Names of nodes and targets	21
Operator control requirements	22

Journaling requirements	22
Signon and signoff procedures	22
Special event handling.	22
Using pools for control reasons.	23
Using pools for functional reasons.	23
Number of nodes	23
Setup program organization	24
Organizing your pools and property sets	24
Organizing pools	24
Organizing property sets	24
Workload balancing in a sysplex	26
Planning FEPI storage	26

Chapter 6. Getting started with FEPI 29	
The installation process	29
A note about loading FEPI modules into the LPA	29
Updating CICS definitions	29
Installing FEPI resource definitions	30
Starting CICS.	30

Chapter 7. Configuring FEPI 33	
Configuring CICS	33
Configuring VTAM.	34
Availability of network resources	35
Selection of FEPI session parameters	35
Pacing of FEPI sessions	36
Configuring the back-end systems.	37
The configuration programs you should write	38
Writing configuration programs	38
Writing setup programs	40
Running setup programs	41
Varying the resources installed by the setup program	41
An example FEPI configuration.	42
Writing monitoring programs	46
Handling unexpected events.	46
Handling CLSDST(PASS).	48
Writing operator transactions	50
Other functions	50
Global user exit programs	50

Chapter 8. FEPI operation 53	
Controlling FEPI resources	53
SERVSTATUS.	53
ACQSTATUS	53
LASTACQCODE	55
INSTLSTATUS	55
WAITCONVNUM	55
STATE	55
FEPI performance	55
Using CICS monitoring	56
Using statistics data	56
Shutdown	57
Normal shutdown	57
Immediate shutdown	58

Forced shutdown	58
Using FEPI with XRF	58
XRF and VTAM	58
FEPI resource definition and XRF	59
XRF takeover of front-end system	59
XRF takeover of back-end system	60
Using FEPI with VTAM persistent sessions	62
Restart of front-end system using persistent sessions	62
Restart of back-end system using persistent sessions	62

Chapter 9. Operator control of FEPI . . . 65

CEMT - master terminal transaction	65
CEMT DISCARD	65
CEMT INQUIRE FECONNECTION	66
CEMT INQUIRE FENODE	69
CEMT INQUIRE FEPOOL	71
CEMT INQUIRE FEPROPSET	73
CEMT INQUIRE FETARGET	73
CEMT SET FECONNECTION	75
CEMT SET FENODE	76
CEMT SET FEPOOL	77
CEMT SET FETARGET	77
VTAM commands	78

Chapter 10. Customizing FEPI. 81

Front End Programming Interface exits XSZARQ and XSZBRQ	81
XSZBRQ	81
XSZARQ	83
The UEPSZACT and UEPSZACN exit-specific parameters	83
Using XMEOUT to control message output.	84
FEPI journaling	85
FEPI journal operation.	85
Printing FEPI journal records	86

Chapter 11. FEPI system programming reference. 89

The FEPI SPI commands	89
Command format	90
Errors and exception conditions	90
FEPI ADD POOL	91
FEPI DELETE POOL	93
FEPI DISCARD NODELIST	94
FEPI DISCARD POOL.	94
FEPI DISCARD PROPERTYSET	95
FEPI DISCARD TARGETLIST	95
FEPI INQUIRE CONNECTION.	96
FEPI INQUIRE NODE.	99
FEPI INQUIRE POOL	101
FEPI INQUIRE PROPERTYSET	105
FEPI INQUIRE TARGET	108
FEPI INSTALL NODELIST	109
FEPI INSTALL POOL.	111
FEPI INSTALL PROPERTYSET	113
FEPI INSTALL TARGETLIST	117
FEPI SET CONNECTION	118
FEPI SET NODE	120

FEPI SET POOL	121
FEPI SET TARGET	122
FEPI SP NOOP.	123
Transient data queue records	123
Fields	124

Chapter 12. FEPI problem determination 127

Debugging FEPI applications	127
FEPI dump	127
Using CICS dump facilities to investigate FEPI problems	129
FEPI trace	130
Taking and interpreting trace entries	131
FEPI messages	131
FEPI abends.	132
Restart	132
Message DFHSZ4099E	132
Message DFHSZ4155I	133
Reporting a FEPI problem to IBM	133

Part 3. FEPI application programming. 135

Chapter 13. Basics of FEPI programming 137

Communication and conversations	137
Structure and design	138
Programming	139

Chapter 14. FEPI key stroke and screen-image applications. 141

General sequence of commands	141
Sending key stroke data	142
Error handling	143
Receiving field-by-field	144
Command completion	144
Error handling	145
Multiple attentions	145
Sending screen-image data	146
Receiving screen-image data	147
Command completion and errors.	148
Extracting field data	148
CONVERSE	148

Chapter 15. FEPI data stream applications 149

When to use the data stream interface	149
General sequence of commands	150
Receiving.	150
Command completion	151
Error handling	153
Sending	153
CONVERSE	153
SLU2 mode considerations	154
SLU P mode considerations	155

Chapter 16. FEPI application design 157

Programs	157
Access program	157
Begin-session handler	158
Unsolicited-data handler	158
End-session handler	159
Application organization	159
Application style	159
Started tasks.	160
Conversations	161
Signon security	164
How to use PassTickets	164
Benefits	165
Requirements	165
Error handling	165
Time-outs	166
Lost session	166
Previous SEND failed	166
Communication errors	167
Bypass by user exit	167
Unknown conversation ID	167
Operator/system action	167
Shutdown	167
System considerations	168
IMS considerations	168
Performance.	171

Chapter 17. Specialized FEPI functions	173
Set and test sequence number (STSN)	173
DRx responses	174
SNA commands	174

Part 4. FEPI application programming reference 175

Chapter 18. The FEPI API commands 177	
Command format	178
Errors and exception conditions	178
Chapter 19. FEPI ALLOCATE PASSCONVID 181	
Chapter 20. FEPI ALLOCATE POOL 183	
Chapter 21. FEPI AP NOOP 185	
Chapter 22. FEPI CONVERSE DATASTREAM 187	
Chapter 23. FEPI CONVERSE FORMATTED. 193	
Chapter 24. FEPI EXTRACT CONV . . 199	
Chapter 25. FEPI EXTRACT FIELD . . 201	
Chapter 26. FEPI EXTRACT STSN . . 205	

Chapter 27. FEPI FREE 207	
Chapter 28. FEPI ISSUE 209	
Chapter 29. FEPI RECEIVE DATASTREAM 213	
Chapter 30. FEPI RECEIVE FORMATTED. 217	
Chapter 31. FEPI REQUEST PASSTICKET 221	
Chapter 32. FEPI SEND DATASTREAM 223	
Chapter 33. FEPI SEND FORMATTED 225	
Chapter 34. FEPI START. 227	
Chapter 35. Start data. 229	
Fields	229
Chapter 36. Data formats 231	
Chapter 37. Ending status 233	

Part 5. Appendixes 235

Appendix A. FEPI sample programs 237	
List of samples	237
VS COBOL II Sample Restrictions	239
Installing the samples	239
Using the samples.	240
The back-end CICS program	240
The back-end IMS program.	241
Description of the samples	242
Setup program	242
Monitor and unsolicited data-handler	243
Begin session	245
Key stroke CONVERSE	246
Screen image SEND and START	247
Screen image RECEIVE and EXTRACT FIELD	249
3270 data stream passthrough	250
End-session handler	250
SLU P one-out one-in.	251
SLU P pseudoconversational	252
STSN handler	254

Appendix B. CVDA and RESP2 values for FEPI commands. 255	
FEPI CVDAs and numeric values in alphabetic sequence	255
FEPI CVDAs and numeric values in numeric sequence	257
FEPI RESP2 values	260

Notices 265
Trademarks 266

Bibliography. 267
CICS books for CICS Transaction Server for z/OS 267
CICSplex SM books for CICS Transaction Server
for z/OS 268

Other CICS publications. 268

Accessibility. 269

Index 271

Preface

What this manual is about

This manual documents intended Programming Interfaces that allow the customer to write programs to obtain the services of Version 4 Release 1.

This manual describes the Front End Programming Interface (FEPI) of CICS® Transaction Server for z/OS®, Version 3 Release 2.

Who this manual is for

This manual is intended primarily for CICS system programmers and administrators responsible for installing and configuring FEPI, and for application programmers responsible for writing FEPI “front-end” application programs.

What you need to know to understand this manual

To configure FEPI, you need to be familiar with all aspects of CICS administration (such as system definition, resource definition, customization, and operations) and the programming interface to CICS. For information about CICS system definition, see the *CICS System Definition Guide*. For information about defining resources to CICS, see the *CICS Resource Definition Guide*. For programming information about customizing CICS, see the *CICS Customization Guide*. For programming information about EXEC CICS commands, see the *CICS Application Programming Reference* and the *CICS System Programming Reference*. You should also be familiar with the IBM® ACF/VTAM telecommunication access method and, if you are accessing IMS™ back-end systems, with IBM IMS administration.

To write FEPI “front-end” applications, you need to know how to write programs in at least one of the programming languages that CICS supports. More importantly, you also need knowledge of data communication and protocols. And, if you will be accessing IMS back-end systems, you must also be familiar with using IMS and writing IMS applications.

Notes on terminology

In this manual, VTAM® refers to ACF/VTAM. The term “CICS”, without any qualification, refers to the CICS element of IBM CICS Transaction Server for z/OS.

CICS Transaction Server for z/OS, Version 4 Release 1 supports CICS applications written in:

- Assembler language
- C
- C++
- COBOL
- PL/I

In this book, the phrase “the languages supported by CICS” refers to the above languages.

KB equals 1024 bytes; MB equals 1024 KB.

The following terms have different meanings for FEPI, CICS, IMS, and VTAM:

application

FEPI uses application in the normal sense of a program or suite of programs that do work. VTAM uses application for programs that communicate directly using VTAM; in a FEPI environment, this means the back-end systems on one hand, and FEPI on the other.

conversation

A FEPI conversation is not the same as an IMS conversation, although they would normally coincide, and it is not related to CICS conversational mode. It is analogous to a CICS APPC conversation.

inbound, input

In FEPI and CICS usage, these describe data received by a program from elsewhere. From the point-of-view of the back-end system, this data is outbound or output to a terminal.

message

VTAM and IMS use message to refer to any data transmission, and not just to data displayed for a user's attention.

node

In VTAM and IMS, a node is a named point in a network. In FEPI, nodes are those points (VTAM nodes) that are the secondary LU terminals simulated by FEPI.

outbound, output

In FEPI and CICS usage, these describe data sent by a program to somewhere else. From the point-of-view of the back-end system, this data is inbound or input from a terminal.

secondary

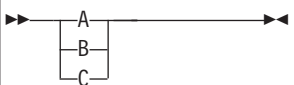
In VTAM, secondary describes one of the partners of an LU-LU pair; the terminals simulated by FEPI are secondary LUs. This is not the same as the CICS usage of secondary.

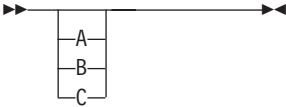
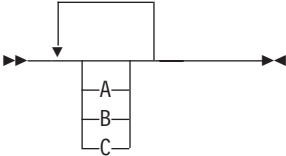
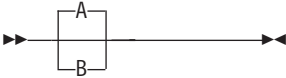
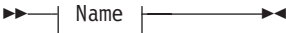
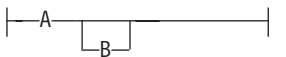
CICS syntax notation

The syntax of CICS commands is presented in a standard way.

The **EXEC CICS** that always precedes each command's keyword is not included; nor is the "END_EXEC" statement used in COBOL or the semicolon (;) used in PL/I and C that you must code at the end of each CICS command. In the C language, a null character can be used as an end-of-string marker, but CICS does not recognize this; you must never, therefore, have a comma or period followed by a space (X'40') in the middle of a coding line.

You interpret the syntax by following the arrows from left to right. The conventions are:

Symbol	Action
	A set of alternatives—one of which you <i>must</i> code.

Symbol	Action
	A set of alternatives—one of which you <i>may</i> code.
	A set of alternatives—any of which you may code.
	Alternatives where A is the default.
 <p>Name:</p> 	Use with the named section in place of its name.
Punctuation and uppercase characters	Code exactly as shown.
Lowercase characters	Code your own text, as appropriate (for example, name).

Changes in CICS Transaction Server for z/OS, Version 4 Release 1

For information about changes that have been made in this release, please refer to *What's New* in the information center, or the following publications:

- *CICS Transaction Server for z/OS What's New*
- *CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.2*
- *CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.1*
- *CICS Transaction Server for z/OS Upgrading from CICS TS Version 2.3*

Any technical changes that are made to the text after release are indicated by a vertical bar (|) to the left of each new or changed line of information.

Part 1. FEPI concepts and facilities

This part of the book gives an overview of FEPI, and some general information about functions, services, and implementing applications.

- Chapter 1, “Introducing FEPI,” on page 3 explains what FEPI is and what problems it solves; it also describes some planning considerations.
- Chapter 4, “FEPI functions and services,” on page 11 describes the various types of FEPI commands and introduces the concepts and functions used by FEPI applications.

Chapter 1. Introducing FEPI

The Front End Programming Interface is an integral part of CICS. The function is called a front-end programming interface because it enables you to write CICS application programs that access other CICS or IMS programs. In other words, it provides a front end to those programs. The interface simulates the terminals that the other programs use.

This section contains the following topics:

- “Problems FEPI can solve”
- “How FEPI fits into your system” on page 4
- Chapter 2, “Planning to use the Front End Programming Interface,” on page 7.

Problems FEPI can solve

Many users have CICS and IMS applications that they want to use differently; for example, to extend their use by incorporating them into other applications. But they cannot change the way the applications are used because they cannot change the application programs.

FEPI allows existing CICS and IMS application programs to be used in different ways, in different combinations, in different environments, and on different systems, without changing them, because it provides a simple integrated interface to these programs. FEPI also lets you write new programs that add function to old programs.

There are many reasons why existing application programs can't be changed. Perhaps the application was bought in a package, so that you don't have the source. Perhaps someone else owns the application; perhaps it runs on someone else's system. Perhaps the source has been lost, and there's no one around who knows the program well enough. Perhaps the program logic is so complex that any changes are considered too dangerous.

Or perhaps it is an application that was written for one specific environment, such as IBM 3270 information display systems, and you want to use it for another, or you want to extend its function. You don't want to change the application, because it must still work with the 3270s.

To get around this, you can run the existing application unchanged and provide a front-end program to interface to it. Using FEPI, a front-end program can simulate a terminal. This means the program can gain access to applications written to support that terminal. That program can then use the existing applications, and the existing application is unaware that anything has changed.

Therefore, the existing application can be used differently without being changed in any way. The changes are in the simulating program. For example, newly written applications can collect data from several existing applications. The existing applications can be on the same system as the simulating program, or on a different system.

Advantages over alternative solutions

There are other ways of accessing existing programs differently, but they all have their drawbacks.

Can CICS multiregion operation (MRO) or intersystem communication (ISC) be used to access remote applications?

Yes, but using MRO or ISC often requires some changes to the existing application—for example, to change the type of terminal supported or to provide an interface that uses a communication area.

Can VTAM program-to-program support be used?

Yes, if your programmers can write an access program to issue the appropriate VTAM calls. But these VTAM calls cannot be part of a CICS application program.

How FEPI fits into your system

FEPI allows CICS front-end application programs to communicate with unchanged back-end applications running on CICS or IMS systems that are local or remote. The back-end applications continue to work just as if they are being accessed from the type of terminal they were originally written for.

A *FEPI application* is a CICS application that is designed to use FEPI to communicate with existing back-end applications. It is also known as a terminal front-end program.

The *front end* is the system on which the FEPI application runs, and the *back end* is the system on which the existing application runs. (They can run in the same CICS region.)

Figure 1 on page 5 shows the relationship between FEPI and other components of your system. Note, particularly, the unchanged applications in the lower part of the figure, and the new CICS FEPI application near the top. To an existing application, the front-end application looks like a terminal.

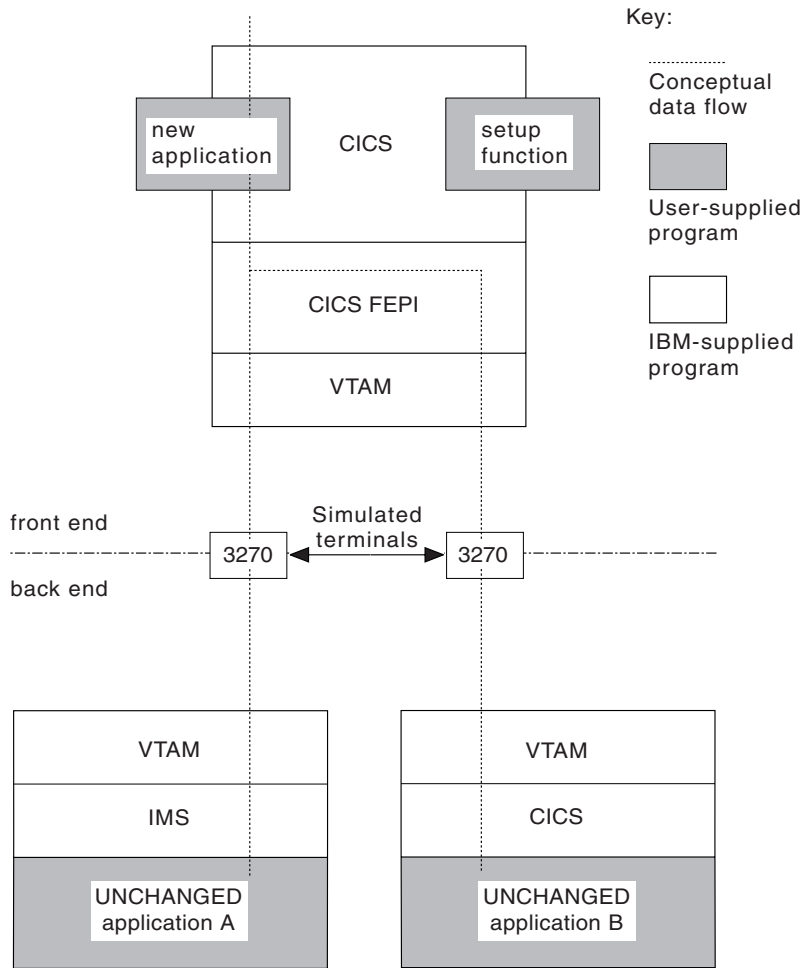


Figure 1. Structure of FEPI and application programs

Chapter 2. Planning to use the Front End Programming Interface

This section explains what hardware and software you need to use the Front End Programming Interface (FEPI), what MVS™ system integrity is involved, what resources you need, and what to consider when installing FEPI and customizing your system.

Hardware and software requirements

There are different requirements for the front-end and the back-end.

Front-end requirements

For front-end systems, FEPI is an integral part of CICS Transaction Server for z/OS. Other hardware and software requirements are the same as for CICS Transaction Server for z/OS.

Extra 37x5 controllers and network control programs (NCPs) may be needed to provide the necessary intersystem connections.

Back-end requirements

Applications running on the following, and subsequent compatible releases, are supported:

- CICS Transaction Server for z/OS
- IMS Version 9.1 or higher

FEPI provides simulation for two very common classes of terminals on these systems:

- 3270-types for CICS and IMS applications (using LU 2 protocol)
- A family of programmable terminals, including the 4700, accessed through an LU 0 protocol (called SLU P), for IMS applications.

System integrity

All application programs that use FEPI run in problem-program mode in user-key storage. No part of FEPI needs to be authorized to run.

IBM accepts authorized program analysis reports (APARs) where the installation of the FEPI function introduces an exposure to the system integrity of MVS. Refer to the *MVS Integrity Programming Announcement* dated 21 October 1981.

Storage

Some storage below the 16 MB line is required, but the bulk resides above the 16 MB line in storage managed by CICS.

For details, see “Planning FEPI storage” on page 26.

There are no inherent resource limits in FEPI. It is limited only by what is configured and the available system storage.

Installation

FEPI is distributed through normal IBM Program Library channels.

It is a part of CICS and cannot be ordered separately. See Chapter 6, "Getting started with FEPI," on page 29 for more information.

Chapter 3. Configuring your system for FEPI

You need to configure your system specifically for CICS FEPI, for new application programs, and possibly for existing applications.

About this task

You might need to adapt your VTAM setup, your CICS system, and CICS FEPI to use the interface effectively.

Procedure

1. Change the default value of the **FEPI** system initialization parameter to YES (the default is NO). The **FEPI** system initialization parameter controls whether FEPI is available or not. When the CICS region starts, FEPI runs as a system transaction that is started automatically. You do not need to be start or stop it independently.
2. Write a setup program to configure FEPI with the necessary resources when commands are issued from a front-end application program. FEPI does not use a configuration file or CICS RDO. The setup program can get the configuration data from a file or from whatever source it identifies.
3. Define CICS FEPI applications to CICS in the normal way.
4. Optional: Define simulated terminals for FEPI to use.

Chapter 4. FEPI functions and services

The Front End Programming Interface (FEPI) function provides access, by means of simulated terminals, to CICS and IMS applications available through a communication network.

The functions and services provided by FEPI enable you to write application program to provide a front end to other CICS and IMS applications without having to change those applications in any way. This is done by simulating a terminal in session.

Introducing FEPI functions

An application program using FEPI can provide a front end to other CICS or IMS applications. Because this is done by simulating a terminal in session with the non-FEPI application, that application does not have to be changed in any way.

Thus you can write FEPI applications that provide a single integrated interface to previously disparate applications. The scope and usability of your CICS and IMS applications can be extended by using them in combination, in different environments, or on different systems.

Because a FEPI application communicates with other applications that can run in different systems, it is necessary to distinguish between systems and identify the direction of data flows. The convention is:

Front-end

The front-end system is the one in which the FEPI application runs.

Back-end

Back-end identifies the system in which the other CICS or IMS applications run. (This is equivalent to “partner” system, used elsewhere by CICS.)

Outbound

Identifies data sent by the FEPI application to the back-end application.

Inbound

Identifies data received by the FEPI application from the back-end application.

FEPI provides a programming interface. Its functions can be invoked only through that interface, which is an extension to the EXEC CICS programming interface. All FEPI requests are made by issuing **EXEC CICS FEPI** commands; all the commands have the qualifier FEPI. The languages supported by the EXEC CICS programming interface (Assembler, COBOL, C, PL/I) can be used. For educational and initial development purposes, you could use CECI, rather than formally writing a program.

All functions are available in the normal way to all applications, except that some functions are intended for system programmers, and their use can be restricted. All the other facilities that you can use with CICS applications, such as the execution diagnostic facility (EDF) and the command interpreter transaction, CECI, are available.

Samples

To help you develop your own CICS FEPI applications, and to show you what FEPI can do, FEPI includes detailed samples. They form an integrated set, and include a program that sets up the FEPI configuration needed to run the other samples.

The samples are supplied in source format in the SDFHSAMP library, and include two back-end application programs, that show many of the principles and techniques discussed in the FEPI programming section. Although the samples are copyrighted, you may use and copy them freely for educational purposes to help you write FEPI applications.

The names of the samples have the form DFH0xZyy. Z shows that the sample is a FEPI sample and x identifies the source language of the sample. A for Assembler language, C for C, P for PL/I, and V for COBOL. yy identifies the specific program.

FEPI programming commands

EXEC CICS FEPI commands provide several ways of developing CICS FEPI applications.

The commands are at three logical levels:

High-level:

a straightforward interface for normal 3270 applications

Data stream-level:

for use with IMS SLU P applications and more complicated 3270 applications

Specialized-level:

for access to complex VTAM communication functions and events, designed for use by vendors and experienced CICS FEPI application developers.

High-level FEPI commands

The high-level front-end programming interface consists of two interfaces for everyday use: *key stroke* and *screen-image*, collectively known as *formatted* data. They allow programmers to build their own CICS FEPI applications in a straightforward manner. However, the programmer must understand data communication and protocols.

See Chapter 14, "FEPI key stroke and screen-image applications," on page 141 for details.

The key stroke interface

The key stroke interface allows programmers writing in any of the CICS-supported languages, to specify the keys that an operator might press while using an existing application. The key strokes are specified using easily coded mnemonics; no hexadecimal values are required.

The screen-image interface

The screen-image interface allows programmers writing in any language supported by CICS, to define the contents of a 3270 screen, using a data structure appropriate to the programming language.

It uses a buffer with one byte for each screen position (for example, 1920 bytes for a 24 × 80 character screen). This buffer can be defined in any way that suits the application program and the programming language. It is passed as a complete screen buffer to the back-end application.

In both cases, key stroke and screen-image, the data received from the back-end application is presented as a screen image.

Data-stream-level commands

For many applications, the key stroke and screen-image interfaces should be quite adequate.

However, where they are not, FEPI data-stream-level commands give an application complete control of the 3270 data stream. These commands are also needed for SLU P applications, which can use only this interface. FEPI does not buffer or interpret the data stream; it is presented as it arrives from the back-end application, and the front-end application must be prepared to handle whatever is presented. Similarly, data sent by the front-end application is transmitted without verification.

A detailed knowledge of data communication and protocols and of data stream format is required.

See Chapter 15, “FEPI data stream applications,” on page 149 for details.

Specialized-level commands

A number of specialized functions can be accessed through FEPI.

These specialized functions are as follows:

STSN for SLU P applications:

Set and test sequence number (STSN) is a communication protocol used to check and control transmissions. FEPI normally handles all necessary STSN processing automatically. However, FEPI also provides access to STSN information for those applications that need to control sequence number data.

Application access to definite responses:

When a flow is received, the receiving LU can choose what response to return to the sending LU. FEPI normally handles this automatically, but also provides facilities for applications to determine this flow.

Other VTAM facilities:

Some applications use a VTAM facility known as CLSDST(PASS); this can be used in more sophisticated CICS FEPI application programming.

See Chapter 17, “Specialized FEPI functions,” on page 173 for details.

List of commands

All the logical levels use more or less the same set of commands, though the options used may vary.

The EXEC CICS FEPI application programming commands are:

ALLOCATE

Establishes communication with a back-end application

FREE Frees communication with a back-end application

SEND Sends data from a CICS FEPI application to a back-end application

RECEIVE
Receives data into a CICS FEPI application from a back-end application

CONVERSE
Sends data to and receives data from a back-end application

ISSUE Sends control data to a back-end application

EXTRACT
Gets field data and attributes, set-and-test-sequence-number (STSN) data, or conversation status

START
Schedules a CICS transaction to handle inbound data.

Setup and resources

Besides the application programming functions that communicate with back-end applications, FEPI also provides system programming functions that define and inquire about *FEPI resources* and perform control functions.

Defining and configuring FEPI resources is called *setup program*. The **EXEC CICS FEPI** commands that provide these functions are:

INSTALL, ADD
Sets up communication resources

DISCARD, DELETE
Discards communication resources

INQUIRE
Queries FEPI resource status

SET Controls FEPI resources.

The setup functions are usually performed by a customer-written transaction that is started from a second-phase program list table post initialization (PLTPI) program. See “The configuration programs you should write” on page 38.

FEPI resources can be controlled, like other CICS resources, using the **CEMT SET** and **CEMT INQUIRE** functions. CECI can also be used. See Chapter 9, “Operator control of FEPI,” on page 65.

FEPI resources

There are four types of FEPI resource: *pool*, *property set*, *target*, and *node*. The relationships between them are listed below.

Pool A collection of nodes and targets

Property set
Defines the characteristics of a pool

CICS1, CICS2, IMS1, IMS 2...
Back-end systems

Node 1, Node 2, Node 3...
Simulated terminals

A FEPI pool can have one or more nodes and one or more targets. The same nodes and targets can be in any number of pools, except that the same node-target pair (a *connection*) cannot occur in more than one pool.

A CICS FEPI application can reach a target only by specifying a pool, which defines the set of nodes that can be used to make the connection, and the characteristics of the communication.

A target and an open node in the same pool are 'connected'; when bound, they are 'in session'. To *bind* means to establish a *session* on a *connection*, to make it ready to allow communication.

The process of communicating with a back-end system is called a *conversation*; it is the fundamental entity that a FEPI application deals with. Only one conversation can use a connection at one time, although any number can do so consecutively. For efficiency, the session on the connection is kept bound between conversations, unless you choose otherwise. Furthermore, a conversation is owned by the task that establishes it; no other task can use it.

Note: The use of the term conversation does not mean that the back-end or front-end application has to be conversational, in the CICS meaning of the term.

The resources are further explained in Chapter 7, "Configuring FEPI," on page 33 and the more complex relationships possible between them are illustrated in "An example FEPI configuration" on page 42.

CICS FEPI application programs

A CICS FEPI application consists of several distinct logical functions.

These logical functions are as follows:

Access programs:

Communicate with the back-end applications

Begin-session handler:

Handles begin-session processing

End-session handler:

Handles end-session processing

STSN handler:

Assists message synchronization

Unsolicited-data handler:

Handles unsolicited inbound data

Monitor:

Handles unexpected events such as the loss of a session or errors in setup.

These functions can be in separate programs, or contained in one program. The need for each function depends on the requirements of the application; in many cases default processing is all that you need. You might need several styles of each function, again depending on the requirements of your application.

The application programmer always writes the access programs. The system programmer usually writes the monitors to handle the unexpected events that FEPI reports to transient data queues such as CSZX. As for the other functions, sometimes the system programmer writes them providing, perhaps, just one

instance of each, so that they are common to everyone. (This approach has the advantage that adherence to standard procedures—for such things as signon and signoff—is enforced.) In other installations, the application programmers provide them.

In many cases, writing a CICS FEPI application is straightforward. However, some applications need more sophisticated programming. The programmer not only has to understand all the displays and protocols of the back-end application and system (CICS or IMS), but must also understand the detailed data-stream protocols. For further information, see Chapter 16, “FEPI application design,” on page 157.

Terminals supported

To access back-end applications, FEPI has VTAM secondary logical unit (SLU) support, so that CICS FEPI applications can simulate certain logical unit (LU) types. FEPI uses VTAM program-to-program support to provide this function, and to communicate between front-end and back-end applications.

Note: FEPI cannot send VTAM logon data.

FEPI provides simulation support for two families of terminals. The names **SLU2** and **SLU P** are used to identify the two types of support:

SLU2 for the 3270 family of terminals, used in many CICS and IMS applications. See the *3270 Data Stream Programmer's Reference*.

SLU P for a family of programmable terminals, including the 4700, accessed through an LU 0 protocol, for IMS applications. This protocol is defined in *IMS/VS Programming Guide for Remote SNA Systems* (for IMS/VS Version 2) or *IMS/ESA Customization Guide* (for IMS/ESA® Version 3 and later).

Data-stream-level and specialized-level commands can be used with both families of terminals, but the high-level commands, which use formatted data, are only for SLU2.

The mode of a conversation must be either SLU2 or SLU P; it cannot be mixed. For SLU2 conversations, formatted data or data stream data can be used, but cannot be mixed in the same conversation. The mode and data type are controlled by the pool used, which is set up by the system programmer.

These terminals are supported only when they are used to communicate with CICS or IMS systems.

FEPI Security

This section introduces FEPI security.

Signon security

Because FEPI is a terminal emulator, the back-end system “sees” the front-end as a terminal rather than a system; it cannot differentiate between FEPI emulation and a real device.

Thus, CICS bind, link, and attach-time security are not applicable to FEPI connections. If security is enabled in the back-end system, in order for your FEPI

application to access protected resources the emulated terminal must be signed on to the back-end. The alternative is that you do not use CICS security with FEPI—that is, you make all the back-end transactions accessed by FEPI available to the CICS default user. This option is clearly unacceptable; it means that you must either run a security risk or deprive your FEPI applications of access to sensitive data.

When signing on to a back-end system, FEPI applications can ask the external security manager (ESM) to supply a password substitute, or *PassTicket*. Using PassTickets to sign on means that FEPI applications do not need to store user passwords (which is risky), or ask users to reenter them (which is irritating). For information about implementing signon security, see “Signon security” on page 164.

Command security

You can restrict access to the FEPI system programming commands by defining operator profiles to your ESM.

For details of how to do this, see “Command-level security” on page 30. All application programming commands are generally available.

Problem determination, customization, and performance

The following functions are provided by FEPI.

- Debugging tools, trace, dump routines, and messages are available to help you determine the source of an error. These areas are described in Chapter 12, “FEPI problem determination,” on page 127.
- Two CICS global user exits are available for you to use with FEPI. They are described in Front End Programming Interface exits.
- Data that flows to and from CICS FEPI applications can be journaled for audit trails. For details, see “FEPI journaling” on page 85.
- You can use CICS monitoring and statistics data to help you tune FEPI applications, and to control the resources that they use. For details, see “Using CICS monitoring” on page 56.

Part 2. FEPI installation and administration

This section is intended for system programmers and administrators responsible for installing and configuring FEPI.

It contains the following topics:

- Chapter 5, "Planning for FEPI," on page 21 lists the things you need to consider when organizing your FEPI nodes, pools, and property sets.
- Chapter 6, "Getting started with FEPI," on page 29 describes how to install FEPI.
- Chapter 7, "Configuring FEPI," on page 33 describes the tasks required to implement your planned FEPI system.
- Chapter 8, "FEPI operation," on page 53 describes how FEPI operates.
- Chapter 9, "Operator control of FEPI," on page 65 describes the CICS-supplied transactions and VTAM commands that operators can use to control FEPI resources.
- Chapter 10, "Customizing FEPI," on page 81 describes how to use the FEPI global user exits and journaling function.
- Chapter 11, "FEPI system programming reference," on page 89 describes the FEPI system programming commands that are used to control FEPI resources.
- Chapter 12, "FEPI problem determination," on page 127 describes how to identify the source of errors that affect your FEPI applications.

Chapter 5. Planning for FEPI

This section is about planning your system and FEPI configuration.

To understand it, you need to be familiar with the basic FEPI concepts and terminology described in Part 1, “FEPI concepts and facilities,” on page 1. You must also be familiar with all aspects of CICS administration and operations; if you plan to use IMS, you also need to be familiar with IMS administration and operations.

The section contains the following topics:

- “Analysis and planning”
- “Organizing your pools and property sets” on page 24
- “Workload balancing in a sysplex” on page 26
- “Planning FEPI storage” on page 26.

Analysis and planning

When you are planning to implement FEPI, you must consider the configuration requirements, how to organize your pools, their properties, and the connections.

1. You must consider the following:
 - Details of the back-end applications and systems
 - Names of nodes and targets
 - Operator control requirements
 - Journaling requirements
 - Signon and signoff procedures
 - Special event handling
 - Pools required for control reasons
 - Pools required for functional reasons
 - Number of nodes
 - Setup program organization.
2. Decide how to organize your pools, their properties, and the connections.

Back-end applications and systems

You need to know whether the back-end systems are CICS or IMS, the terminal types they use, and the timing and volume of transactions expected.

Also, are there any restrictions on the use of the terminals? For example:

- Is a specific terminal required, or can any terminal be used?
- Is a specific LU or terminal type defined in the target application; for example, a 3278 model 3?

Names of nodes and targets

Decide on a naming convention for your nodes and targets.

Decide which VTAM node names are available for use by FEPI as simulated terminals. Remember that FEPI nodes are VTAM APPL definitions, not logical units (LUs). Do not use names starting with "DFH".

The back-end system already has defined VTAM primary PLU names (applids) which you must use. However, you can define your own local target names to associate with these applids. This means that FEPI applications are not affected if an applid is changed; you associate the local name with the new back-end target name. Do not use names starting with "DFH".

Operator control requirements

The CEMT INQUIRE and SET master terminal transactions can be used to view and amend the state of FEPI resources.

CEMT DISCARD can be used to remove resources from FEPI. This is described in Chapter 9, "Operator control of FEPI," on page 65. If you decide you need extra functions for operators, you will need to write appropriate programs.

Journaling requirements

Journaling is available if you need it.

Among the reasons for using FEPI journaling are:

- To create audit trails
- To monitor performance
- To control message security.

For further information, see "FEPI journaling" on page 85.

Signon and signoff procedures

You need to know if there are any specific requirements for signon and signoff to back-end systems. Central control might be required, or applications could perform signon and signoff individually.

Special event handling

In addition to signon and signoff, you must consider what special event handling is required and whether it should be handled by central functions or by applications individually.

The special event handling you must consider are as follows:

- The receipt of unsolicited data
- Unexpected events
- Beginning a session
- Ending a conversation or session
- Shutdown of the front-end CICS system.

If some sort of enforcement is required, or you want central provision for convenience, commonality, or the upholding of conventions and standards, you must supply a set of standard handlers. Otherwise, the application programs must handle each event. If you need special back-end processing when CICS shuts down, you need an end-session handler.

Unexpected events (including errors in setup) are reported to a transient data (TD) queue, so that a monitoring transaction can be triggered to handle them; they also send a message to the FEPI message log CSZL. You must decide how to handle these events, and which queues to use.

For more detailed information about the design and structure of applications, including information about using the various event handlers, see Chapter 16, “FEPI application design,” on page 157.

If you want central control over the range of FEPI commands that applications are permitted to issue, you can use the XSZBRQ global user exit, which is described in Front End Programming Interface exits.

Using pools for control reasons

You can use pools for a number of control purposes.

For example, you could define them so as to:

- Restrict users and applications to particular targets or nodes, or restrict access to some targets to particular times of day.
- Force specific begin-session and end-session effects.
- Split resources among different types of back-end requests, according to (for example) priority, or to the department issuing the request. By doing this, you can ensure that there is always a set of connections to a target for time-sensitive requests, while other connections handle long-running requests that are not time-sensitive.
- Ration the use of connections, especially for long-running requests, so that each set of users has access to only a limited number of connections.
- Ease signon considerations.

Using pools for functional reasons

Pools determine the data format and special event handlers used by your FEPI applications. These attributes may be specified by the application programmer, or they may be imposed by the system programmer for central control, especially of signon and signoff.

If you need several types of special event handling, you might need to define your own pool-specific transient data queues, as well as the default queues.

Number of nodes

The number of nodes required depends on:

- How the pools are structured
- How much storage is available
- How many concurrent sessions are required to a particular target.

The number of concurrent sessions to a particular target may depend on the volumes of data to be transmitted and the speed of the network.

Although a node can have only one session with a particular target at a time, it can communicate with several different targets concurrently, and several nodes can communicate with the same target concurrently.

Setup program organization

You must decide how to organize your setup program; for example, your setup program could consist of a single module, or a set of related modules.

You must also make the following decisions:

- Whether your programs should take replaceable parameters or fixed values. You might use mainly fixed programs, with a flexible program for one-off changes.
- When programs are to be run - started from a second-phase PLTPI program, under operator control, or at set times of the day.
- Where the definitions required by the setup program are to be obtained - from panel entry, from a file, or by other means.

Organizing your pools and property sets

When you have done the analysis work described in the previous section, you can decide how to organize your pools, their properties, and the connections between nodes and targets.

Organizing pools

There are several ways of organizing your pools.

About this task

- If possible, restrict each pool to a single target, but specify as many nodes as you believe you need to satisfy concurrent access to the target. The reasons for taking this approach are:
 - It avoids the need for the front-end application to specify a target.
 - It makes it easier to avoid duplicate connection definitions.
 - Because a connection is created for every node-target combination within a pool, having large numbers of both nodes and targets within the same pool may generate more resources than are required.
 - The overhead associated with a pool is very small. Therefore there is no reason not to define many pools.
 - The expected concurrent usage of each target may be different. If you have more than one target in the pool, it becomes difficult to estimate the number of nodes required.
- You can define a pool containing only one node and one target. This lets a FEPI application allocate a specific session, which is necessary if the target system associates any special qualities with a particular terminal ID. You can use the XSZBRQ global user exit to control access to the pool.
- You can define pools that use different nodes to reference the same target. By making each pool available to a different group of users, you can eliminate competition for resources. Alternatively, you could use each pool to support a different set of properties, according to application requirements.
- If you plan to use the **VTAM CLSDST(PASS)** command, other considerations might apply. See "Handling CLSDST(PASS)" on page 48.

Do not use names starting with "DFH" for pools.

Organizing property sets

Property sets allow you to define the properties of pools (such as the data format and special functions they use) separately from the definition of the pool itself.

You can use a single property set to define any number of pools. You must define as many property sets as you need to satisfy every unique pool requirement. Because the overhead associated with a property set is very small, there is no reason why you should not define a large number of them.

The properties are:

Device attributes

This specifies which family the simulated terminal belongs to, SLU2 or SLU P. For SLU2, it also determines the presentation size of the display (24 x 80, 32 x 80, and so on), and whether it supports extended attributes such as color.

Many back-end applications can be run with any terminal type, so you can use the default device type (SLU2, 3278 model 2). But if you have applications that demand particular terminal types, you need to define pools with the appropriate device types.

Data handling

This specifies which command level to use (high-level with formatted data, or data stream), how much data can be handled, and how contention is to be handled.

High-level is simpler to use and suits many front-end applications; applications that require sophisticated functions or use SLU P, and those performing a simple pass-through, need the more complex data-stream-level. In most cases the default data size of 4096 is adequate; increase it only if you know there are large amounts of data to send and receive in a single command. Set contention handling so that the front end wins—as for a real terminal—unless you have some particular reason for not doing so.

Session management

This specifies whether begin-session and end-session are to be handled by special transactions, and whether initial inbound data is expected. For SLU P, it also includes whether message resynchronization (“set and test sequence number” (STSN)) is to be handled.

The use of event handlers was introduced on page “Signon and signoff procedures” on page 22; it is generally preferable to use specially written transactions for session management, rather than to leave it to be handled individually by applications.

If a back-end system sends initial data (a “good morning” message) you must specify this as a property of the pool, so that FEPI waits for the data to arrive and ensures that the front-end application receives it; otherwise the results will be unpredictable. For SLU2, IMS always sends initial data; CICS might or might not do so, depending on your system definition.

FEPI does all the necessary STSN handling automatically, but you can specify a transaction to handle it yourself.

Unexpected events

This specifies how unsolicited data and other unexpected events (including setup errors) are to be handled.

General considerations of the need for transactions and queues have been discussed earlier in this section. If you choose not to handle unsolicited data in your own transaction, you can tell FEPI how to handle it for you—positively or negatively; if the back-end system is IMS, you must specify that FEPI should respond positively. All unexpected events are logged in the FEPI message log (CSZL), even if you specify no unexpected event queue.

Journaling

This specifies what sort of data journaling is required, and which journal to use.

Do not use names starting with "DFH" for property sets.

Workload balancing in a sysplex

In an MVS sysplex, you can create a CICSplex consisting of sets of functionally-equivalent CICS terminal-owning regions (TORs) and application-owning regions (AORs). If the FEPI back-end system is a TOR in such a CICSplex, you can use the VTAM generic resource function to perform workload balancing across the available TORs.

A VTAM application program such as CICS can be known to VTAM by a generic resource name, as well as by the specific network name defined on its VTAM APPL definition statement. A number of CICS regions can use the same generic resource name.

A FEPI application, wishing to start a session with a CICSplex that has several terminal-owning regions, names a target that you have defined as the generic resource name of the TORs. Using the generic resource name, VTAM is able to select one of the CICS TORs to be the target for that session. For this mechanism to operate, the TORs must all register to VTAM under the same generic resource name. VTAM is able to perform dynamic workload balancing of the terminal sessions across the available terminal-owning regions.

For information about defining FEPI targets as VTAM generic resource names, see the APPLLIST option of the **FEPI INSTALL TARGETLIST** system programming command. For further information about VTAM generic resources, see the *CICS Intercommunication Guide*.

Planning FEPI storage

FEPI does not require any additional MVS storage beyond that recommended for basic CICS.

As for dynamic storage, the storage used by FEPI is allocated exclusively from CDSA and ECDSA; CDSA usage is only that required to support VTAM processing. The following information allows you to estimate the storage requirements of a particular FEPI configuration.

Table 1. Dynamic storage requirements (in bytes)

Item	ECDSA	CDSA
Basic	80K	
For each node	288	180
For each node that is currently available for communication	192	
For each target	236	
For each pool	272 + 64 x (number of nodes in pool) + 64 x (number of targets in pool)	
For each property set	176	

Table 1. Dynamic storage requirements (in bytes) (continued)

Item	ECDSA	CDSA
For each connection (note 1)	432 if using data stream data 688 if using formatted data	
For each connection that is currently available for communication	384 + additional value from Table 2 if using formatted data	
For each current conversation	128	
For each command in progress	2.5K + size of user data (Note 2)	
Note:		
1. The number of connections is (number of nodes in pool) x (number of targets in pool) for each pool.		
2. This is the data that is to be sent and received, or used for defining resources. If global user exits are used, twice the data size is needed; similarly if journaling is used.		

For each connection that is currently available for communication and that uses formatted data, additional ECDSA storage is required; the amount depends on the device type and capabilities defined, as shown in Table 2.

Table 2. Connection storage requirements (in bytes) by device type and function

Device type	Basic	Additional for color support	Additional for extended data stream support	Maximum
327x model 2	3840	1920	5760	11520
327x model 3	5120	2560	7680	15360
327x model 4	6880	3440	10320	20640
327x model 5	7128	3564	10692	21384

You should add some contingency (say 10%) to your final estimate.

Chapter 6. Getting started with FEPI

FEPI is installed automatically when you install CICS. However, to make it operative you need to install some additional resources.

The installation process

The FEPI installation process has a number of steps.

These steps are as follows:

1. Updating CICS resource definitions
2. Installing FEPI resource definitions
3. Starting CICS.

A note about loading FEPI modules into the LPA

Any of the FEPI modules can be loaded in the MVS Link Pack Area (LPA).

However, as with CICS modules in general, it is not recommended that you do so. For information about installing modules in the LPA, see the *CICS Transaction Server for z/OS Installation Guide*.

Updating CICS definitions

The RDO group DFHFEPI, which is on the product tape, contains resource definitions for the FEPI programs and the FEPI transaction CSZI. FEPI programs are prefixed with DFHSZ.

DFHFEPI is included in the default startup group list, DFHLIST.

Use the CEDA transaction:

- To define your FEPI application programs
- If you have installed FEPI modules in the LPA, modify the definitions of the modules in the CICS system definition file (the CSD), so that they specify USELPACOPY(YES).

Transient data queues

Sample definitions for the transient data (TD) queues required by FEPI are supplied in group DFHDCTG. You can use the sample definitions, or create your own, together with any extra queues that you need.

The required queues are:

CSZL The FEPI message log. You can define CSZL as an intrapartition, extrapartition, or indirect queue. Note that CSZL must be defined as non-recoverable.

It is recommended that you define CSZL as an indirect queue, pointing to CSSL.

CSZX The queue for information about unexpected events (including setup errors) that do not relate to specific pools. You can define CSZX as an intrapartition, extrapartition, or indirect queue. Note, however, that it must be defined as non-recoverable.

It is recommended that you define CSZX as an intrapartition queue, with a trigger level of 1, so that each event is processed immediately it is reported. (You must also, of course, write and install the event-handling transaction that is to be triggered.)

Any pool-specific TD queues that you require

Such queues receive information about events that affect specific pools. They can be defined as intrapartition, extrapartition, or indirect queues. Note, however, that they must be defined as non-recoverable.

It is recommended that you define pool-specific queues as intrapartition queues with trigger levels of 1, so that each event is processed immediately it is reported.

For information about defining transient data queues, see the *CICS Resource Definition Guide*.

System initialization parameter, FEPI=YES/NO

Code **FEPI=YES**, to specify that FEPI is available.

(The default is **FEPI=NO**.) For information about setting system initialization parameters, see *../com.ibm.cics.ts.doc/dfha2/parameters/dfha2_fepi.dita*, in the *CICS System Definition Guide*.

Command-level security

If your installation uses CICS command-level security, you can restrict access to the EXEC CICS FEPI system programming commands (and to the equivalent commands that you can issue with the CEMT master terminal transaction) by defining access authorizations to your external security manager (ESM).

The commands you can protect in this way are those listed in Chapter 11, “FEPI system programming reference,” on page 89 and in the CEMT section of Chapter 9, “Operator control of FEPI,” on page 65. You cannot restrict access to the FEPI application programming commands (as listed in Part 4, “FEPI application programming reference,” on page 175).

To protect the FEPI system programming commands, use the resource identifier ‘FEPIRESOURCE’ when defining resource profiles to the ESM. Note that, if you use command security, you must ensure that authorized users of CEMT are also authorized to use the FEPI commands.

For RACF[®] users, details of how to define resource profiles to the ESM are in the *z/OS Security Server RACF Security Administrator’s Guide*. For information about using RACF with CICS, see *../com.ibm.cics.ts.doc/dfht5/topics/dfht5_overview.dita*, in the *CICS RACF Security Guide*. Users of other security managers must refer to the documentation for their own product.

Installing FEPI resource definitions

Ensure that the RDO group DFHFPEPI is in your startup group list. (DFHFPEPI is in the DFHLIST startup group list, so this should have been done automatically when you installed CICS.)

About this task

Starting CICS

Start your CICS region.

This is described in `../com.ibm.cics.ts.doc/dfha2/topics/dfha2_cics_startup.dita#dfha24z`, in the *CICS System Definition Guide*.

Chapter 7. Configuring FEPI

When you have completed the planning work to understand how to set up FEPI in your system, you can carry out the configuration tasks.

Before you begin

To configure FEPI, you need to be familiar with all aspects of CICS administration (such as system definition, customization, resource definition, and operations) and the programming interface to CICS. You should also be familiar with VTAM and, if you are accessing IMS back-end systems, with IMS administration.

About this task

The stage needs to be set just so.

Procedure

1. Define your FEPI applications to CICS
2. Define nodes to VTAM
3. Define simulated terminals to the back-end systems
4. Write the following for FEPI itself:
 - a. A setup program, to install your FEPI resources.
 - b. A monitoring program, to handle unexpected events.
 - c. Optional: Global user exit programs, common functions and transactions for operator control and administration.

What to do next

This section contains the following topics:

- “Configuring CICS”
- “Configuring VTAM” on page 34
- “Configuring the back-end systems” on page 37
- “The configuration programs you should write” on page 38.

Related information

Introduction to system programming commands

CICS API commands

Configuring CICS

Before you begin

Chapter 6, “Getting started with FEPI,” on page 29 covers everything that FEPI itself requires: the RDO group DFHFEPI in the startup group list; definitions of the transient data queues CSZL and CSZX; and any required security access controls.

About this task

Now you have to define your FEPI applications to CICS in the usual way. This includes the setup programs, any common functions, and any additional transient

data queues that you need for handling pool-specific events. Note that, in an intercommunication environment, FEPI itself must be run in the application-owning region (AOR) and all transactions that FEPI may start must run locally. This is because FEPI commands cannot be function shipped.

Procedure

1. Define transactions that are to be started by FEPI (the event handlers and pseudoconversational access programs) as CICS started tasks, with SPURGE=NO and TPURGE=NO to prevent them from being accidentally canceled by CICS
2. Define any additional transient data queues. See “Transient data queues” on page 29 for details of the queues.
3. Before starting CICS, you should ensure that your system has enough storage available to support your FEPI configuration: for details see “Planning FEPI storage” on page 26.
4. If you are using a setup transaction to install your FEPI nodes, targets, and pools that is started by a program list table (PLT) program, you need to include your PLT program in the second part of the program list table post initialization (PLTPI) list. This process is described in “Running setup programs” on page 41. If you use this method, you need to include your PLT program in the second part of the program list table post initialization (PLTPI) list.

What to do next

For information about coding entries in the PLTPI list, see the *CICS Resource Definition Guide*.

Related concepts

PLT — program list table

Configuring VTAM

For FEPI to communicate with the network, some information must be defined to VTAM.

Procedure

1. Each FEPI node (simulated secondary LU terminal) must have a VTAM application minor node definition. The name of this minor node must be the same as the node name specified on the **FEPI INSTALL NODELIST** command. For example, the FEPI node called 'FEPI0001' would require the following application minor node definition in VTAM:
DG4FEPI1 APPL ACBNAME=FEPI0001
2. If your network uses a naming convention to manage network resources, you can allow a network-independent name to be used by specifying it on the ACBNAME keyword of the VTAM APPL statement. If this is not the case, you can simplify the definition of the VTAM application minor node by omitting the ACBNAME keyword (which means that the margin-name, DG4FEPI1 in the example, must be the same as the FEPI node name).

FEPI does not impose any additional restrictions on the naming of nodes, other than that the names should not begin with “DFH”; apart from this, any values acceptable to VTAM are acceptable to FEPI.

3. If you require password protection of the minor nodes, you can use the PRTCT keyword of the VTAM APPL statement to specify a password of 1–8 characters. The password must then be specified on the corresponding **FEPI INSTALL NODELIST** command.
4. If you are defining multiple FEPI nodes, you can place them all in a single member (also known as a VTAM application major node) or in several members. You can also add them to an existing VTAM application major node. How you choose to organize the VTAM definitions can depend on how your installation manages its network resources, or how you plan to manage the FEPI configuration. VTAM application minor node definition statements are stored collectively as one or more members of an MVS partitioned data set (usually SYS1.VTAMLST), accessed by VTAM via the VTAMLST data-definition statement in the VTAM startup JCL.

What to do next

For general information about configuring VTAM, see the *VTAM Network Implementation Guide* and the *VTAM Resource Definition Reference*.

Availability of network resources

For FEPI to communicate with the network using a node, both the application minor node and the defining major node must be active, and the minor node must be in a connectable condition.

If FEPI is initialized before VTAM, and is instructed to acquire this node, it retries the VTAM OPEN request several times. Similarly, if a target application is unavailable, FEPI makes another attempt at session initiation. After this, the operator will need to intervene to establish connectivity.

Selection of FEPI session parameters

When FEPI establishes a session with a back-end system, it searches the VTAM LOGON mode (logmode) table for an entry that corresponds to the simulated device type specified on the FEPI INSTALL PROPERTYSET command used to define the pool to which the node-target connection belongs.

If it finds such an entry, it uses it to set the parameters for the session. Suitable mode table entries for FEPI are in the LOGON mode table ISTINCLM. Table 3 shows how entries in ISTINCLM correspond to FEPI device types.

Table 3. Relation of FEPI device-types to ISTINCLM mode table entries

DEVICE CVDA on FEPI INSTALL PROPERTYSET	Mode table entry in ISTINCLM	Session parameters
T3278M2	D4A32782	LU2 3278 model 2
T3278M3	D4A32783	LU2 3278 model 3
T3278M4	D4A32784	LU2 3278 model 4
T3278M5	D4A32785	LU2 3278 model 5
T3279M2	SNX32702	LU2 3279 model 2
T3279M3	SNX32703	LU2 3279 model 3
T3279M4	SNX32704	LU2 3279 model 4
T3279M5	SNX32705	LU2 3279 model 5
TPS55M2	SNX32702	LU2 PS/55, 24 lines

Table 3. Relation of FEPI device-types to ISTINCLM mode table entries (continued)

DEVICE CVDA on FEPI INSTALL PROPERTYSET	Mode table entry in ISTINCLM	Session parameters
TPS55M3	SNX32703	LU2 PS/55, 32 lines
TPS55M4	SNX32703	LU2 PS/55, 43 lines
LUP	IBM3600	Secondary LU P (IMS protocol LU 0)
Note: The mode entries are fixed by FEPI; you cannot use any other entries.		

If *ISTINCLM* is defined as your default LOGON mode table, no additional definitions are required, and FEPI sessions use the characteristics that these entries specify. If you have defined a different default table, which does not contain the supplied entries, or if you want to associate a different set of characteristics with the names listed above (for example, class-of-service or pacing specifications), then you must provide the required entries in a customized mode table. This must be associated with the node via the *MODETAB* keyword of the VTAM APPL statement used to define the node to VTAM. For example:

```
DG4FEPI1 APPL ACBNAME=FEPI0001,MODETAB=mode-table-name
```

Note:

1. If you choose to define your own mode table, it needs to contain only those entries that differ from the set supplied in the default mode table (for example, *ISTINCLM*). If VTAM cannot find a given entry in the node-specific mode table, it automatically searches the system default table for an entry of the same name.
2. FEPI establishes the presentation space size of a terminal, based on the session parameters received in response to the session request, *not* on any fixed dimension implied by the device type specified for the pool (although the device type does establish a default value when a default BIND is received).
3. An externally initiated session (one started by the primary LU or by the operator through the VARY LOGON command) can specify any entry name in the mode table. If you expect to make use of external session initiation, it is advisable to specify the *DLOGMOD* keyword on the APPL statement used to define the node in question. This keyword identifies the mode table entry to be used in those cases where the session initiation request did not specify session parameters. It can be specified regardless of whether the *MODETAB* keyword is used. For example:

```
DG4FEPI1 APPL ACBNAME=FEPI0001,  
MODETAB=mode-table-name,DLOGMOD=mode-table-entry-name
```
4. If you define your own mode entries, ensure that all the parameters in an entry are appropriate. These logmode entries should be explicitly named in the APPL statements as described in note 3.

Pacing of FEPI sessions

The pacing values used for FEPI sessions should be consistent with whatever installation standards are in effect for other LU2 and SLU P sessions in the network.

Configuring the back-end systems

The only configuration that you must perform for the back-end systems is to provide and manage simulated terminals (LUs) for FEPI to use. These terminals are defined to the back-end CICS or IMS system just like real terminals. They can be explicitly defined or autoinstalled as required.

About this task

The simulated terminals do not have to be defined to VTAM in the back-end system, where they appear as real terminals on that system. VTAM uses the various network definitions to determine how and where to route data; it can be routed locally, cross-domain, or cross-network. The LU name corresponds to the front-end node name. Similarly, the VTAM applid of the back-end system corresponds to the applid in the FEPI target definition. The diagram of the sample configuration in Figure 2 on page 43 illustrates these relationships.

If your back-end systems use the extended recovery facility (XRF), you must use their generic APPLIDs, rather than specific ones, in your FEPI target definitions. See “Using FEPI with XRF” on page 58.

Procedure

- If you are configuring CICS as the back-end system, the following terminal definitions (TYPETERMs) are acceptable:

- DFHLU2E2
- DFHLU2E3
- DFHLU2E4
- DFHLU2E5
- DFHLU2M2
- DFHLU2M3
- DFHLU2M4
- DFHLU2M5

These definitions match the VTAM mode table entries shown in Table 3 on page 35. You must create your own TYPETERMs for 3279 model 5 and PS/55 devices, if required, because no such definitions are supplied by CICS. For information about defining terminals to CICS, see the *CICS Resource Definition Guide*.

- If you are configuring IMS as the back-end system, use the following settings on the TYPE or TERMINAL system definition macros:
 1. Required: NAME must match the NODE name specified to and used by FEPI
 2. Required: MODETBL must specify the correct LOGMODE.

The following nondefault settings are recommended. (FEPI will support the default settings as well.)

1. Specify OPTIONS=OPTACK for more efficient communication.
2. Specify OPTIONS=FORCRESP so transactions are run in response mode. If you let this default, you might get nonresponse mode regardless of how the transactions are defined.
3. Specify OPTIONS=NORELRQ to make IMS ignore external requests for the node.
4. Specify OPTIONS=BID to indicate that the VTAM BID command should always precede output messages that occur while between brackets.

5. Specify `OUTBUF=nnn` to set a bigger output buffer than the default of 256 bytes.

The following example defines some IMS terminals for use by FEPI. You might need to customize it for use in your own IMS environment.

```
TYPE UNITYPE=SLUTYPEP,MODETBL=IBM3600,          x
OPTIONS=(OPTACK,FORCRESP,NORELRQ,BID),OUTBUF=512
TERMINAL NAME=IMSLUP01
NAME IMSLUP01
TERMINAL NAME=IMSLUP02
NAME IMSLUP02
TERMINAL NAME=IMSLUP03
NAME IMSLUP03
TERMINAL NAME=IMSLUP04
NAME IMSLUP04
```

The configuration programs you should write

You must write a setup program to define your FEPI nodes, targets, property sets, and pools.

You can also choose to write:

- A monitoring program to handle unexpected events (including setup errors)
- Any common functions not provided by individual FEPI applications
- One or more global user exit programs
- Some specialized operator transactions, to simplify the control of FEPI resources.

A number of samples have been provided to give you an example of the types of programs that you can write. See Appendix A, “FEPI sample programs,” on page 237 for details.

Writing configuration programs

FEPI programs are CICS applications, and so all aspects of CICS programming apply.

Before you begin

You should familiarize yourself with the guidance about writing CICS application programs. See the *CICS Transaction Server for z/OS Application Programming Guide*. Particularly relevant are the sections about designing efficient applications and dealing with exception conditions in the *CICS Application Programming Guide*.

About this task

To write a configuration program, use the FEPI system programming commands. They are an extension of the **EXEC CICS** commands and have similar names and similar functions. The FEPI commands also have similar keywords, but they are distinguished by having “FEPI” as a prefix.

Procedure

1. You can use the following system programming commands in your configuration program:

Definition:

```
EXEC CICS FEPI INSTALL
      Define communication resources
```

EXEC CICS FEPI ADD

Add resources to a pool

EXEC CICS FEPI DELETE

Remove targets or nodes from a pool

EXEC CICS FEPI DISCARD

Remove communication resources completely from FEPI.

Operations:

EXEC CICS FEPI INQUIRE

Query FEPI status and resources

EXEC CICS FEPI SET

Control FEPI resources.

2. When translating your programs, you must specify the FEPI option, which instructs the translator to process FEPI commands. You do not need the SP option.
3. Select whether your FEPI configuration programs are AMODE(24) or AMODE(31). The configuration programs can issue FEPI commands in either 24- or 31-bit addressing mode, and reside above or below the 16MB line.
4. Consider how your configuration program should handle exception conditions.

Related information

CICS API commands

Exception conditions

As with all CICS commands, FEPI commands might produce exception conditions that you can check using the RESP option, or capture using HANDLE CONDITION. Most FEPI command errors return INVREQ. The particular error in each case is uniquely identified by the RESP2 value.

All the FEPI exception conditions and RESP2 values are listed in Chapter 11, "FEPI system programming reference," on page 89. There are copy books that contain declarations for the RESP2 values:

- DFHSZAPA for Assembler language
- DFHSZAPO for COBOL
- DFHSZAPP for PL/I
- DFHSZAPC for C.

For the system programming commands, errors are reported as unexpected events to the CSZX or other transient data queue, and to the FEPI message log CSZL, as well as by exception conditions on the command.

If there is an error, the command does nothing, and output values are not changed. Some commands operate on a list of resources; an error in one resource does not prevent the command from operating on the other resources in the list.

You can use EDF and CECI to debug FEPI programs. Because FEPI commands can be quite long, you will probably find the NAME field of CECI useful.

All resource names used by FEPI are a fixed length of 8 characters; they must be padded with blanks if necessary. For commands that use lists, make sure that the list field is a multiple of 8 characters long and that the number option is set correctly; neither the translator nor CECI checks these and unpredictable results could occur if they are wrong.

Writing setup programs

There are many considerations in designing setup programs, and so there is no single recommended way of writing them. However, there are certain functions that your setup program must perform.

About this task

On the distribution tape, the following sample setup programs are provided:

- An Assembler language sample setup program with filename DFH0AZXS
- A COBOL sample setup program with filename DFH0VZXS
- A C sample setup program with filename DFH0CZXS.

These programs install resources to make FEPI function with the other sample programs. They show you one way of writing setup programs.

Your setup programs must:

Procedure

1. Install all node names that are available for FEPI.
2. Install all targets that FEPI is permitted to access.
3. Install properties. See “Organizing property sets” on page 24 for guidance on what choices to make. In defining the properties of connections in pools, the following options must be set:

Device attributes

DEVICE

Data handling

FORMAT, MAXLENGTH, CONTENTION

Session management

BEGINSESSION, ENDESESSION, INITIALDATA, STSN

Unexpected events

EXCEPTIONQ, UNSOLDATA, UNSOLDATAACK

Journaling

MSGJRNL, FJOURNALNUM, FJOURNALNAME

4. Install pools.
5. Associate nodes and targets with the pools to define connections.

What to do next

In addition to a setup program, you may need a corresponding program to deal with deleting and discarding resources.

By default, FEPI resources are available for use as soon as they are installed or associated with a pool. For control, performance, or other reasons, you might want to override this; if so, you must provide a further program (or operations procedure) to bring the resources into service when you require them.

Many of the FEPI commands used by your setup program can use lists; using lists helps to improve performance. If some items in a list fail, errors (both programming errors and resource problems) are reported to your monitoring program, *not* to the setup program. If you want to track the errors in the setup program itself, without using the monitoring program, restrict your lists to a single

item. Errors are then reported on the command itself.

Running setup programs

The setup program is typically initiated by a program list table (PLT) program.

About this task

Using this method, the setup program is run automatically at every CICS startup, including an XRF takeover. Follow this procedure:

Procedure

1. Write your setup program.
2. Define it to CICS, using RDO, and associate it with a transaction. You can define your setup program statically, or allow it to be installed automatically (autoinstalled) when it is invoked. For details of the CICS autoinstall facility for programs, see the *CICS Resource Definition Guide*.
3. Write a PLT program containing the command:

```
EXEC CICS START TRANSID(tranid) INTERVAL(1)
```

where tranid is the ID of your setup transaction. (For programming information about writing PLT programs, see the *CICS Customization Guide*.)

4. Define your PLT program to CICS, and include it in the second part of the program list table post initialization (PLTPI) list. For information about coding entries in the PLTPI list, see the *CICS Resource Definition Guide*.

What to do next

There may be a good reason for you to decide not to use the PLT to start the setup transaction. For example, you may want to have several, time-sensitive, setup programs, each having a corresponding discard program. If you decide not to use the PLT, you must arrange to start the setup transactions manually.

You should restrict access to the setup programs, because they are of a sensitive nature.

Related concepts

PLT — program list table

Varying the resources installed by the setup program

Unless your setup program contains some conditional logic, you always get the same set of FEPI resources installed. This may be exactly what you require, but if not, here are a few techniques that might prove useful.

Checking startup type

Your setup program can determine how the CICS system started by issuing an **EXEC CICS INQUIRE SYSTEM STARTUP** command. It could use this to install different sets of FEPI resources for warm and cold starts.

Recording the status of resources

If you install all your FEPI resources at CICS startup, and then alter their accessibility, consider writing a non-terminal transaction that runs frequently and uses the **FEPI INQUIRE** commands to determine the status of each FEPI resource.

Write these to a *recoverable* temporary storage file. (You could, for example, use an XSZARQ global user exit program to log changes to FEPI resources.) At restart time, your setup program can read the file to determine the required access settings.

Using timed actions

You could take advantage of CICS automatic transaction initiation (ATI) at specified times to control FEPI resources. If you want to terminate FEPI access to another system at a specific time each day, schedule a transaction to run at the required time. When this transaction runs it can either make the required FEPI resources unavailable for access, or discard them. Because FEPI resources remain available for use by current tasks in this circumstance, this has no effect on existing FEPI users.

You could use timed initiation in a similar way to make FEPI resources available.

Using event handlers

Another way of controlling FEPI resources is to use the begin-session and end-session event handlers. (See “Other functions” on page 50.)

These handlers are invoked when a conversation starts and ends. Although they are primarily designed to handle signon and signoff to the back-end systems, you can take advantage of the fact that all FEPI functions are available to them. So you can use them to control access to back-end systems by either installing or discarding FEPI resources.

For example, suppose you want to ensure that no FEPI application is waiting for a connection to a back-end system. In the handlers, issue **FEPI INQUIRE POOL** commands, and look at the WAITCONVNUM option, which returns the number of FEPI applications waiting for a connection. If this option exceeds a certain trigger value, issue FEPI commands to increase the number of connections (that is, add nodes, define new pools, and so on).

This technique can be extended to provide tuning of FEPI access to back-end systems.

An example FEPI configuration

An example configuration is given in Table 4 on page 43. Next, the target lists and node lists used in the example are given. Then there are the definitions used to achieve this configuration. Figure 2 on page 43 is a diagrammatic representation of the configuration.

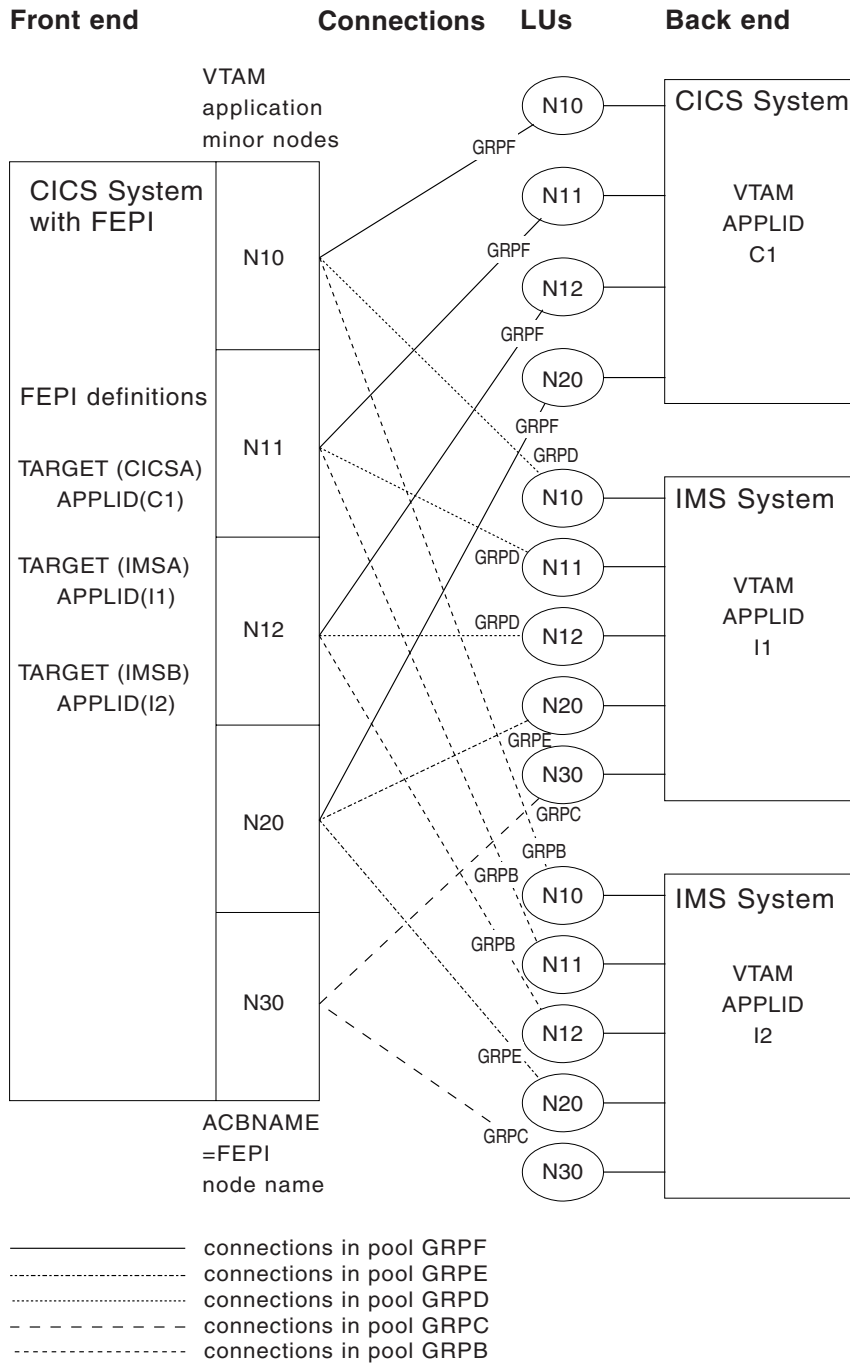


Figure 2. The example FEPI configuration - a diagrammatic representation

Note that this is not the configuration the sample programs use; it illustrates as many aspects of configuration as possible.

Table 4. Resources used in the example FEPI configuration

Pool name	GRPB	GRPC	GRPD	GRPE	GRPF
Property set	SLUP	SLU2M3I	SLU2M3I	SLU2M2I	SLU2M2C
Target names	IMSB	IMSA IMSB	IMSA	IMSA IMSB	CICSA

Table 4. Resources used in the example FEPI configuration (continued)

Node names	N10 N11 N12	N30	N10 N11 N12	N20	N10 N11 N12 N20
Device type	LUP	T3278M3	T3278M3	T3278M2	T3278M2
Logmode name	IBM3600	D4A32783	D4A32783	D4A32782	D4A32782
Exceptional events queue name	IEXEPTP	IEXEPT2	IEXEPT2	IEXEPT2	CEXEPT2
Unsolicited-data transaction name or response	IUP	IU2	IU2	IU2	Negative
Begin-session transaction name	ISIP	ISI2	ISI2	ISI2	CSI2
End-session transaction name	none	IXI2	IXI2	IXI2	CXI2
STSN transaction name	ISTP	n/a	n/a	n/a	n/a
Initial inbound data	No	Yes	Yes	Yes	Yes

Sample lists

Here are the target lists and node lists used in the sample configuration, padded to eight bytes per item.

```

TLIST 'CICSA  IMSA  IMSB  '
TLISTA
      'IMSA  '
TLISTB
      'CICSA  '
TLISTC
      'IMSA  IMSB  '
TLISTD
      'IMSB  '
NLIST
      'N10   N11   N12   N20   N30   '
NLISTA
      'N10   N11   N12   '
NLISTB
      'N20   '
NLISTC
      'N30   '
NLISTD
      'N10   N11   N12   N20   '

```

The following is the list of VTAM application names of the back-end CICS and IMS systems with which FEPI applications will communicate.

```

PLIST 'C1    I1    I2    '

```

Sample definitions

The following definitions illustrate the various possibilities when defining FEPI resources.

Define the back-end subsystems you want FEPI to access:

This defines the logical names (targets) that FEPI uses to refer to back-end systems (in this case CICS, IMSA, and IMSB as given in TLIST), and relates them to their VTAM names (C1, I1, and I2 as given in PLIST).

```
EXEC CICS FEPI INSTALL TARGETLIST(TLIST) TARGETNUM(3)
      APPLIST(PLIST)
```

Define the VTAM minor nodes available to FEPI:

The names are N10, N11, N12, N20, and N30, as given in NLIST.

```
EXEC CICS FEPI INSTALL NODELIST(NLIST) NODENUM(5)
```

Define properties:

The properties define the characteristics of the connections.

SLU P connections

```
EXEC CICS FEPI INSTALL PROPERTYSET(SLUP)
      LUP /* Device type (SLU P) */
      BEGINSSESSION(ISIP) /* Begin session handler */
      STSN(ISTP) /* STSN transaction */
      EXCEPTIONQ(IEXEPT) /* Exception report TD queue */
      UNSOLDATA(IUP) /* Unsolicited-data transaction */
      NOTINBOUND /* No "good morning" message */
```

SLU2 24 x 80 connections to IMS

```
EXEC CICS FEPI INSTALL PROPERTYSET(SLU2M2I)
      T3278M2 /* Device type (3278 model 2, 24 x 80) */
      BEGINSSESSION(ISI2) /* Begin session handler */
      EXCEPTIONQ(IEXEPT2) /* Exception report TD queue */
      UNSOLDATA(IU2) /* Unsolicited-data transaction */
      INBOUND /* Initial data */
      ENDSSESSION(IXI2) /* End session handler */
```

SLU2 32 x 80 connections to IMS

```
EXEC CICS FEPI INSTALL PROPERTYSET(SLU2M3I)
      T3278M3 /* Device type (3278 model 3, 32 x 80) */
      BEGINSSESSION(ISI2) /* Begin session handler */
      EXCEPTIONQ(IEXEPT2) /* Exception report TD queue */
      UNSOLDATA(IU2) /* Unsolicited-data transaction */
      INBOUND /* Initial data */
      ENDSSESSION(IXI2) /* End session handler */
```

SLU2 24 x 80 connections to CICS

```
EXEC CICS FEPI INSTALL PROPERTYSET(SLU2M2C)
      T3278M2 /* Device type (3278 model 2, 24 x 80) */
      BEGINSSESSION(CSI2) /* Begin session handler */
      EXCEPTIONQ(CEXEPT2) /* Exception report TD queue */
      NEGATIVE /* Response to unsolicited data */
      INBOUND /* "Good morning" message */
      ENDSSESSION(CXI2) /* End session handler */
```

Define the pools of connections:

The pools define connections between targets and nodes; they specify which nodes can be used to access which target, and what properties the connection has.

```
EXEC CICS FEPI INSTALL POOL(GRPB) PROPERTYSET(SLUP)
      TARGETLIST(TLISTD) TARGETNUM(1)
      NODELIST(NLISTA) NODENUM(3)
```

```

EXEC CICS FEPI INSTALL POOL(GRPC) PROPERTYSET(SLU2M3I)
      TARGETLIST(TLISTC) TARGETNUM(2)
      NODELIST(NLISTC) NODENUM(1)
EXEC CICS FEPI INSTALL POOL(GRPD) PROPERTYSET(SLU2M3I)
      TARGETLIST(TLISTA) TARGETNUM(1)
      NODELIST(NLISTA) NODENUM(3)
EXEC CICS FEPI INSTALL POOL(GRPE) PROPERTYSET(SLU2M2I)
      TARGETLIST(TLISTC) TARGETNUM(2)
      NODELIST(NLISTB) NODENUM(1)
EXEC CICS FEPI INSTALL POOL(GRPF) PROPERTYSET(SLU2M2C)
      TARGETLIST(TLISTB) TARGETNUM(1)
      NODELIST(NLISTD) NODENUM(4)

```

Writing monitoring programs

You must write a monitoring program to handle unexpected events that are reported by FEPI and errors returned by system programming commands.

FEPI reports these events by writing a record to a transient data (TD) queue. You can define pool-specific TD queues for FEPI, where information about events that relate to specific pools is reported. (There is also a common FEPI TD queue, CSZX, where events that do not relate to specific pools are reported.) Note that, if a pool-specific event occurs, and you have not defined a corresponding queue, information about the event is lost. Also, FEPI TD queues must be defined as NONRECOVERABLE; if a queue is 'recoverable', FEPI does not write to it, and discards any information about unexpected events.

Typically, you would arrange for the monitoring program to be triggered whenever an item is placed in a TD queue. (Define the queue with a trigger level of 1.) A single monitoring program can service several queues, by using EXEC CICS ASSIGN QNAME to check which queue triggered it. According to the nature of the event, the monitoring program might write a message, log the event, or embark on a full conversation.

For example, using this method, whenever a session is lost, the monitoring program is invoked. The TD queue data provides information about what happened. Your monitoring program can obtain this in the usual way with EXEC CICS READQ TD. The following copy books describe the structure of the data:

- DFHSZAPA for Assembler language
- DFHSZAPO for COBOL
- DFHSZAPP for PL/I
- DFHSZAPC for C.

Your program may then choose to reestablish the lost session, to reinitialize, and so on. It may also set indicators for the application programs if contact with a target has been lost altogether.

Monitoring programs are written using the techniques and commands discussed in Part 3, "FEPI application programming," on page 135. See also the overview of the sample monitoring program in "Monitor and unsolicited data-handler" on page 243.

Handling unexpected events

This section suggests some actions your monitoring program could take after various types of unexpected event.

The type of event is indicated by the EVENTTYPE area in the TD queue record. In most cases, the EVENTVALUE area gives specific details of the failure; the values are the same as the RESP2 values listed in “FEPI RESP2 values” on page 260.

Events in CSZX TD queue records

The CSZX TD queue records report a number of events relating to FEPI resources and sessions.

INSTALLFAIL

A FEPI resource has failed to be installed. This is probably because you are trying to install a duplicate name. This might indicate either a logic error or a possible security violation.

Recommended action: Report possible application logic error, for investigation.

DISCARDFAIL

A FEPI resource has not been discarded. This is probably because you are trying to discard a nonexistent object. This might indicate a logic error.

Recommended action: Report possible application logic error, for investigation.

SETFAIL

A FEPI resource has rejected a SET request. This is probably because you are trying to manipulate a resource that does not exist. However, there is also the possibility of rejection due to VTAM considerations. So SETFAIL might indicate either a logic error or a network failure.

Recommended action: Schedule a transaction to repeat the operation (if not a logic error).

ACQFAIL

A FEPI resource has failed to be acquired. This is probably because of a network failure, and so FEPI automatically retries the acquire request several times at intervals; the count in EVENTDATA shows whether there will be any more retries. However, there is also the possibility of an error in either the VTAM definition or the back-end system definition of the object.

Recommended action: After FEPI stops retrying, suggest investigating the condition of the resource from a VTAM viewpoint. The VTAM sense code describing the problem is in EVENTDATA. See the appropriate VTAM manual for more information. For nodes, this is the *VTAM Programming* manual; for connections, *VTAM Messages and Codes*. Further information is in the *SNA Formats* manual.

SESSION

An unsolicited bind was received, probably because of a CLSDST(PASS). See “Handling CLSDST(PASS)” on page 48.

Events in pool-specific TD queue records

Pool-specific TD queue records report a number of events relating to problems with connections and pool definitions.

SESSIONLOST

An active connection has failed. This is probably due to the back-end system failing. However, this error is also generated if an operator cancels an active connection.

Recommended action: Suggest that the operator:

- Investigate the condition of the connection from a VTAM viewpoint. The VTAM sense code that describes the problem is in EVENTDATA. See the *VTAM Messages and Codes* and *SNA Formats* manuals for more details.
- See whether the back-end system is still running.
- Check that the back-end system has not “closed” the FEPI simulated terminal.

SESSIONFAIL

A connection has failed to start. This is probably due to a setup inconsistency or to a failure of the back-end system, and so FEPI automatically retries the acquire request several times at intervals; the count in EVENTDATA shows whether there will be any more retries. However, this failure is also generated if an operator has canceled the connection.

Recommended action: After FEPI stops retrying, suggest the operator:

- Investigate the condition of the connection from a VTAM viewpoint. The VTAM sense code that describes the problem is in EVENTDATA. See the *VTAM Messages and Codes* and *SNA Formats* manuals for more details.
- See whether the back-end system is still running.
- Check that the back-end system has not “closed” the FEPI simulated terminal.
- Check that the terminal type definition in the back-end matches the FEPI device type.

ADDFAIL

An attempt to add a target or node to a pool has failed. The probable cause of this error is an attempt to add a resource that is already in the pool. This indicates a possible logic error.

Recommended action: Report possible application logic error, for investigation.

DELETEFAIL

An attempt to delete a target or node from a pool has failed. This is probably caused by an attempt to delete a resource that is not in the pool, indicating a possible logic error.

Recommended action: Report possible application logic error, for investigation.

Handling CLSDST(PASS)

A back-end system can end a network session with a VTAM CLSDST(PASS) request.

This indicates that the back-end will reestablish a session with the front-end using a different PLU name (a *third-party* PLU). The front-end system detects reestablishment of the session by receiving an unsolicited bind request; so when the back-end system ends a session, it is important for it to indicate that an unsolicited bind is to be expected.

Note: To determine whether a lost session was caused by a CLSDST(PASS) request, a FEPI application can issue a FEPI INQUIRE CONNECTION command. If the value of LASTACQCODE is X'32020000', the back-end system issued a CLSDST(PASS) to unbind the session.

The three most likely scenarios are described in the following sections.

Unsolicited bind not expected

FEPI unconditionally rejects the bind request.

Third-party PLU name known and unsolicited bind expected

The prospective PLU names must be defined to FEPI as targets.

You might need to restrict access to the pools that include these targets to make sure the connection is not already in use when the CLSDST(PASS) takes place. The simplest way to configure this is to define a pool containing the node and all the targets it can be placed in session with. Install all connections except the initial one with an ACQSTATUS of RELEASED so the back-end system can successfully acquire the session. No other special processing is required and no TD queue record is written in this case.

Third-party PLU name not known and unsolicited bind expected

The necessary resource definitions must be managed dynamically.

Note: Managing the resource definitions dynamically (described under “Conversation in progress”) is the only method that allows the conversation to persist across the CLSDST(PASS).

When FEPI receives the unsolicited bind, it writes a record to the CSZX TD queue, with an EVENTTYPE of SESSION, and with the third-party PLU name in the TARGET area. At this point, the bind has not been accepted or rejected. A VTAM display for either the back-end or the front-end system would show the connection to be in a PSESST/B state. You are responsible for managing these TD queue records and making the necessary FEPI configuration updates so that processing can continue. If no action is taken, the session remains in this state until a VTAM VARY NET,TERM command is issued to terminate the session request.

There are two cases, according to whether or not there is a conversation in progress on the connection when the CLSDST(PASS) occurs. (This can be determined from the STATE option of the FEPI INQUIRE CONNECTION command.) In both cases, you need to determine which pool has the connection that the CLSDST(PASS) applies to, because the TD queue record does not report either the pool or the old target name. If the node is used in only one pool, the old target name can be found easily by browsing connections using FEPI INQUIRE CONNECTION; if not, use some other technique, such as the USERDATA option of the FEPI SET commands.

Conversation in progress:

Nodes for which this kind of processing is required should be defined in pools containing only the node and the initial target, because of the nature of the processing involved.

About this task

The monitor program should:

1. Install a new pool with the same properties as the current one.
2. Install a new target whose PLU name is the third-party PLU name given in the TARGET area of the TD queue record.
3. Add the target to the new pool. This should be the only target in that pool.

4. Delete the node identified in the TD queue record from the pool in which it currently exists. If necessary, to ensure continuity, the monitor program can add another node to the pool before deleting the old node.
5. Add the node to the newly created pool. The new connection is now established.

When the session ends, the connection reverts to a RELEASED state. If necessary, use an end-session handler to perform any necessary cleanup, such as reversing the process described above.

The front-end application must also anticipate CLSDST(PASS) processing. See “Lost session” on page 166 for more details.

Conversation not in progress:

The CLSDST(PASS) occurred as a result of trying to acquire a connection.

About this task

The monitor program should:

1. Install a new target whose PLU name is the third-party PLU name given in the TARGET area of the TD queue record.
2. Add the target to the pool, specifying a desired connection acquire status of ACQUIRED. The new connection is now established.

If necessary, use an end-session handler to cleanup the dynamically defined targets. These connections always become RELEASED when the session ends and can be left for reuse, if required.

Writing operator transactions

You might find it useful to write some specialized operator transactions of your own to control FEPI resources.

For more information, see “Controlling FEPI resources” on page 53.

Other functions

The other functions you might need to write for FEPI itself are the begin-session, end-session, and unsolicited-data handlers.

These are extensions of the FEPI application programs, and are described in Part 3, “FEPI application programming,” on page 135. If you write them as common functions, you need to know what the application programs do. Alternatively, the application programmer may write them.

Global user exit programs

There are two global user exits:

XSZBRQ

Invoked before a FEPI command is executed

XSZARQ

Invoked after a FEPI command is executed.

XSZBRQ is passed the parameters input to the command, and can be used to monitor commands, to bypass commands that violate installation conventions, or to change the parameters of a command, subject to the rules applying to global user exits. XSZARQ is passed the parameters output from the command.

For details of the FEPI global user exits, see Front End Programming Interface exits. For programming information about writing and using global user exit programs, see The user exit programming interface (XPI), in the *CICS Customization Guide*.

Chapter 8. FEPI operation

This section describes how FEPI operates. It includes information on controlling FEPI resources, performance, and shutdown. It also describes using FEPI with XRF and VTAM persistent sessions.

The section contains the following topics:

- “Controlling FEPI resources”
- “FEPI performance” on page 55
- “Shutdown” on page 57
- “Using FEPI with XRF” on page 58
- “Using FEPI with VTAM persistent sessions” on page 62.

Controlling FEPI resources

The FEPI INQUIRE and SET functions can be carried out by a program, or by using the master terminal transaction, CEMT. You may find it useful to write some specialized operator transactions of your own.

The FEPI INQUIRE command (and its CEMT equivalent) tells you what resources are defined and their statuses. The only thing you cannot do directly is determine which nodes and targets are in a particular pool. Do this using CEMT to inquire about the connections in a particular pool:

```
CEMT I FECONNECTION POOL(poolname)
```

To do this from an application program, browse all connections and select those in the pool you want.

Here are the resource statuses of most interest:

SERVSTATUS

SERVSTATUS is used with connections, nodes, pools, and targets.

It specifies the service status of the resource—that is, whether it can be used for a conversation. The service status can be set to `INSERVICE` to allow usage, or to `OUTSERVICE` to stop usage for any *new* conversation. Note that setting `OUTSERVICE` does not end any existing conversations that are using the resource; the status is `GOINGOUT` until the existing conversations end.

ACQSTATUS

ACQSTATUS is used with connections and nodes.

It specifies the “acquire status” of the resource. For a connection, this means whether it should have a session established (bound) or ended (unbound). For a node, it means whether the VTAM ACB for the node should be opened or closed. The acquire status can be set to `ACQUIRED` (a status of `ACQUIRING` indicates that the acquisition has not yet been completed), or to `RELEASED`.

Setting `RELEASED` does not end any existing conversations that are using the resource; the acquire status is `RELEASING` until the existing conversations end.

However, for connections, a conversation that is unowned and in a “pending” state (see “STATE” on page 55) is ended immediately if the acquire state is set to RELEASED; this means that connections being used by a failed application can be recovered.

ACQUIRING and RELEASING are shown as BEING ACQUIRED and BEING RELEASED by CEMT.

Network and other problems can cause connections to become stuck in a RELEASING or ACQUIRING state, in which case the operator might need to intervene using VTAM operator commands.

If a FEPI connection remains in a RELEASING state for longer than expected, try the following:

1. Note the node and target associated with the connection; use CEMT INQUIRE FETARGET to find the VTAM application name that the target represents.

2. Issue the VTAM command

```
D NET,E,ID=nodename
```

to find out the state of network session associated with the connection.

3. Note the session status. See the *VTAM Programming* manual for an explanation of the status. If no session exists and a subsequent INQUIRE of the connection status using CEMT shows the state still as BEING RELEASED, there has been a system failure; you should collect diagnostic information.

4. If the session is in ‘session takedown processing’, you can use the VTAM command

```
D NET,SESSION
```

to find out what signals are needed to complete processing.

5. If you can resolve the problem using commands on the back-end system, attempt to do so.

6. If there is no other way to resolve the session status, you can use the VTAM command

```
V NET,TERM
```

to end the network procedure in progress. FEPI will then be able to complete processing.

It is not so easy to find out when an ACQUIRING state has persisted for too long. However, if you cannot determine why the session has not been established, follow the procedure described above. If no session is active for the connection, FEPI is currently waiting for the retry interval to expire. The system log should contain VTAM messages explaining why the session cannot be established. The LACQCODE option of CEMT INQUIRE FECONNECTION gives the reason code VTAM provided for the last session failure.

Also be sure to check that the node on which the connection depends is properly acquired; if not, resolve whatever problem is indicated by the LACQCODE option for the node.

Note that, under normal circumstances, after a FEPI FREE RELEASE command has been issued the session does not remain in RELEASED state, because FEPI

automatically tries to reacquire the session. However, if a FEPI SET CONNECTION ACQSTATUS(RELEASED) command is issued before the FREE RELEASE, the session remains in RELEASED state.

LASTACQCODE

The INQUIRE CONNECTION or INQUIRE NODE commands can use the option LASTACQCODE (LACQCODE in CEMT), which returns the result of the last acquire request.

This is the sense code from the last VTAM operation, where zero indicates success. For a full explanation of VTAM sense codes, see the appropriate VTAM manual: for nodes, this is *VTAM Programming*; for connections, *VTAM Messages and Codes*. Further information is in the *SNA Formats* manual.

INSTLSTATUS

INSTLSTATUS is used with connections, nodes, pools, and targets. It specifies whether the resource is installed, or is in the process of being discarded, waiting for the conversations that are using it to end.

WAITCONVNUM

WAITCONVNUM shows how many conversations are currently waiting to start using a connection or pool.

If WAITCONVNUM is nonzero for significant periods of time, it might mean that you need to allocate extra resources to meet the demand. Or it might mean that applications are holding on to resources for too long.

STATE

STATE is used with connections.

It shows the state of the conversation that is using a connection. See State for the values that STATE can have.

If any of the “pending” states (PENDSTSN, PENDBEGIN, PENDDATA, PENDSTART, PENDFREE, PENDINGRELEASE, PENDUNSOL, or PENDPASS) is shown, it indicates that the conversation is unowned, pending the event or task shown. If a “pending” state persists, it is likely that the application has failed in some way; you should consider resetting the connection by issuing a FEPI SET CONNECTION RELEASED command.

FEPI performance

You cannot tune FEPI itself—it is already optimized for speed of response.

However, you can influence the performance of FEPI application programs.

FEPI runs under a separate CICS task control block (TCB) and CICS permits only *one* application program to issue a FEPI command at a time. This is a major influence on FEPI performance. Although many application programs can have FEPI commands being processed at any time, only one application can *issue* a FEPI command.

In a lightly loaded system, this means that CICS does not run FEPI until a command is issued. Thus, performance is impacted by the overhead of starting up

the TCB so that the FEPI command can be processed. In a heavily loaded system, this overhead is not present, because the TCB is already active processing earlier FEPI commands. This is in contrast to a traditional CICS system, where a lightly loaded system may perform better than a heavily loaded one.

FEPI tries to minimize this overhead by issuing timer requests that ensure that the TCB is not inactive for more than one second.

There are three main principles that should be used in FEPI applications to provide the best performance:

1. Each FEPI command generates a CICS WAIT even if no network transmission is involved, and so the number of commands issued should be minimized.
2. Data transmission should be kept to a minimum.
3. Session disconnection should be avoided.

Techniques to use in application programs in support of these principles are given in "Performance" on page 171.

As to FEPI system programming, command usage can be reduced by using lists of resources on a command where possible. However, when a command using a list results in a VTAM operation, you could:

- "Flood" VTAM by requesting too many operations at once
- "Flood" the back-end system with requests for session initiation
- "Flood" the front-end system with started begin- or end-session transactions.

So you must carefully evaluate the benefits of using lists.

Using CICS monitoring

CICS monitoring data can help with performance tuning and resource planning for applications that use FEPI.

By default, CICS performance class monitoring records include the following data about the user task:

- The number and type of requests made to FEPI
- The time spent waiting for requests to FEPI to complete
- The number of requests to FEPI that are timed out.

For detailed information about the FEPI-related fields in performance class monitoring records, see Performance data in group DFHFEPI, in the *CICS Performance Guide*. For information about using the DFHMCT TYPE=RECORD macro to control which FEPI fields are monitored, see the Monitoring control table, in the *CICS Resource Definition Guide*.

Using statistics data

CICS statistics can help with performance tuning and resource planning for applications that use FEPI.

The standard CICS statistics reports contain data about usage of:

- FEPI pools
- FEPI connections
- FEPI targets.

To obtain the current statistics for a FEPI pool, connection, or target, a utility program can issue an **EXEC CICS COLLECT STATISTICS** command. For example, the command `EXEC CICS COLLECT STATISTICS SET(pointer) POOL(GRPD)` returns the current statistics for the 'GRP'D' pool. To map the returned statistics, your utility program should include the appropriate CICS-supplied copybook:

DFHA22DS

FEPI pool statistics

DFHA23DS

FEPI connection statistics

DFHA24DS

FEPI target statistics.

The copybooks are supplied in COBOL, PL/I, and assembler language.

To cause all FEPI statistics to be written immediately to the SMF statistics data set, you can use either the EXEC CICS or the CEMT version of the **PERFORM STATISTICS RECORD FEPI** command. For details of the **CEMT COLLECT STATISTICS** and **PERFORM STATISTICS RECORD** commands, see CEMT - master terminal transaction, in the *CICS Supplied Transactions*; for programming information about the equivalent **EXEC CICS** commands, see the *CICS System Programming Reference*.

To format and print FEPI-related statistics in the DFHSTATS data set, you can use the CICS-supplied utility program, DFHSTUP. To print only the FEPI statistics, specify the command parameter `SELECT TYPE=FEPI`. For information about how to use the DFHSTUP program, see Statistics utility program (DFHSTUP), in the *CICS Operations and Utilities Guide*. For detailed information about fields in the FEPI statistics records, see FEPI statistics, in the *CICS Performance Guide*.

Shutdown

FEPI shutdown is triggered as part of CICS shutdown—you cannot shut down FEPI alone.

There are three forms of shutdown:

- Normal
- Immediate
- Forced.

Normal shutdown

A normal shutdown of CICS causes FEPI to shut down normally - active transactions are allowed to terminate. When all active conversations have ended, and all FEPI resources have been discarded, FEPI shuts down.

While FEPI is shutting down, no *new* conversations can be started, but existing owned conversations continue. However, these cannot use the **FEPI START** or **FEPI FREE PASS** commands. Existing unowned conversations are ended immediately. Any FEPI transactions that you want to be able to start during CICS shutdown must be defined in the transaction list table (XLT).

If an end-session handler is invoked at the end of conversations, it is told that the session is to be ended because of CICS shutdown. The handler can choose to perform additional back-end operations that might be needed because of the shutdown. If you require this function, make sure the end-session handler

transaction is defined in the transaction list table (XLT), and that it does not adversely affect the performance of CICS shutdown. (For details of how to define entries in the XLT, see the *CICS Resource Definition Guide*.)

CICS normal shutdown waits until FEPI shutdown has completed before continuing processing. So if you know when CICS shutdown is to occur, you should initiate FEPI DISCARD operations before starting CICS termination. Removing FEPI resources as they become inactive allows existing FEPI conversations to continue, but prevents new ones from starting. You could achieve the same effect by setting the status of FEPI resources to `OUTSERVICE,RELEASED`.

If shutdown is not proceeding, then before you force it to continue, consider carefully whether the problem is due to:

- A back-end system taking a long time to respond. In this case, do not attempt to speed things up—you may generate integrity errors in the back-end system.
- A FEPI failure. In this case, issue the following commands, pausing after each step to see whether CICS is still waiting:
 1. `CEMT DISCARD FExxxx(*)`, to remove all FEPI resources
 2. `CEMT SET FECONNECTION(*) OUTSERVICE RELEASED`, to end any waiting conversations
 3. `CEMT SET TASK(nnn) FORCE`, to end any running FEPI transactions
 4. Attempt to issue **VTAM VARY NET, INACT, FORCE** commands from the system console to terminate connections.

If CICS shutdown still does not proceed, you cannot perform a warm shutdown. Try issuing a `CEMT P SHUT IMMEDIATE` command. If this fails, you must cancel CICS.

Immediate shutdown

An immediate shutdown of CICS immediately terminates FEPI. There is nothing you can do to influence this process.

Forced shutdown

A forced shutdown of CICS immediately terminates FEPI. There is nothing you can do to influence this process.

Using FEPI with XRF

This section discusses FEPI in a CICS extended recovery facility (XRF) environment.

To understand it, you need to have read the *CICS/ESA Version 3.3 XRF Guide*, and to be familiar with CICS XRF VTAM USERVAR processing—the *VTAM Programming* manual contains relevant material.

The effect of an XRF takeover of a CICS back-end system with which FEPI is in communication is described. Although IMS XRF processing is not discussed here, the same considerations apply.

XRF and VTAM

FEPI uses VTAM secondary LU support for communication and the simulated terminals defined to the back-end CICS system behave in a different way to real devices.

In an XRF environment, the simulated terminals in the back-end system cannot behave as VTAM class 1 terminals because there is no 3745/3725/3720 controller acting as the boundary network node (BNN). They behave like VTAM class 2 terminals, which is the default setting for CICS and IMS terminal definitions. Consequently, simulated terminals do not support VTAM XRF, and CICS XRF facilities are provided by tracking mechanisms that are explained in the *CICS/ESA Version 3.3 XRF Guide*.

When a FEPI connection is acquired, the back-end CICS generates a TCTTE (if one is not present already) using autoinstall. At this point, in a CICS XRF environment, the active CICS informs the alternate that a terminal has been defined. If the active is then taken over, the alternate knows which terminals are defined, and can take actions to recover the links.

As part of takeover processing, a VTAM BIND is issued to reestablish the session with each simulated terminal. However, FEPI also has detected that the connection has ended, and attempts to contact the (new active) back-end system by issuing a similar bind. This results in a **bind race**. The outcome of this bind race depends on the circumstances of the exchange. However, the bind issued by the new active CICS will probably be rejected, and the FEPI bind accepted. This results in DFHZCxxx messages being produced during the takeover (see “Connections with a conversation—with data flow” on page 61). If FEPI reestablishes the connection, these messages can be ignored. You can remove these bind races by defining the back-end CICS terminal so it behaves as a VTAM class 3 terminal (no XRF support). To define the simulated terminals as class 3, specify RECOVPTION=NONE in CICS, or BACKUP=NO in IMS.

FEPI resource definition and XRF

In an XRF environment, the applid specified on the FEPI INSTALL TARGETLIST command must be the *generic* applid of the back-end system.

Specifying either the primary or secondary applid of the target results in processing errors. If you use the generic applid, FEPI is able to cater for the back-end system undergoing an XRF takeover.

However, you can define a pool that contains the specific applids of both the active and alternate systems. In this case, the alternate targets cannot be contacted until an XRF takeover has been performed. Similarly, the active targets cannot be contacted after takeover. If you define pools in this way (perhaps to provide backup support without XRF), you should manage the ACQUIRED-RELEASED status yourself, to minimize FEPI retry processing.

XRF takeover of front-end system

This section describes what happens when the CICS system running FEPI undergoes an XRF takeover.

Effect on back-end transactions

Each back-end transaction is abended, due to the loss of the simulated terminal—which is usually the principal facility for the task. Consequently, the ATNI (or equivalent)abend processing is unable to send the usual message indicating a transaction abend to the principal facility.

Transactions that attempt to handle terminal control errors should already be written to cope with this circumstance, and you should not need to alter them.

Effect on back-end terminals

FEPI is acting as the “terminal”, so an XRF takeover of the FEPI system results in the loss of the “terminal” in the back-end system. CICS takes the usual actions for the loss of a (real) terminal. There are three cases to consider:

“Terminals” without a conversation:

If you are using autoinstall, the TCTTEs representing these “terminals” are deleted after a delay; if the delay is long enough, the alternate front-end CICS may reestablish the sessions before the TCTTEs are deleted.

“Terminals” with a conversation—no data flow:

If you are using autoinstall, the TCTTEs representing these “terminals” are deleted after a delay; if the delay is long enough, the alternate front-end CICS may reestablish the sessions before the TCTTEs are deleted.

“Terminals” with a conversation—with data flow:

These “terminals” are usually running a transaction when the “terminal” is lost. This results in the transaction being abended with the normal CICS abend code for a terminal failure (usually ‘ATNI’). The abend is usually accompanied by a DFHZCxxxx message indicating that the “terminal” has suffered an unrecoverable failure.

You may have to modify your node error program to prevent retry loops, but normally the default action (not to retry) is taken. When node error processing ends, if autoinstall is used, the “terminal” is deleted.

Effect on the alternate FEPI CICS system

The alternate FEPI CICS takes over operation of the failed CICS in the normal fashion. However, FEPI resources are not recovered automatically after an XRF takeover.

FEPI restarts at a late stage of takeover, after all RDO resources have been reinstalled. Nevertheless, when the second phase of the PLTPI list is entered, FEPI is ready to receive commands. Therefore, if you follow the recommendation to start your FEPI setup transaction from a PLTPI program, FEPI resources are reinstalled as part of the takeover. If you do *not* run your setup transaction in this way, then after a takeover you must arrange for it to be run manually, so that your FEPI resources are reinstalled.

However you handle resource definition in an XRF-environment, you must be prepared to cope with the possibility that FEPI resources have been manipulated in the failed CICS, so that the environment after takeover is not the same as that immediately before takeover. For example, resources may have been installed or deleted, or SERVSTATUS or ACQSTATUS values altered, after your setup transaction was run in the failed CICS.

XRF takeover of back-end system

This section describes what happens when the CICS back-end system with which FEPI is communicating undergoes an XRF takeover.

Effect on FEPI application programs

FEPI application programs are unable to distinguish between a loss of session due to an XRF takeover of the back-end system, and one due to a FEPI failure.

In both cases, a typical RESP2 value of '215' ('Session lost') is returned on the next FEPI command issued after the takeover has started. Alternatively, the application may get an indication of a state error, meaning that the command cannot be issued because the connection is not active. The application should *immediately* issue a FEPI FREE command to free the conversation.

If an end-session handler is active, it gets invoked, even though the conversation has ended.

If the application program believes that the back-end is undergoing an XRF takeover, it should reissue a FEPI ALLOCATE command for the back-end. When the takeover is complete, and FEPI has reestablished contact, the FEPI ALLOCATE completes successfully (together with any specified begin-session processing). If the TIMEOUT option is used, consider its setting in relation to how long you expect the alternate back-end system to take to complete takeover.

It is the responsibility of the application program to perform any processing in the new active back-end system necessitated by the XRF takeover.

Effect on FEPI connections

In general, FEPI successfully copes with the XRF takeover of a back-end system with which it is communicating.

However, when the new active back-end system attempts to establish its terminal sessions, communication with FEPI may result in some strange terminal control messages. You should ignore these until FEPI has had time to contact the back-end system.

While FEPI is attempting to reestablish contact with the back-end system:

- Connections are in ACQUIRING state, with a last acquire code of (probably) X'320C0000'.
- Message DFHSZ4155I may be produced, with reason codes (typically X'320C0000' or X'81062900') showing that FEPI is attempting to reestablish contact with the back-end system.

There are three cases to consider:

Connections without a conversation:

These connections reestablish contact with the new active back-end when the back-end's ACB is opened.

Connections with a conversation—no data flow:

These connections reestablish contact with the new active back-end when the back-end's ACB is opened. You may get some messages in the back-end system indicating that the TCTTE was deleted and reinstalled.

Connections with a conversation—with data flow:

These connections generate errors in the back-end system when it attempts to reestablish contact with the "terminal".

You may see messages DFHZC3492E, DFHZC2411E, DFHZC3422E, DFHZC3437I, or DFHZC3462I being generated—all of which say that the standby back-end could not reestablish contact with the "terminal". However, as long as the conversation

that was running on the connection has been freed, FEPI subsequently reestablishes contact and reinstalls the “terminal”.

Using FEPI with VTAM persistent sessions

When creating FEPI applications, you need to be aware of the possible effects of the use of VTAM persistent sessions in the front- or back-end systems.

For information about CICS support for VTAM persistent sessions, see the *CICS Recovery and Restart Guide*.

Restart of front-end system using persistent sessions

Using persistent sessions in the front-end does not give FEPI any additional recoverability benefits. FEPI is always cold started; thus, to FEPI, the effect of restarting a front-end system for which persistent sessions support is enabled is indistinguishable from a cold start of CICS.

Restart of back-end system using persistent sessions

In the back-end system, there are terminal definitions that are used when the FEPI simulated terminals establish sessions with the target.

These definitions may be hard-coded, or may be autoinstall model definitions. If the terminal definitions have been set up to use persistent session support, and the back-end system is restarted within the persistent session delay interval, the terminal sessions are recovered.

Effect on FEPI application programs

It is likely that FEPI application programmers have little say in the way that persistent session support is used in the back-end system.

They therefore need to be aware of the different ways in which terminal sessions can be recovered, so that their applications cater for all possibilities. If the back-end (target) is a CICS Transaction Server for z/OS, Version 3 Release 2 system, the way in which a session is recovered depends on the setting of the RECOVOPTION and RECOVNOTIFY options of the TYPETERM definition.

RECOVOPTION(SYSDEFAULT)

On restart within the persistent session delay interval, CICS selects the optimum procedure to recover a session.

For LU2, if the session is busy and CICS is in send mode, CICS sends an end bracket. If the session is busy and CICS is not in send mode, CICS sends an SNA CLEAR request to reset the conversation state.

If a FEPI conversation is in progress when the target system terminates, your application could see one of the following:

- A timeout on a RECEIVE, CONVERSE, or START command, while it waits for the target to restart.
Deal with this in the normal way for a timeout.
- A FEPI RECEIVE or CONVERSE command completes as a result of the end bracket sent by CICS. The RU on this data flow may be empty or may contain a user-defined message, depending on the value of the RECOVNOTIFY option.

Your application may need to perform some backout processing.

- An INVREQ response with a RESP2 value of 230 on a FEPI SEND, RECEIVE, CONVERSE, ISSUE, or START command, indicating that an SNA CLEAR was received.

Your application may need to perform some backout processing.

You must also consider the value specified for RECOVNOTIFY:

RECOVNOTIFY(MESSAGE)

A message (defined in the BMS maps DFHXRC3 and DFHXRC4) is sent to the “terminal”. Your FEPI application must contain logic to deal with this data flow.

If there is no active conversation at the time of restart, the flow is received as unsolicited data at the FEPI front-end.

RECOVNOTIFY(TRANSACTION)

A transaction is initiated in the target. The default is the Good Morning transaction. Your application must contain logic to deal with this data flow.

If there is no active conversation at the time of restart, the flow is received as unsolicited data at the FEPI front-end.

RECOVNOTIFY(NONE)

The “terminal” is not notified that a restart has occurred. Your application need take no special action.

RECOVOPTION(CLEARCONV)

On restart within the persistent session delay interval, CICS sends an SNA CLEAR request to reset the conversation states. The CLEAR is sent only if the session was busy at the time of system restart. If a FEPI conversation is in progress when the target system terminates, your application could see one of the following:

- A timeout on a RECEIVE, CONVERSE, or START command, while it waits for the target to restart.
Deal with this in the normal way for a timeout.
- An INVREQ response with a RESP2 value of 230 on a FEPI SEND, RECEIVE, CONVERSE, ISSUE, or START command, indicating that an SNA CLEAR was received.

Your application may need to perform some backout processing.

You must also consider the value specified for RECOVNOTIFY. The possible values are as described above, for RECOVOPTION(SYSDEFAULT).

RECOVOPTION(RELEASESESS)

On restart within the persistent session delay interval, CICS sends an UNBIND request to release an active session. The request is sent only if the session was busy at the time of system restart.

If a FEPI conversation is in progress when the target system terminates, your application could see one of the following:

- A timeout on a RECEIVE, CONVERSE, or START command, while it waits for the target CICS to restart.
Deal with this in the normal way for a timeout.
- An INVREQ response with a RESP2 value of 215 on any FEPI command, indicating a 'session lost' condition.

Deal with this in the normal way for a session loss.

RECOVOPTION(UNCONDREL)

On restart within the persistent session delay interval, CICS sends an UNBIND

request to release an active session. The request is sent whether or not the session was busy at the time of system restart.

If a FEPI conversation is in progress when the target system terminates, your application could see either of the symptoms described for RECOVPTION(RELEASESESS).

RECOVPTION(NONE)

Even if the system is restarted within the persistent session delay interval, the session is not recovered—it has no persistent session support.

Deal with this in the normal way for a session loss.

Chapter 9. Operator control of FEPI

Two CICS-supplied transactions, CEMT and CETR, provide operator control of FEPI: you can use the **CEMT INQUIRE**, **SET**, and **DISCARD** commands to control FEPI resources such as nodes, targets, and pools; and the CETR transaction to control FEPI trace.

You can also use VTAM commands to manage communication with target systems.

FEPI application programs, and the CICS resources they use, are controlled just like other CICS applications and resources.

CEMT - master terminal transaction

The CEMT transaction has a range of commands that support FEPI.

These commands, which are described below, work exactly like other CEMT commands - for example, in supporting resource selection by families (AB*, for example), lists (AB,CD,EF, for example), and by subdefining groups. Note that 4-character option names are used in the display.

CEMT DISCARD

The **CEMT DISCARD** command removes targets, nodes, pools, or property sets completely from FEPI.

Syntax

Press the Clear key to clear the screen. Type CEMT DISCARD (the minimum abbreviation is CEMT DISC), followed by any one of:

- FENODE(*nodename*)
- FEPOOL(*poolname*)
- FEPROPSET(*propsetname*)
- FETARGET(*targetname*).

For example, `cemt disc fen(fepnode1)` removes the node *fepnode1* from FEPI.

Typing ? at the beginning of either the first or second line gives a syntax prompt.

CEMT DISCARD

```
▶▶ CEMT DISCard  FENode—(—nodename—)————▶▶
                  FEPOol—(—poolname—)————▶▶
                  FEPRopset—(—propsetname—)——▶▶
                  FETarget—(—targetname—)——▶▶
```

Options

FENode(*nodename*)

The name of the FEPI node to be discarded.

FEPO01 (*poolname*)

The name of the FEPI pool to be discarded.

FEPropset (*propsetname*)

The name of the FEPI property set to be discarded.

FETarget (*targetname*)

The name of the FEPI target to be discarded.

CEMT INQUIRE FECONNECTION

Display information about FEPI connections.

Description

INQUIRE FECONNECTION displays information about the state of FEPI connections. A connection is identified by specifying the target and node. The results are given in order of target within the node. Family selection can be used for TARGET and NODE, but list selection cannot be used.

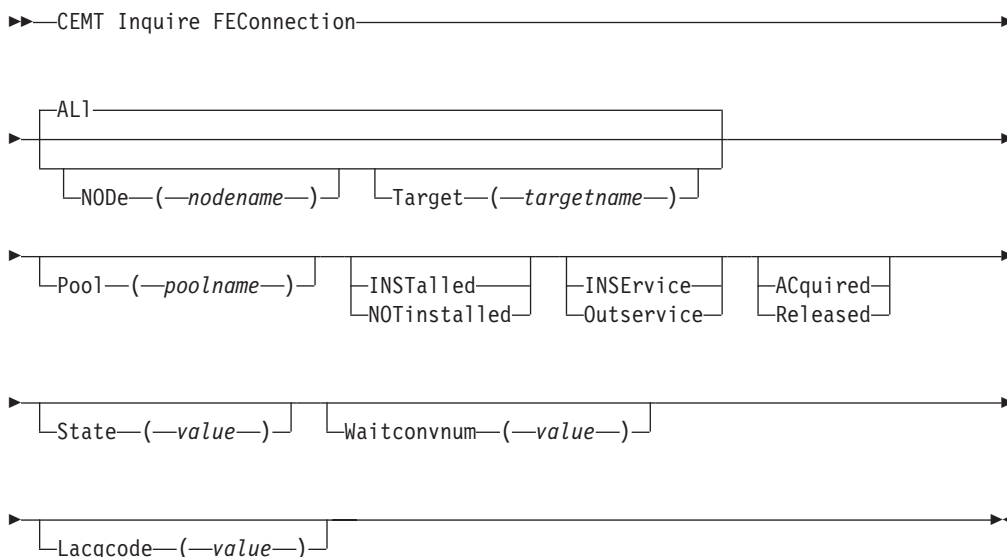
Input

Press the Clear key to clear the screen. There are two ways of commencing this transaction:

- Type CEMT INQUIRE FECONNECTION (the minimum abbreviation is CEMT I FEC). You get a display that lists the current status.
- Type CEMT INQUIRE FECONNECTION (CEMT I FEC) followed by as many of the other attributes as are necessary to limit the range of information that you require. For example, if you enter `cemt i fec p(pool5) acq`, the resulting display will show you the details of all FEPI connections in *pool5* on which sessions are bound.

You can tab to the highlighted fields and overtype them with new values.

CEMT INQUIRE FECONNECTION



ALL

is the default. Information about all connections is given, unless you specify a selection.

NODE(nodename)

is the 8-character name of a node. Information is restricted to connections of which this node forms part.

Target(targetname)

is the 8-character name of a target. Information is restricted to connections of which this target forms part.

Sample screen

```
CEMT IN FEC
STATUS: RESULTS - OVERTYPE TO MODIFY
Node(NODE1 ) Targ(TARGETA ) Pool(POOL5 ) Inst Inse Rele
  Stat(NOCONV ) Wait(00000) Lacq(X'08570002')
Node(NODE1 ) Targ(TARGETB ) Pool(POOL5 ) Inst Inse Rele
  Stat(NOCONV ) Wait(00000) Lacq(X'08570002')
Node(NODE1 ) Targ(TARGET3 ) Pool(POOL3 ) Inst Inse Rele
  Stat(NOCONV ) Wait(00000) Lacq(X'08570002')
```

Figure 3. CEMT INQUIRE FECONNECTION screen

Displayed fields**Node(value)**

displays the 8-character name of a node identifying a connection.

Target(value)

displays the 8-character name of a target identifying a connection.

Pool(poolname)

displays the 8-character name of a pool of connections.

Installed|Not installed

displays a value identifying the install state of the connection. The values are:

Installed

The connection is in a pool that has been defined by INSTALL and is available for use.

Not installed

The connection is in a pool, or involves a node or target that is being discarded, but is still in use.

Inservice|Outservice

displays a value identifying the service state of the connection. The values are:

Inservice

The connection is in service and can be used in a conversation. If OUTSERVICE state has been requested but has not yet completed, a 'GOING OUT' message is shown.

Outservice

The connection is out of service and cannot be used for any conversation.

Acquired|Released

displays a value identifying whether a session on the connection is bound. The values are:

Acquired

A session is bound on the connection. If RELEASED state has been requested but has not yet completed, a 'BEING RELEASED' message is shown. If this persists, you might need to use VTAM commands to recover the connection.

Released

Sessions involving the connection have been unbound. If ACQUIRED state has been requested but has not yet completed, a 'BEING ACQUIRED' message is shown. If this persists, you might need to use VTAM commands to recover the connection.

State(value)

displays a 12-character value identifying the state of the conversation using the connection. The values are:

APPLICATION

A normal application task owns the conversation

BEGINSESSION

A begin-session handling task owns the conversation

FREE An end-session handling task owns the conversation, following a FEPI FREE command

NOCONV

No conversation is active on the connection

PENDBEGIN

A begin-session handling task has been scheduled

PENDDATA

FEPI is waiting for inbound data, following a FEPI START command

PENDFREE

An end-session handling task has been scheduled, following a FEPI FREE command

PENDPASS

The conversation is unowned, following a FEPI FREE PASS command

PENDRELEASE

An end-session handling task has been scheduled, following an unbind request

PENDSTART

Inbound data having arrived, a task specified by FEPI START has been scheduled

PENDSTSN

An STSN-handling task has been scheduled

PENDUNSOL

An unsolicited-data handling task has been scheduled

RELEASE

An end-session handling task owns the conversation, following an unbind request

STSN An STSN-handling task owns the conversation

UNSOLDATA

An unsolicited-data handling task owns the conversation.

The “pending” states indicate the conversation is unowned, pending the event or task indicated. If a “pending” state persists, it is likely that the application has failed in some way; you should consider resetting the connection by issuing a CEMT SET FECONNECTION RELEASED command.

Waitconvnum(value)

displays a value identifying the number of conversations that are waiting to start using a connection. (If a conversation could use any one of several connections, it is counted as waiting on each one.)

Lacqcode(value)

displays a hexadecimal value indicating the result of the last acquire request for the node; that is, the sense code from the last VTAM REQSESS, a zero indicating success. For information about VTAM sense codes, see either the *VTAM Messages and Codes* or the *SNA Formats* manual.

CEMT INQUIRE FENODE

Display information about a FEPI node.

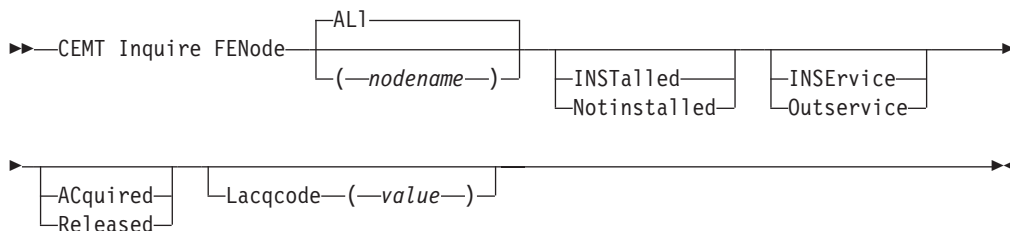
Input

Press the Clear key to clear the screen. There are two ways of commencing this transaction:

- Type CEMT INQUIRE FENODE (the minimum abbreviation is CEMT I FEN). You get a display that lists the current status.
- Type CEMT INQUIRE FENODE (CEMT I FEN) followed by as many of the other attributes as are necessary to limit the range of information that you require. For example, if you enter `cemt i fen inst`, the resulting display will show you the details of all FEPI nodes that have been installed and are ready for use.

You can tab to the highlighted fields and overtype them with new values.

CEMT INQUIRE FENODE



AL1

is the default. Information about all nodes is given, unless you specify a node.

nodename

is the 8-character name of the node to be queried.

Sample screen

```
CEMT IN FEN
STATUS: RESULTS - OVERTYPE TO MODIFY
Feno(NODE1 ) Inst Inse Acqu Lacq(X'00000000')
Feno(NODE2 ) Inst Inse Acqu Lacq(X'00000000')
Feno(NODE3 ) Inst Inse Acqu Lacq(X'00000000')
Feno(NODE4 ) Inst Inse Acqu Lacq(X'00000000')
```

Figure 4. CEMT INQUIRE FENODE screen

Displayed fields

Feno

indicates that this panel relates to an FENODE inquiry.

(value)

displays the 8-character name of a node.

Installed|Not installed

displays a value identifying the install state of the node. The values are:

Installed

The node has been defined by INSTALL and is available for use.

Not installed

The node is being discarded, but is still in use.

Inservice|Outservice

displays a value identifying the service state of the node. The values are:

Inservice

The node is in service and can be used in a conversation. If OUTSERVICE state has been requested but has not yet completed, a 'GOING OUT' message is shown.

Outservice

The node is out of service and cannot be used for any conversation.

Acquired|Released

displays a value identifying whether the state of the VTAM ACB for the node. The values are:

Acquired

The VTAM ACB for the node is open and the VTAM 'set logon start' command has completed. If RELEASED state has been requested but has not yet completed, a 'BEING RELEASED' message is shown. If this persists, you might need to use VTAM commands to recover the node.

Released

The VTAM ACB is closed. If ACQUIRED state has been requested but has not yet completed, a 'BEING ACQUIRED' message is shown. If this persists, you might need to use VTAM commands to recover the node.

Lacqcode(value)

displays a hexadecimal value indicating the result of the last acquire request for the node; that is, the sense code from the last VTAM OPEN ACB, a zero indicating success. For information about VTAM sense codes, see either the *VTAM Messages and Codes* or the *SNA Formats* manual.

Installed|Notinstalled

displays a value identifying the install state of the pool. The values are:

Installed

The pool has been defined by INSTALL and is available for use.

Notinstalled

The pool is being discarded, but is still in use.

Inservice|Outservice

displays a value identifying the service state of the pool. The values are:

Inservice

The pool is in service and can be used in a conversation. If OUTSERVICE state has been requested but has not yet completed, a 'GOING OUT' message is shown.

Outservice

The pool is out of service and cannot be used for any conversation.

Device(value)

displays a value identifying the mode of conversation and the type of device. The values are:

T3278M2

SLU2 mode, 3278 Model 2

T3278M3

SLU2 mode, 3278 Model 3

T3278M4

SLU2 mode, 3278 Model 4

T3278M5

SLU2 mode, 3278 Model 5

T3279M2

SLU2 mode, 3279 Model 2B

T3279M3

SLU2 mode, 3279 Model 3B

T3279M4

SLU2 mode, 3279 Model 4B

T3279M5

SLU2 mode, 3279 Model 5B

TPS55M2

SLU2 mode, PS/55, 24 lines

TPS55M3

SLU2 mode, PS/55, 32 lines

TPS55M4

SLU2 mode, PS/55, 43 lines

LUP SLU P mode, all cases

Waitconvnum(value)

displays a value identifying the number of conversations that are waiting to start using a connection in the pool.

CEMT INQUIRE FEPROPSET

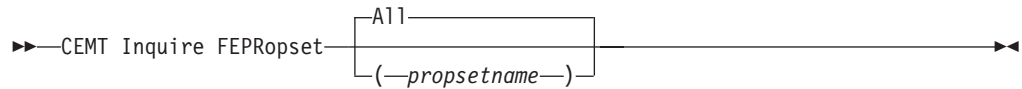
Display information about a set of FEPI properties.

Input

Press the Clear key to clear the screen. There are two ways of commencing this transaction:

- Type CEMT INQUIRE FEPROPSET (the minimum abbreviation is CEMT I FEPR). You get a display that lists all FEPI property sets that are currently installed.
- Type CEMT INQUIRE FEPROPSET (CEMT I FEPR) followed by the name of a particular property set. For example, if you enter `cemt i fepr (feprop1)`, the resulting display will show you whether or not the FEPI property set *feprop1* is installed. (If it is not installed, you get a 'NOT FOUND' response.)

CEMT INQUIRE FEPROPSET



A11

is the default. Information about all property sets is given, unless you specify a particular one.

propsetname

is the name of the property set to be queried.

Sample screen

```
CEMT IN FEPR
STATUS: RESULTS
Fepr(PROP1 )
Fepr(PROP2 )
Fepr(PROP3 )
Fepr(PROP4 )
```

Figure 6. CEMT INQUIRE FEPROPSET screen

Displayed fields

Fepr

indicates that this panel relates to an FEPROPSET inquiry.

(value)

displays the 8-character name identifying a property set.

CEMT INQUIRE FETARGET

Display information about the state of FEPI targets.

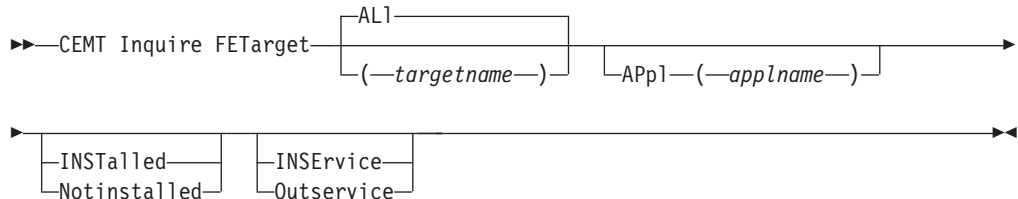
Input

Press the Clear key to clear the screen. There are two ways of commencing this transaction:

- Type CEMT INQUIRE FETARGET (the minimum abbreviation is CEMT I FET). You get a display that lists the current status.
- Type CEMT INQUIRE FETARGET (CEMT I FET) followed by as many of the other attributes as are necessary to limit the range of information that you require. For example, if you enter `cemt i fet inse`, the resulting display will show you the details of all FEPI targets that are in service.

You can tab to the highlighted 'service state' field and overwrite it with a new value.

CEMT INQUIRE FETARGET



AL1

is the default. Information about all targets is given, unless you specify the target to be queried.

targetname

is the name of the target to be queried.

Sample screen

```

CEMT IN FET
STATUS: RESULTS - OVERTYPE TO MODIFY
Feta(TARGETA ) App1(APPL5 ) Inst Inse
Feta(TARGETB ) App1(APPL6 ) Inst Inse
Feta(TARGET1 ) App1(APPL1 ) Inst Inse
Feta(TARGET2 ) App1(APPL2 ) Inst Inse
Feta(TARGET3 ) App1(APPL3 ) Inst Inse
Feta(TARGET4 ) App1(APPL4 ) Inst Inse
  
```

Figure 7. CEMT INQUIRE FETARGET screen

Displayed fields

Feta

indicates that this panel relates to an FETARGET inquiry.

(value)

displays the 8-character name identifying a target.

App1 (applname)

displays the 8-character VTAM application name of the back-end system that the target represents.

Installed|Notinstalled

displays a value identifying the install state of the target. The values are:

Installed

The target has been defined by INSTALL and is available for use.

Notinstalled

The target is being discarded, but is still in use.

Inservice|Outservice

displays a value identifying the service state of the target. The values are:

Inservice

The target is in service and can be used in a conversation. If OUTSERVICE state has been requested but has not yet completed, a 'GOING OUT' message is shown.

Outservice

The target is out of service and cannot be used for any conversation.

CEMT SET FECONNECTION

Change the state of FEPI connections. Family selection can be used for TARGET and NODE, but list selection cannot be used.

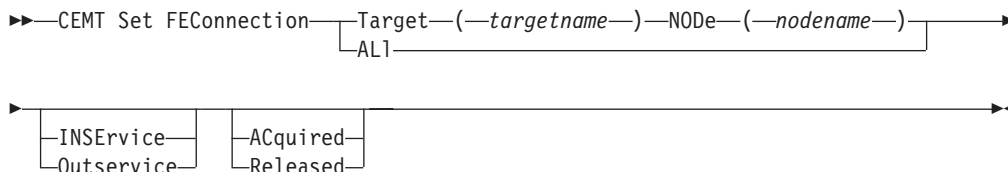
Syntax

Press the Clear key to clear the screen. There are two ways of commencing this transaction:

- Type CEMT SET FECONNECTION (the minimum abbreviation is CEMT S FEC) with either TARGET(*targetname*) NODE(*nodename*) or ALL. You get a display that lists the current status, similar to that obtained by CEMT INQUIRE FECONNECTION. You can tab to the highlighted fields and overtype them with new values.
- Type CEMT SET FECONNECTION (CEMT S FEC) with either TARGET(*targetname*) NODE(*nodename*) or ALL, followed by one or more attribute settings that you want to change. For example, `cemt s fec al ac` causes sessions to be bound for all FEPI connections.

Typing ? at the beginning of either the first or second line gives a syntax prompt. Resetting the values takes effect immediately.

CEMT SET FECONNECTION



Options

ACquired

specifies that the connection is to have a session established (that is, 'bound'). The state is ACQUIRING until this is completed.

ALL

specifies that any change you request is made to all connections that you are authorized to access.

INService

specifies that the connection is to be put in service and can be used in a conversation.

NODe(nodename)

specifies the 8-character name of the node identifying a connection.

Outservice

specifies that the connection is to be put out of service and not to be used for any new conversations, though existing conversations are unaffected. The service state is GOINGOUT until these conversations end.

Released

specifies that the connection is to have its session ended (that is, 'unbound'), when usage of the connection by all owned conversations ends. (An unowned conversation on the connection is ended immediately.) The state is RELEASING until this is completed.

Target(targetname)

specifies the 8-character name of the target identifying a connection.

CEMT SET FENODE

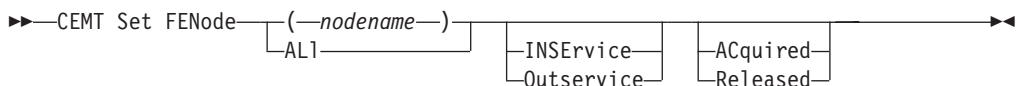
Change the state of FEPI nodes.

Syntax

Press the Clear key to clear the screen. There are two ways of commencing this transaction:

- Type CEMT SET FENODE (the minimum abbreviation is CEMT S FEN) with either a *nodename* or ALL. You get a display that lists the current status, similar to that obtained by CEMT INQUIRE FENODE. You can tab to the highlighted fields and overtype them with new values.
- Type CEMT SET FENODE (CEMT S FEN) with either a *nodename* or ALL, followed by one or more attribute settings that you want to change. For example, `cemt s fen al ac` causes the VTAM ACBs for all FEPI nodes to be opened, and 'set logon start' to be done.

Typing ? at the beginning of either the first or second line gives a syntax prompt. Resetting the values takes effect immediately.

CEMT Set FENode

Options

ACquired

specifies that the VTAM ACB for the node should be opened, and 'set logon start' is to be done. The state is ACQUIRING until this is completed.

ALL

specifies that any change you request is made to all nodes that you are authorized to access.

INService

specifies that the node is in service and can be used in a conversation.

(nodename)

specifies the 8-character name of the node whose state is to be changed.

Outservice

specifies that the node is to be put out of service and not to be used for any new conversations, though existing conversations are unaffected. The service state is GOINGOUT until these conversations end.

Released

specifies that the VTAM ACB for the node is to be closed, when usage of the node by any conversation ends. The state is RELEASING until this is completed.

CEMT SET FEPOOL

Change the state of FEPI pools of connections.

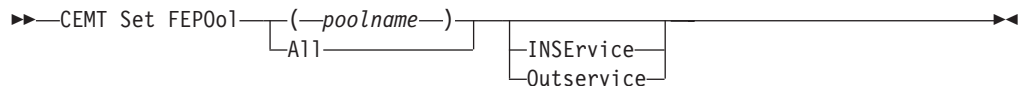
Syntax

Press the Clear key to clear the screen. There are two ways of commencing this transaction:

- Type CEMT SET FEPOOL (the minimum abbreviation is CEMT S FEPO) with either a *poolname* or ALL. You get a display that lists the current status, similar to that obtained by CEMT INQUIRE FEPOOL. You can tab to the highlighted 'service state' field and overtyping it with a new value.
- Type CEMT SET FEPOOL (CEMT S FEPO) with either a *poolname* or ALL, followed by a service state setting. For example, `cemt s fepo fepool1 i` specifies that the *fepool1* pool is in service and available for use by a conversation.

Typing ? at the beginning of either the first or second line gives a syntax prompt. Resetting the values takes effect immediately.

CEMT SET FEPOOL



Options

ALL

specifies that any change you request is made to all pools that you are authorized to access.

INService

specifies that the pool is in service and can be used in a conversation.

Outservice

specifies that the pool is to be put out of service and not be used for any new conversations, though existing conversations are unaffected. The service state is GOINGOUT until these conversations end.

(poolname)

specifies the pool of connections to be changed.

CEMT SET FETARGET

Change the state of FEPI targets.

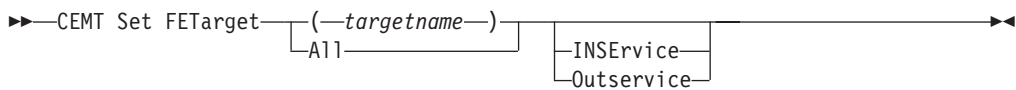
Syntax

Press the Clear key to clear the screen. There are two ways of commencing this transaction:

- Type CEMT SET FETARGET (the minimum abbreviation is CEMT S FET) with either a *targetname* or ALL. You get a display that lists the current status, similar to that obtained by CEMT INQUIRE FETARGET. You can tab to the highlighted 'service state' field and overtype it with a new value.
- Type CEMT SET FETARGET (CEMT S FET) with either a *targetname* or ALL, followed by a service state setting. For example, `cemt s fet fetarg1 i` specifies that the *fetarg1* target is in service and available for use by a conversation.

Typing ? at the beginning of either the first or second line gives a syntax prompt. Resetting the values takes effect immediately.

CEMT Set FETarget



Options

ALL

specifies that any change you request is made to all targets that you are authorized to access.

INService

specifies that the target is in service and can be used in a conversation.

Outservice

specifies that the target is out of service and cannot be used for any new conversations, though existing conversations are unaffected. The service state is GOINGOUT until these conversations end.

(targetname)

specifies the 8-character name of the target to be changed.

VTAM commands

In addition to the resource control facilities provided by FEPI, you can use specific VTAM commands to manage communication with target systems.

They are particularly useful where there are problems in acquiring or releasing sessions; see "ACQSTATUS" on page 53.

These commands are fully described in the *VTAM Operations* manual, but are summarized here. You can:

- Use the VTAM DISPLAY command to inquire about the status of the FEPI nodes (acting as SLUs) and the target systems. It should normally be necessary to use this command only when you experience problems in communicating with a particular target. Note that to understand the displays you require some knowledge of how VTAM operates. VTAM messages are explained in the *VTAM Messages and Codes* manual.
- Use the VTAM VARY command to control the availability of resources within the network. In the case of FEPI, you can use it to force the closure of a node

regardless of whether it is being used in an active conversation. This is achieved by making the VTAM node inactive. However, any pending request to change to a state of RELEASED or OUTSERVICE is able to complete. A subsequent VARY ACTIVE command makes the node available for use again (if its state is still INSERVICE).

- Use the VTAM VARY TERM command to terminate individual connections—that is, to end the session between a particular PLU (target) and SLU (FEPI) pair.
- Use the VTAM DISPLAY SESSIONS command to diagnose problems in establishing sessions. To use this command, you require an understanding of VTAM session processing.

Chapter 10. Customizing FEPI

This section outlines the customization features of FEPI.

It contains:

- “FEPI journaling” on page 85.

It assumes that you are aware of the customization features of CICS (programming information about these is in the CICS Customization Guide).

This section contains Product-sensitive Programming Interface information.

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
- Load balancing
- 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 5 on page 84.

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 time-out 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 5 on page 84.

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 5 on page 84. relates the hexadecimal values passed in UEPSZACT and UEPSZACN to the FEPI commands they represent.

Table 5. 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.

FEPI journaling

This section describes the format of FEPI journal records, and how to print them.

For background information about CICS journaling, you should refer to *Reading log streams using batch jobs*, in the *CICS Operations and Utilities Guide*; for programming information, see *CICS logging and journaling*, in the *CICS Customization Guide*.

Related tasks

Reading log streams using batch jobs (for example, DFHJUP)

FEPI journal operation

You can request FEPI to write inbound, outbound, or both inbound and outbound data to a specified CICS user journal; you cannot write to the system log. This is done using the **MSGJRNL**, **FJOURNALNUM**, and **FJOURNALNAME** options in your property set definitions.

Of the various reasons for using CICS journaling, the following are particularly relevant to FEPI processing:

- Creating audit trails
- Monitoring performance
- Controlling message security.

Table 6 shows the types of FEPI data that can be journaled.

Table 6. FEPI journaled data

FEPI command	Data flow	Type
SEND	Outbound	Data stream Formatted, screen image Formatted, key stroke
RECEIVE	Inbound	Data stream Formatted, screen image
CONVERSE	Outbound	Data stream Formatted, screen image Formatted, key stroke
CONVERSE	Inbound	Data stream Formatted, screen image
EXTRACT FIELD	Inbound	Extract field data

The records journaled by FEPI are identified in the usual way by module and function identifiers. These are listed in Table 7.

Table 7. FEPI journal record identifiers

Identifier-type	Name	Value	Type of data
Module identifier	MODIDFEP	X'5D'	Identifies FEPI records in the journal
Function identifiers	FIDFEPIN FIDFEPOU	X'F0' X'F1'	Identifies FEPI inbound data Identifies FEPI outbound data

In order to identify the conversation for which the data was journaled, FEPI provides a prefix area in the journal record.

Printing FEPI journal records

Each FEPI journal record contains a prefix area which contains FEPI-related information.

You can select FEPI journal records in the following ways, using a batch job like the CICS-supplied utility program DFHJUP:

- Print or copy selected journal records from CICS log streams or SMF data sets, as specified by control statement input
- Select and print journal records on the basis of their sequential position in the log stream or SMF data set
- Select and print journal records as determined by data contained within the records themselves, such as the contents of time, date, or identification fields
- Allow EXIT routines to process any selected journal records
- Print or copy an entire log stream or SMF data set.

The FEPI prefix area lies within the API user header, as shown in Figure 8.

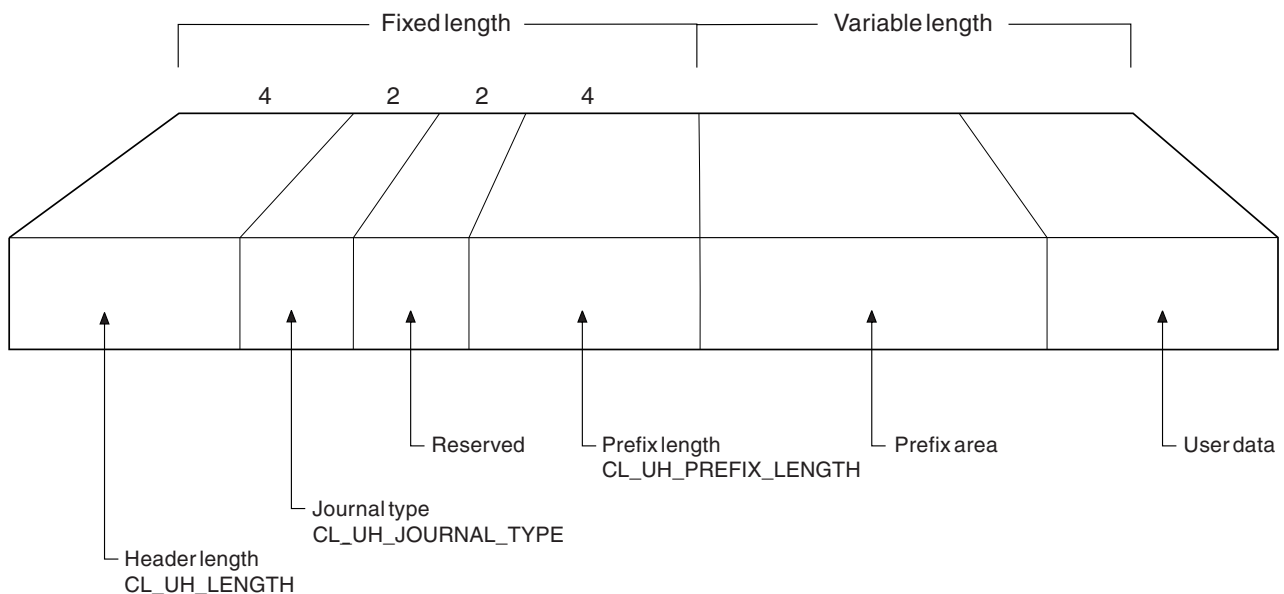


Figure 8. Format of the API user header, showing the position of the prefix area

- CL_UH_LENGTH**
4-byte length of header
- CL_UH_JOURNAL_TYPE**
2-byte journal type
- Reserved**
2-byte reserved field
- CL_UH_PREFIX_LENGTH**
4-byte length of prefix
- Prefix area**
The variable length prefix
- User data**
Variable length user data

The exact format of this FEPI prefix area is shown in Figure 9.

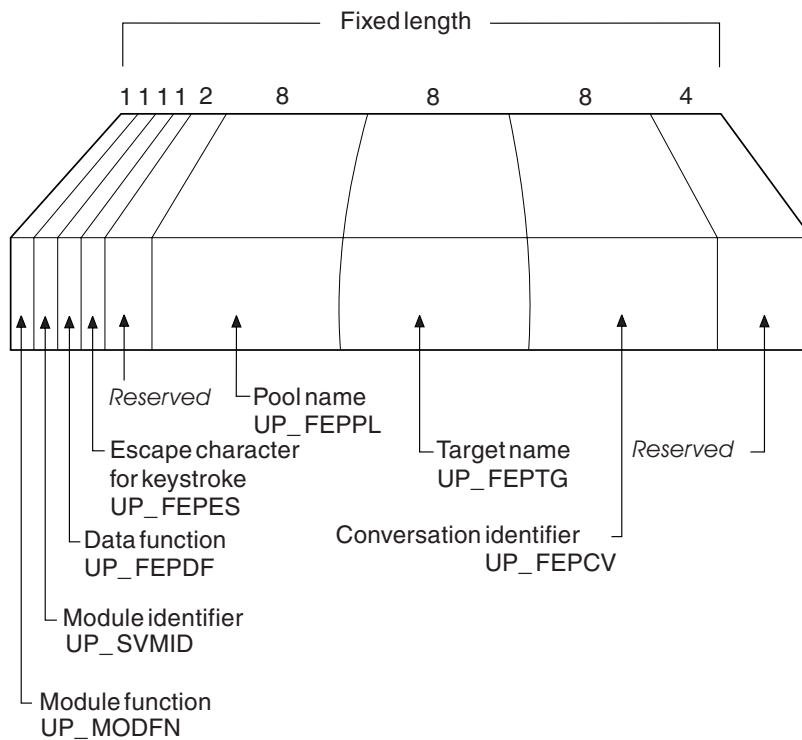


Figure 9. Format of the FEPI prefix area

UP_MODFN
1-byte module function.

UP_SVMID
1-byte module identifier.

UP_FEPDF
1-byte data function.

Field UP_FEPDF can take any of the following values:

Table 8. Values of UP_FEPDF

Field name	Value	Meaning
UP_FEPDD	1	Data stream
UP_FEPDS	2	Formatted, screen image
UP_FEPDK	3	Formatted, keystroke
UP_FEPDE	4	Extract field data

UP_FEPES
1-byte escape character for keystroke.

Reserved
2-byte reserved field.

UP_FEPPL
8-byte pool name.

UP_FEPTG
8-byte target name.

UP_FEPCV

8-byte conversation identifier.

Reserved

4-byte reserved field.

See the *CICS Operations and Utilities Guide* for examples of ways in which you can use the CICS-supplied utility program, DFHJUP, to select FEPI records for printing.

See the *CICS Customization Guide* for details on the structure of journal records.

Related tasks

Reading log streams using batch jobs (for example, DFHJUP)

Chapter 11. FEPI system programming reference

This section describes the FEPI system programming commands that you use for FEPI configuration and operation.

(Application programming commands such as ALLOCATE, CONVERSE, and EXTRACT are described in Part 4, “FEPI application programming reference,” on page 175.)

The section contains the following topics:

- “The FEPI SPI commands”
- “Transient data queue records” on page 123.

The FEPI SPI commands

The FEPI system programming commands are an addition to the system programming group of EXEC CICS commands and have the same features and properties.

The notation used to describe the syntax of FEPI commands is the same as that used to describe all system programming commands in CICS. To use these commands, you must be familiar with:

- The format of EXEC CICS commands
- Input and output values, and CVDAs
- The use of the RESP, RESP2, and NOHANDLE options
- Security checking
- The use of INQUIRE and SET commands
- Browsing.

Unlike other CICS system programming commands, the FEPI system programming commands do not need the ‘SP’ translator option. However, you do need to specify the ‘FEPI’ translator option.

The FEPI INQUIRE and SET commands work in the same way as other CICS INQUIRE and SET commands. They allow you to look at named FEPI resource definitions, browse sets of related definitions, and modify some of the defined values.

FEPI commands can be issued in either 24-bit or 31-bit addressing mode, by programs that reside either above or below the 16MB line. No information is passed through the EXEC interface block (EIB) except that, as for all CICS commands, the EIBRESP, EIBRESP2, EIBFN, and EIBRCODE fields are set.

Arguments and data types

The text used to identify arguments indicates the type of data represented by the argument and whether it is a value used by the command, or an area in which the command returns data. For example:

- POOL(8-character data-value) indicates that the argument is, or identifies, a string of eight characters, and that the string is passed to the command as an input value.
- ACQNUM(fullword binary data-area) indicates that the argument is a user-defined fullword data area in which the command can return a binary number as an output value.

Exceptionally, arguments that are lists have to be data areas, even though they are input values.

Command format

The general format of a command is:

```
EXEC CICS FEPI command option(argument)...
```

where:

command

Is the command name (for example, ADD)

option Is an option name (for example, POOL)

argument

Is the source or destination for data, as required for the specified option, that is passed to or returned from the command.

The way that you terminate the command is determined by the programming language that you use—COBOL, for example, requires an END-EXEC statement.

Errors and exception conditions

All FEPI commands support the RESP and RESP2 options to signal successful completion or an exception condition. Alternatively, you can use HANDLE CONDITION to trap errors.

Most FEPI command errors give the 'INVREQ' exception condition. The particular error in each case is uniquely identified by the RESP2 value.

Both RESP and RESP2 take, as an argument, the name of a user-defined fullword binary data area. Possible values of the RESP2 option are given in the description of each of the commands and a full list is given in "FEPI RESP2 values" on page 260. The following copy books provide declarations for the RESP2 values:

- DFHSZAPA for assembler language
- DFHSZAPO for COBOL
- DFHSZAPP for PL/I
- DFHSZAPC for C.

The following conditions and RESP2 values can occur for any system programming command:

Condition	RESP2	Meaning
INVREQ	10	Command bypassed by user exit.
INVREQ	11	FEPI not installed, or not active.
INVREQ	12	CICS shutting down, command not allowed.
INVREQ	13	FEPI unavailable.
INVREQ	14	FEPI busy or cannot get storage.

Condition	RESP2	Meaning
INVREQ	15	Unknown command.
INVREQ	16	Internal error.
INVREQ	17	FEPI cannot get storage for user exit.
INVREQ	18	Command failed through operator or system action.
NOTAUTH	100	Not authorized for this command.

If there is an error, the command does nothing, and the output arguments are not changed.

By their nature, some commands (for example, FEPI SET NODE INSERVICE) initiate a function and return before the function has completed. Errors in the execution of the function cannot be reported as an exception condition on the command. Such errors are reported by writing a record to a transient data (TD) queue and a message to the message log CSZL. See “Transient data queue records” on page 123 for details.

List processing

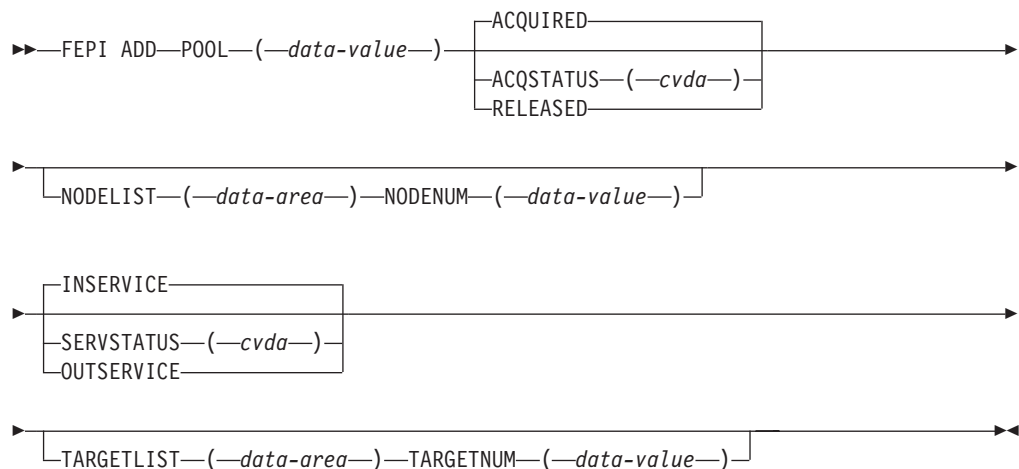
Commands that operate on a list of resources can fail for some of the resources in the list, but succeed for others. If this happens, a ‘list error’ is returned on the command. A record is written to a TD queue for each of the resources for which the command failed.

Even if the command fails for all of the resources in the list, it may still be partially successful if other parameters are valid. For example, a **FEPI INSTALL POOL** command installs a valid pool even if the array of node names specified on the **NODELIST** parameter does not exist.

FEPI ADD POOL

Add targets or nodes to an existing pool.

FEPI ADD POOL



Description

FEPI ADD POOL adds targets or nodes, or both, to an existing pool, thereby creating new connections in the pool. The targets or nodes must not be in the pool already. You can specify initial service and acquire states for these new connections. The command completes when the resources have been added to the pool but without waiting for the requested states to be achieved.

Options

ACQSTATUS(cvda)

specifies the initial acquire state of the connections being created. All the new connections have the same state. The relevant CVDA values are:

ACQUIRED

The connections are to have sessions established (that is, be 'bound').

RELEASED

The connections are not to have sessions established (that is, be left 'unbound').

NODELIST(data-area)

specifies a contiguous array of 8-character node names to be added to the pool. They must already be defined by FEPI INSTALL NODELIST, but can have any service state.

NODENUM(fullword binary data-value)

specifies the number of names in the NODELIST, in the range 0–256.

POOL(8-character data-value)

specifies the name of the pool to which the targets or nodes, or both, are being added.

SERVSTATUS(cvda)

specifies the initial service state of the connections being created. All the new connections have the same state. The relevant CVDA values are:

INSERVICE

The connections are to be in service, and so can be used in a conversation.

OUTSERVICE

The connections are to be out of service and cannot be used for any conversation.

TARGETLIST(data-area)

specifies a contiguous array of 8-character target names to be added to the pool. They must already be defined by FEPI INSTALL TARGETLIST, but can be in any service state.

TARGETNUM(fullword binary data-value)

specifies the number of names in TARGETLIST, in the range 0–256.

Conditions

INVREQ

RESP2 values:

110 SERVSTATUS value not valid.

111 ACQSTATUS value not valid.

115 POOL name unknown.

- 116 TARGET name unknown.
- 117 NODE name unknown.
- 119 The command failed for one or more items in the list.
- 130 TARGETNUM value is out of range.
- 131 NODENUM value is out of range.
- 173 NODE name already exists in the specified pool.
- 174 TARGET name already exists in the specified pool.
- 175 Connection already exists.

FEPI DELETE POOL

Remove targets or nodes from a specified FEPI pool.

FEPI DELETE POOL

```

▶▶ FEPI DELETE POOL (—data-value—)
▶
└─ NODELIST (—data-area—) NODENUM (—data-value—)
▶
└─ TARGETLIST (—data-area—) TARGETNUM (—data-value—)
▶▶

```

Description

FEPI DELETE POOL removes targets or nodes, or both, from a specified pool, thereby removing connections from the pool. The targets or nodes must be in the pool already. The command completes immediately, without waiting for the necessary deletions to be achieved. When the connections are deleted, they are no longer defined to FEPI.

Options

NODELIST(data-area)

specifies a contiguous array of 8-character node names that are to be deleted from the pool.

NODENUM(fullword binary data-value)

specifies the number of names in the NODELIST, in the range 0–256.

POOL(8-character data-value)

specifies the name of the pool from which targets or nodes are to be removed.

TARGETLIST(data-area)

specifies a contiguous array of 8-character target names that are to be deleted from the pool.

TARGETNUM(fullword binary data-value)

specifies the number of names in TARGETLIST, in the range 0–256.

Conditions

INVREQ

RESP2 values:

- 115 POOL name unknown.

- 116 TARGET name unknown.
- 117 NODE name unknown.
- 119 The command failed for one or more items in the list.
- 130 TARGETNUM value out of range.
- 131 NODENUM value out of range.

FEPI DISCARD NODELIST

Remove nodes from FEPI.

FEPI DISCARD NODELIST

►►—FEPI DISCARD—NODELIST—(*—data-area—*)—NODENUM—(*—data-value—*)—►►

Description

FEPI DISCARD NODELIST removes nodes completely from FEPI. The state of each node to be discarded is set to OUTSERVICE RELEASED (see “FEPI SET NODE” on page 120). When this state is achieved, the node is deleted from any pool that it is in. The nodes are then discarded so that they are no longer defined to FEPI. The command completes immediately without waiting for the necessary service and acquire states to be achieved.

Options

NODELIST(*data-area*)

specifies a contiguous array of 8-character node names that are to be discarded.

NODENUM(*fullword binary data-value*)

specifies the number of names in NODELIST, in the range 1–256.

Conditions

INVREQ

RESP2 values:

- 117 NODE name unknown.
- 119 The command failed for one or more items in the list.
- 131 NODENUM value out of range.

FEPI DISCARD POOL

Remove a pool of connections.

FEPI DISCARD POOL

►►—FEPI DISCARD—POOL—(*—data-value—*)—►►

Description

FEPI DISCARD POOL removes a pool of connections completely from FEPI. The state of the connections in the pool is set to OUTSERVICE RELEASED (see “FEPI SET CONNECTION” on page 118), and the state of the pool is set to OUTSERVICE

(see “FEPI SET POOL” on page 121). When these states have been achieved, the pool and its connections are discarded, so that they are no longer defined to FEPI. The command completes immediately, without waiting for the necessary service and acquire states to be achieved.

Options

POOL(8-character data-value)

specifies the name of the pool to be discarded.

Conditions

INVREQ

RESP2 values:

115 POOL name unknown.

FEPI DISCARD PROPERTYSET

Remove a set of properties.

FEPI DISCARD PROPERTYSET

►►—FEPI DISCARD—PROPERTYSET—(*—data-value—*)—————►►

Description

FEPI DISCARD PROPERTYSET removes a set of properties. The properties are discarded immediately so that they are no longer defined to FEPI, but any pool that was installed using the properties is not affected.

Options

PROPERTYSET(8-character data-value)

Specifies the name of the set of properties to be discarded.

Conditions

INVREQ

RESP2 values:

171 PROPERTYSET name unknown.

FEPI DISCARD TARGETLIST

Remove targets from FEPI.

FEPI DISCARD TARGETLIST

►►—FEPI DISCARD—TARGETLIST—(*—data-area—*)—TARGETNUM—(*—data-value—*)—————►►

Description

FEPI DISCARD TARGETLIST removes targets completely from FEPI. The state of the targets to be discarded is set to OUTSERVICE (see “FEPI SET TARGET” on page 122). When this state has been achieved, the targets are deleted from any pool they are in, and are then discarded, so that they are no longer defined to FEPI. The

command completes immediately, without waiting for the necessary service and acquire states to be achieved.

Options

TARGETLIST(data-area)

specifies a contiguous array of 8-character target names that are to be discarded.

TARGETNUM(fullword binary data-value)

specifies the number of names in TARGETLIST, in the range 1–256.

Conditions

INVREQ

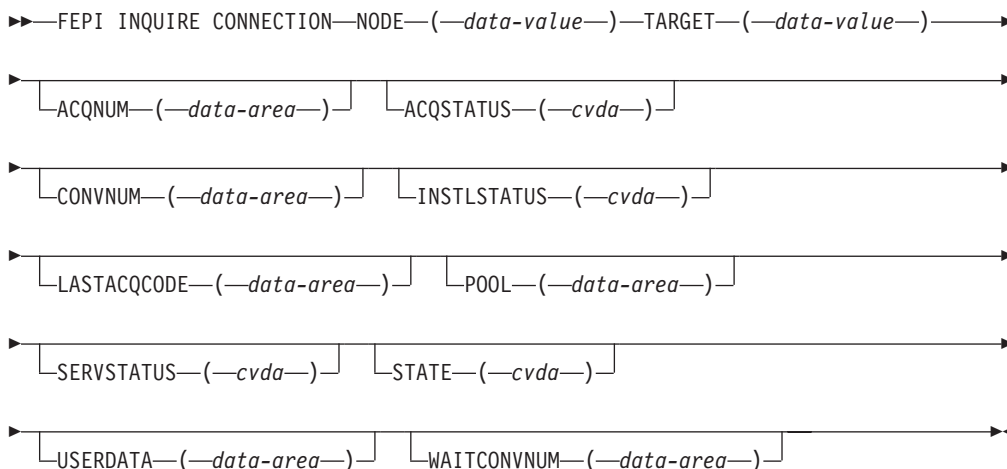
RESP2 values:

- 116 TARGET name unknown.
- 119 The command failed for one or more items in the list.
- 130 TARGETNUM value out of range.

FEPI INQUIRE CONNECTION

Inquire on a FEPI connection.

FEPI INQUIRE CONNECTION



Description

FEPI INQUIRE CONNECTION returns information about a FEPI connection. A connection is identified by specifying its target and node.

The following commands allow you to browse all FEPI connections.

```

FEPI INQUIRE CONNECTION START
FEPI INQUIRE CONNECTION NEXTNODE|NEXTTARGET
NODE(8-character data-area)
TARGET(8-character data-area)
[The options are as for FEPI INQUIRE CONNECTION]
FEPI INQUIRE CONNECTION END
  
```

Conditions: INVREQ, NOTAUTH

The next connection for which information is returned depends on whether NEXTNODE or NEXTTARGET is specified. If NEXTNODE is specified, the information returned is for:

- The next node connected to the current target
- If there are no more nodes connected to the current target, then the first node connected to the next target.

If NEXTTARGET is specified, the information returned is for:

- The next target connected to the current node
- If there are no more targets connected to the current node, then the first target connected to the next node.

Options

ACQNUM(fullword binary data-area)

returns the number of times that the connection has been acquired.

ACQSTATUS(cvda)

returns the acquire state; that is, whether a session on the connection is bound or not. The relevant CVDA values are:

ACQUIRED

The session is bound.

ACQUIRING

A state of ACQUIRED has been requested but binding a session has not yet been completed.

RELEASED

No session is bound.

RELEASING

A state of RELEASED has been requested but unbinding the session has not yet been completed.

If ACQUIRING or RELEASING persist, the operator might need to intervene using VTAM commands to recover the connection.

CONVNUM(fullword binary data-area)

returns the number of conversations that have used the connection.

INSTLSTATUS(cvda)

returns the install state of the connection. The relevant CVDA values are:

INSTALLED

The connection is in a pool defined by INSTALL and is available for use.

NOTINSTALLED

The connection is in a pool, or involves a node or target that is being discarded but is still in use.

LASTACQCODE(fullword binary data-area)

returns the result of the last acquire request for the connection; that is, the sense code from the last VTAM REQSESS, zero indicating success.

Note: CLSDST(PASS)—X'32020000'—can be returned in this field. This is the unbind flow received by CICS during CLSDST(PASS) processing.

For details of VTAM sense codes, see the *VTAM Messages and Codes* manual, or *SNA Formats* manual.

NODE(8-character data-value/8-character data-area)

is the node identifying the connection.

POOL(8-character data-area)

returns the name of the pool that defines the connection.

SERVSTATUS(cvda)

returns the service state of the connection. The relevant CVDA values are:

INSERVICE

The connection is in service and can be used in a conversation.

OUTSERVICE

The connection is out of service and cannot be used for any new conversation, but a conversation using the connection is unaffected. The service state is GOINGOUT until any such conversation ends.

GOINGOUT

A state of OUTSERVICE has been requested but the connection is still being used by some conversation.

STATE(cvda)

returns the state of the conversation using the connection. The relevant CVDA values are:

NOCONV

No conversation is active on the connection.

PENDSTSN

An STSN-handling task has been scheduled.

STSN An STSN-handling task owns the conversation.

PENDBEGIN

A begin-session handling task has been scheduled.

BEGINSESSION

A begin-session handling task owns the conversation.

APPLICATION

A normal application task owns the conversation.

PENDDATA

FEPI is waiting for inbound data, following a FEPI START command.

PENDSTART

Inbound data having arrived, a task specified by FEPI START has been scheduled.

PENDFREE

An end-session handling task has been scheduled, following a FEPI FREE command.

FREE An end-session handling task owns the conversation, following a FEPI FREE command.

PENDRELEASE

An end-session handling task has been scheduled, following an unbind request.

RELEASE

An end-session handling task owns the conversation, following an unbind request.

PENDUNSOL

An unsolicited-data handling task has been scheduled.

UNSOLDATA

An unsolicited-data handling task owns the conversation.

PENDPASS

The conversation is unowned, following a FEPI FREE PASS command.

The 'pending' states indicate that the conversation is unowned, pending the event or task indicated; the state ceases to be pending when a task issues a FEPI ALLOCATE PASSCONVID command. If a 'pending' state persists, it is likely that the application has failed in some way; you should consider resetting the connection by issuing FEPI SET CONNECTION RELEASED.

TARGET(8-character data-value/8-character data-area)

is the target identifying the connection.

USERDATA(64-character data-area)

returns the user data for the connection. If no user data has been set, nulls are returned.

WAITCONVNUM(fullword binary data-area)

returns the number of conversations that are waiting to start using the connection. Note that, if a conversation could use any one of several connections, it is counted as waiting on each one.

Conditions

ILLOGIC

RESP2 values:

- 1 For START: browse of this resource type is already in progress. For NEXT or INQUIRE: END was not issued.

END

RESP2 values:

- 2 For NEXT: all resource definitions have been retrieved.

INVREQ

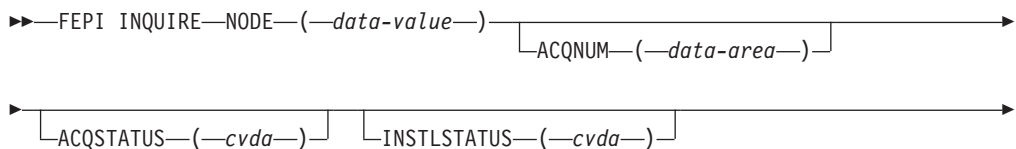
RESP2 values:

- 116 TARGET name unknown.
- 117 NODE name unknown.
- 118 Connection unknown (TARGET and NODE names known, but not in a common pool).

FEPI INQUIRE NODE

Inquire on a FEPI node.

FEPI INQUIRE NODE





Description

FEPI INQUIRE NODE returns information about a FEPI node.

The following commands allow you to browse all FEPI NODE definitions.

```
FEPI INQUIRE NODE START
FEPI INQUIRE NODE(8-character data-area) NEXT
[The options are as for FEPI INQUIRE NODE]
FEPI INQUIRE NODE END
```

Options

ACQNUM(fullword binary data-area)

returns the number of times that the node has been acquired.

ACQSTATUS(cvda)

returns the acquire state—that is, whether the VTAM ACB is opened or closed. The relevant CVDA values are:

ACQUIRED

The VTAM ACB for the node is open and 'set logon start' has completed.

ACQUIRING

A state of ACQUIRED has been requested but opening the VTAM ACB for the node and issuing 'set logon start' has not yet been completed.

RELEASED

Sessions on any connections involving the node have been unbound and the VTAM ACB has been closed.

RELEASING

A state of RELEASED has been requested but closing the VTAM ACB for the node has not yet been completed.

If ACQUIRING or RELEASING persist, the operator might need to intervene using VTAM commands to recover the node.

INSTLSTATUS(cvda)

returns the install state of the node. The relevant CVDA values are:

INSTALLED

The node has been defined by INSTALL and is available for use.

NOTINSTALLED

The node is being discarded, but is still in use.

LASTACQCODE(fullword binary data-area)

returns the result of the last acquire request for the node; that is, the return code from the last VTAM OPEN ACB, zero indicating success. For details of VTAM return codes, see the *VTAM Programming* manual.

NODE(8-character data-value/8-character data-area)

is the name of the node.

SERVSTATUS(cvda)

returns the service state of the node. The relevant CVDA values are:

INSERVICE

The node is in service and can be used in a conversation.

OUTSERVICE

The node is out of service and cannot be used for any conversation.

GOINGOUT

A state of OUTSERVICE has been requested but the node is still being used by a conversation.

USERDATA(64-character data-area)

returns the user data for the node. If no user data has been set, nulls are returned.

Conditions**ILLOGIC**

RESP2 value:

- 1 For START: browse of this resource type is already in progress. For NEXT or END: START was not issued.

END

RESP2 value:

- 2 For NEXT: all resource definitions have been retrieved.

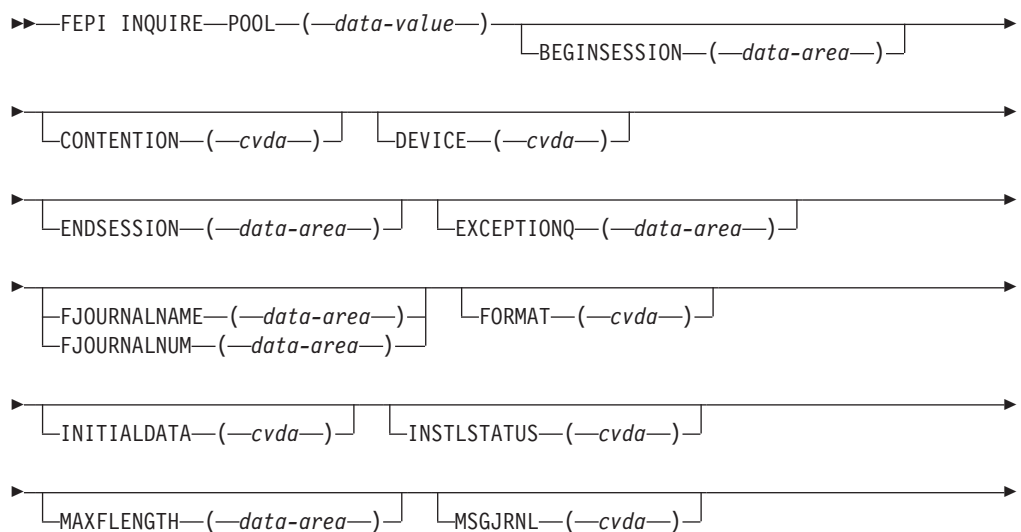
INVREQ

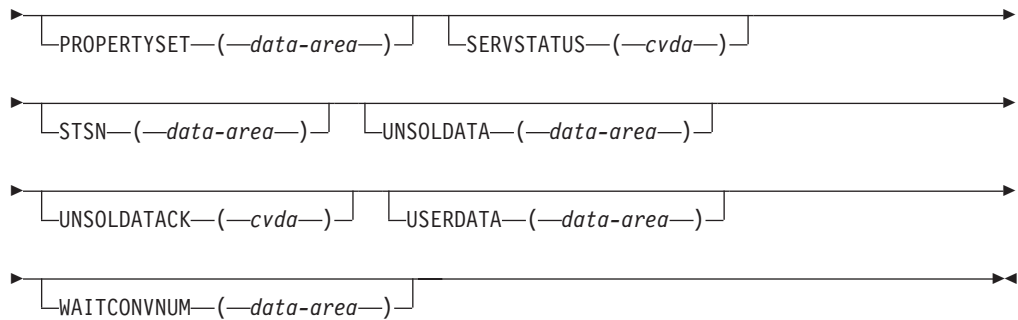
RESP2 value:

- 117 NODE name unknown.

FEPI INQUIRE POOL

Inquire on a FEPI pool.

FEPI INQUIRE POOL



Description

FEPI INQUIRE POOL returns information about a FEPI pool of connections.

The following commands allow you to browse all FEPI POOL definitions.

```
FEPI INQUIRE POOL START
FEPI INQUIRE POOL(8-character data-area) NEXT
[The options are as for FEPI INQUIRE POOL]
FEPI INQUIRE POOL END
```

Options

BEGINSESSION(4-character data-area)

returns the name of the transaction performing begin-session processing, or blanks if no transaction was specified.

CONTENTION(cvda)

returns a value that specifies what happens when a FEPI SEND command is issued and there is inbound data with 'begin bracket'. The relevant CVDA values are:

LOSE FEPI SEND command fails; a FEPI RECEIVE must be issued to get the inbound data.

WIN FEPI SEND command succeeds; inbound data is rejected with a negative response.

DEVICE(cvda)

returns a value that identifies the mode of conversation and the type of device. Defined values are:

T3278M2
SLU2 mode, 3278 Model 2

T3278M3
SLU2 mode, 3278 Model 3

T3278M4
SLU2 mode, 3278 Model 4

T3278M5
SLU2 mode, 3278 Model 5

T3279M2
SLU2 mode, 3279 Model 2B

T3279M3
SLU2 mode, 3279 Model 3B

T3279M4
SLU2 mode, 3279 Model 4B

T3279M5
SLU2 mode, 3279 Model 5B

TPS55M2
SLU2 mode, PS/55, 24 lines

TPS55M3
SLU2 mode, PS/55, 32 lines

TPS55M4
SLU2 mode, PS/55, 43 lines

LUP SLU P mode, all cases.

ENDESESSION(4-character data-area)
returns the name of the transaction performing end-session processing, or blanks if no transaction was specified.

EXCEPTIONQ(4-character data-area)
returns the name of the TD queue to which exceptional events are notified, or blanks if no queue was specified.

FJOURNALNAME(8-character data-area)
returns the 1- to 8-character name of the journal where data is to be logged.

FJOURNALNUM(fullword binary data-area)
returns the number of the journal where data is to be logged.

FORMAT(cvda)
returns a value that identifies the data format. The relevant CVDA values are:

FORMATTED
Formatted operation

DATASTREAM
Data stream operation

NOTAPPLIC
Option is not applicable for the specified pool.

INITIALDATA(cvda)
returns a value indicating whether initial inbound data is expected when a session is started. The relevant CVDA values are:

NOTINBOUND
No inbound data expected

INBOUND
Inbound data expected.

INSTLSTATUS(cvda)
returns the install state of the pool. The relevant CVDA values are:

INSTALLED
The pool has been defined by INSTALL and is available for use.

NOTINSTALLED
The pool is being discarded, but is still in use.

MAXLENGTH(fullword binary data-area)
returns the maximum length of data that can be returned on any FEPI

RECEIVE, CONVERSE, or EXTRACT FIELD command for a conversation, or that can be sent by any FEPI SEND or CONVERSE command for a conversation.

MSGJRNL(cvda)

returns a value indicating whether journaling is performed for inbound and outbound data. The relevant CVDA values are:

NOMSGJRNL

No journaling is to be performed.

INPUT

Inbound data is journaled.

OUTPUT

Outbound data is journaled.

INOUT

Inbound and outbound data are journaled.

POOL(8-character data-value/8-character data-area)

is the name of the pool.

PROPERTYSET(8-character data-area)

returns the name of the set of properties with which the pool was installed.

SERVSTATUS(cvda)

returns the service state of the pool. The relevant CVDA values are:

INSERVICE

The pool is in service and can be used in a conversation.

OUTSERVICE

The pool is out of service and cannot be used for any conversation.

GOINGOUT

A state of OUTSERVICE has been requested but the pool is still being used by some conversation.

STSN(4-character data-area)

returns the name of the transaction handling STSN data, or blanks if no transaction was specified.

UNSOLDATA(4-character data-area)

returns the name of the transaction handling unsolicited data (data received outside a conversation), or blanks if no transaction was specified.

UNSOLDATAACK(cvda)

if there is no unsolicited data processing, this indicates what acknowledgment FEPI gives to a BID. The relevant CVDA values are:

NEGATIVE

Negative response X'0813', BID not accepted

POSITIVE

Positive response, BID accepted and subsequent data is accepted and discarded

NOTAPPLIC

Option is not applicable for the specified pool.

USERDATA(64-character data-area)

returns the user data for the pool. If no user data has been set, nulls are returned.

WAITCONVNUM(fullword binary data-area)

returns the number of conversations that are waiting to start using a connection in the pool.

Conditions**ILLOGIC**

RESP2 value:

- 1 For START: browse of this resource type is already in progress. For NEXT or END: START was not issued.

END

RESP2 value:

- 2 For NEXT: all resource definitions have been retrieved.

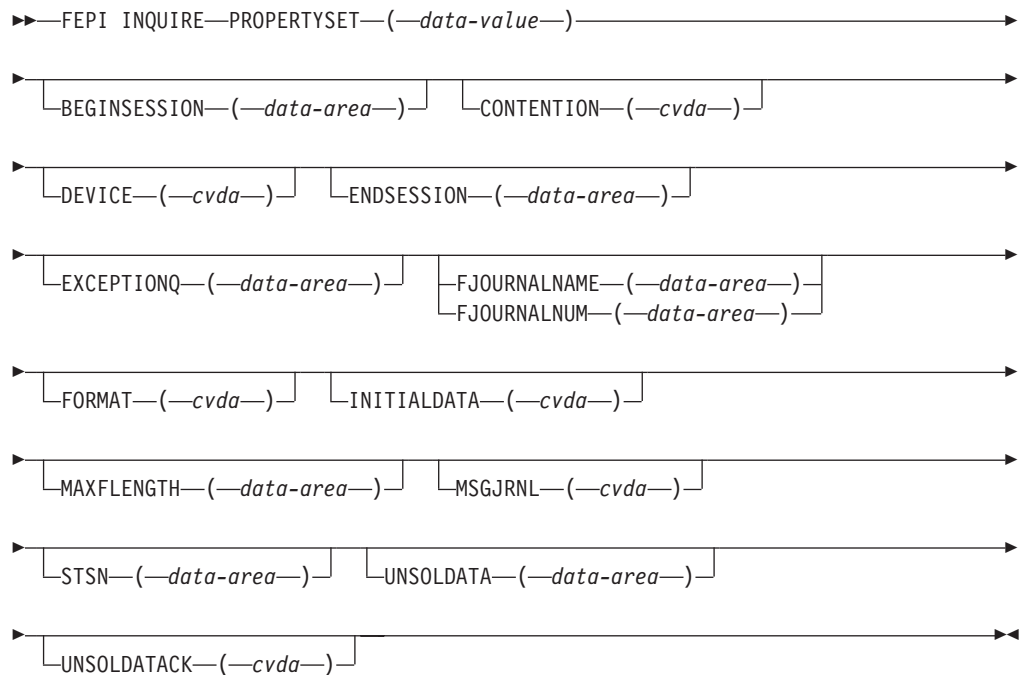
INVREQ

RESP2 value:

- 115 POOL name unknown.

FEPI INQUIRE PROPERTYSET

Inquire on a FEPI property set.

FEPI INQUIRE PROPERTYSET**Description**

FEPI INQUIRE PROPERTYSET returns information about a FEPI property set.

The following commands allow you to browse all FEPI PROPERTYSET definitions:

```
FEPI INQUIRE PROPERTYSET START
FEPI INQUIRE PROPERTYSET(8-character data-area) NEXT
[The options are as for FEPI INQUIRE PROPERTYSET]
FEPI INQUIRE PROPERTYSET END
```

Options

BEGINSESSION(4-character data-area)

returns the name of the transaction performing begin-session processing, or blanks if no transaction was specified.

CONTENTION(cvda)

returns a value that specifies what happens when a FEPI SEND command is issued and there is inbound data with 'begin bracket'. The relevant CVDA values are:

LOSE FEPI SEND command fails; a FEPI RECEIVE must be issued to get the inbound data.

WIN FEPI SEND command succeeds; inbound data is rejected with a negative response.

DEVICE(cvda)

returns a value that identifies the mode of conversation and the type of device. Defined values are:

T3278M2

SLU2 mode, 3278 Model 2

T3278M3

SLU2 mode, 3278 Model 3

T3278M4

SLU2 mode, 3278 Model 4

T3278M5

SLU2 mode, 3278 Model 5

T3279M2

SLU2 mode, 3279 Model 2B

T3279M3

SLU2 mode, 3279 Model 3B

T3279M4

SLU2 mode, 3279 Model 4B

T3279M5

SLU2 mode, 3279 Model 5B

TPS55M2

SLU2 mode, PS/55, 24 lines

TPS55M3

SLU2 mode, PS/55, 32 lines

TPS55M4

SLU2 mode, PS/55, 43 lines

LUP SLU P mode, all cases.

ENDSESSION(4-character data-area)

returns the name of the transaction performing end-session processing, or blanks if no transaction was specified.

EXCEPTIONQ(4-character data-area)

returns the name of the TD queue to which exceptional events are notified, or blanks if no queue was specified.

FJOURNALNAME(8-character data-area)

returns the 1- to 8-character name of the journal where data is to be logged.

FJOURNALNUM(fullword binary data-area)

returns the number of the journal where data is to be logged.

FORMAT(cvda)

returns a value that identifies the data format. The relevant CVDA values are:

FORMATTED

Formatted operation

DATASTREAM

Data stream operation

NOTAPPLIC

Option is not applicable for the specified pool.

INITIALDATA(cvda)

returns a value indicating whether initial inbound data is expected when a session is started. The relevant CVDA values are:

NOTINBOUND

No inbound data expected

INBOUND

Inbound data expected.

MAXLENGTH(fullword binary data-area)

returns the maximum length of data that can be returned on any FEPI RECEIVE, CONVERSE, or EXTRACT FIELD command for a conversation, or that can be sent by any FEPI SEND or CONVERSE command for a conversation.

MSGJRNL(cvda)

returns a value indicating whether journaling is performed for inbound and outbound data. The relevant CVDA values are:

NOMSGJRNL

No journaling is to be performed.

INPUT

Inbound data is journaled.

OUTPUT

Outbound data is journaled.

INOUT

Inbound and outbound data are journaled.

PROPERTYSET(8-character data-value/8-character data-area)

is the name of the set of properties.

STSN(4-character data-area)

returns the name of the transaction handling STSN data (SLU P mode only), or blanks if no transaction was specified.

UNSOLDATA(4-character data-area)

returns the name of the transaction handling unsolicited data (data received outside a conversation), or blanks if no transaction was specified.

UNSOLDATAACK(cvda)

indicates what acknowledgment FEPI gives to a BID, if there is no unsolicited-data processing. The relevant CVDA values are:

NEGATIVE

Negative response X'0813', BID not accepted

POSITIVE

Positive response, BID accepted and subsequent data is accepted and discarded

NOTAPPLIC

Option is not applicable for the specified pool.

Conditions**ILLOGIC**

RESP2 value:

- 1 For START: browse of this resource type is already in progress. For NEXT or END: START was not issued.

END

RESP2 value:

- 2 For NEXT: all resource definitions have been retrieved.

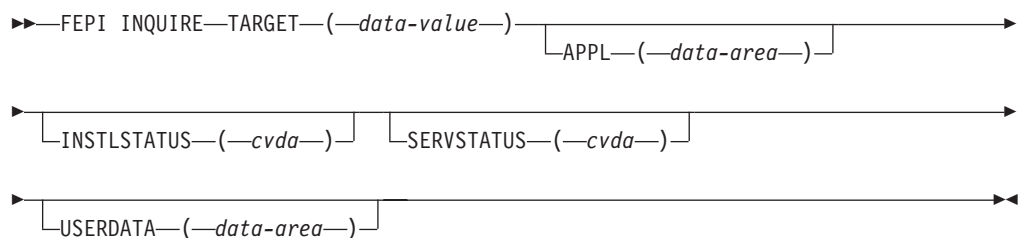
INVREQ

RESP2 value:

- 171 PROPERTYSET name unknown.

FEPI INQUIRE TARGET

Inquire on a FEPI target.

FEPI INQUIRE TARGET**Description**

FEPI INQUIRE TARGET returns information about a FEPI target.

The following commands allow you to browse all FEPI TARGET definitions.

```
FEPI INQUIRE TARGET START
FEPI INQUIRE TARGET(8-character data-area) NEXT
[The options are as for FEPI INQUIRE TARGET]
FEPI INQUIRE TARGET END
```

Options

APPL(8-character data-area)

returns the VTAM application name of the back-end system that the target system represents.

INSTLSTATUS(cvda)

returns the install state of the target. The relevant CVDA values are:

INSTALLED

The target has been defined by INSTALL and is available for use.

NOTINSTALLED

The target is being discarded but is still in use.

SERVSTATUS(cvda)

returns the service state of the target. The relevant CVDA values are:

INSERVICE

The target is in service and can be used in a conversation.

OUTSERVICE

The target is out of service and cannot be used for any conversation.

GOINGOUT

A state of OUTSERVICE has been requested but the target is still being used by some conversation.

TARGET(8-character data-value/8-character data-area)

is the name of the target.

USERDATA(64-character data-area)

returns the user data for the target. If no user data has been set, nulls are returned.

Conditions

ILLOGIC

RESP2 value:

- 1 For START: browse of this resource type is already in progress. For NEXT or END: START was not issued.

END

RESP2 value:

- 2 For NEXT: all resource definitions have been retrieved.

INVREQ

RESP2 value:

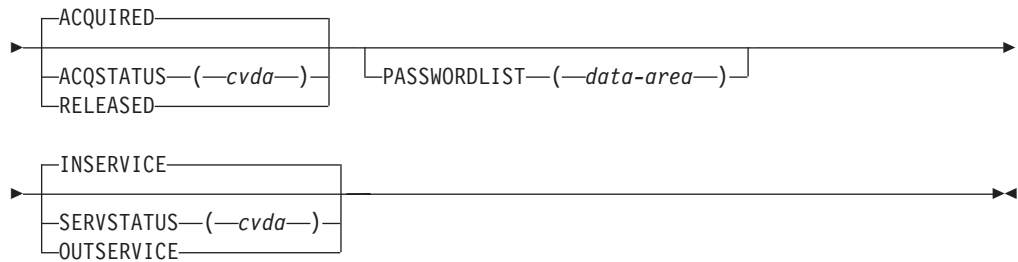
- 116 TARGET name unknown.

FEPI INSTALL NODELIST

Define new nodes to FEPI.

FEPI INSTALL NODELIST

►►—FEPI INSTALL—NODELIST—(—data-area—)—NODENUM—(—data-value—)—————►



Description

FEPI INSTALL NODELIST defines new nodes to FEPI. You may specify initial service and acquire states for these new nodes. A node cannot be used for a conversation until it has been acquired, put in service, and added to a pool so that it is connected to a target. The command completes when the nodes have been defined without waiting for the requested states to be achieved.

Options

ACQSTATUS(*cvda*)

specifies the initial acquire state of the nodes being defined. All nodes in the list have the same state. The relevant CVDA values are:

ACQUIRED

The VTAM ACB for the node is to be opened and 'set logon start' is to be done.

RELEASED

The VTAM ACB for the node is not to be opened.

NODELIST(*data-area*)

specifies a contiguous array of 8-character node names (that is, VTAM application minor node names in the front-end) to be defined. Names must not contain null characters (X'00'), leading blanks, or embedded blanks.

NODENUM(*fullword binary data-value*)

specifies the number of names in NODELIST, in the range 1-256.

PASSWORDLIST(*data-value*)

specifies a contiguous array of 8-character passwords. They correspond one-to-one with the node names in NODELIST. The passwords are those that VTAM requires to access the application minor nodes. They are not required if passwords are not used. You can use a value of 8 null characters (X'00') to indicate 'no password'.

SERVSTATUS(*cvda*)

specifies the initial service state of the nodes being defined. All nodes in the list have the same state. The relevant CVDA values are:

INSERVICE

The nodes are in service and can be used in a conversation.

OUTSERVICE

The nodes are out of service and cannot be used for any conversation.

Conditions

INVREQ

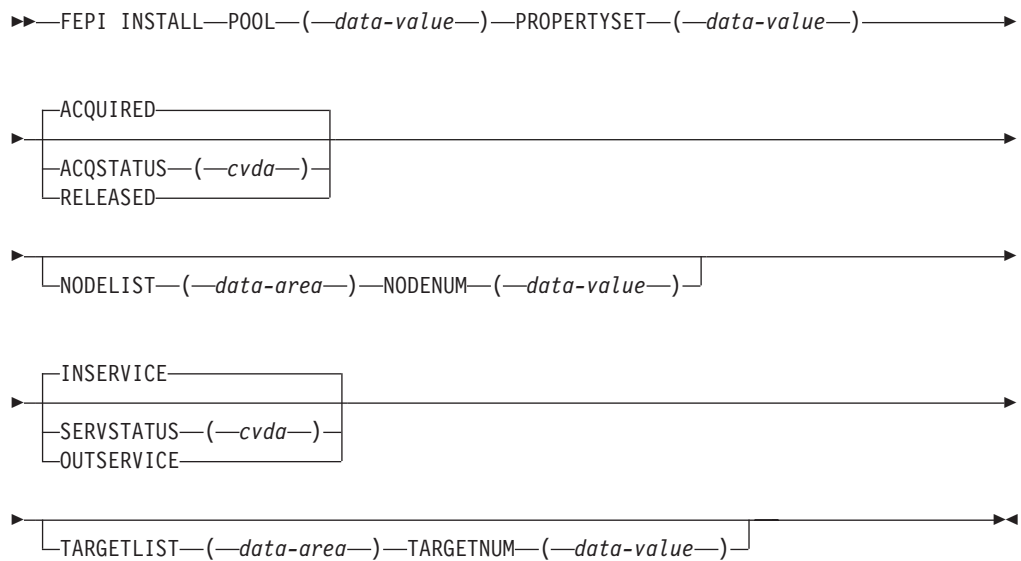
RESP2 values:

- 11 FEPI not installed or not active.
- 110 SERVSTATUS value not valid.
- 111 ACQSTATUS value not valid.
- 119 The command failed for one or more items in the list.
- 131 NODENUM value out of range.
- 163 NODE name not valid.
- 173 NODE name already exists.
- 176 The VTAM OPEN ACB failed.

FEPI INSTALL POOL

Define a new pool of connections.

FEPI INSTALL POOL



Description

FEPI INSTALL POOL defines a new pool of connections to FEPI. Any targets and nodes specified in the command are added to it, thereby creating new connections in the pool. You may specify an initial service state for the pool, and initial service and acquire states for any new connections. A pool cannot be used for a conversation until it has been put in service. The command completes when the pool has been created and any resources added; it does not wait for the requested states to be achieved.

Options

ACQSTATUS (cvda)

specifies the initial acquire state of the connections being created. All the new connections have the same state. The relevant CVDA values are:

ACQUIRED

The connections are to have sessions established (that is, 'bound').

RELEASED

The connections are not to have sessions established (that is, left 'unbound').

NODELIST(data-area)

specifies a contiguous array of 8-character node names. They must already be defined by FEPI INSTALL NODELIST.

NODENUM(fullword binary data-value)

specifies the number of names in NODELIST, in the range 0–256.

POOL(8-character data-value)

specifies the name of the pool to be defined. The name must not contain null characters (X'00'), leading blanks, or embedded blanks.

PROPERTYSET(8-character data-value)

specifies the name of the set of properties for the pool, which must have been installed already.

SERVSTATUS(cvda)

specifies the initial service state of the pool being defined and of the connections being created. All the new connections have the same state. The relevant CVDA values are:

INSERVICE

The pool and any connections are in service and can be used in a conversation.

OUTSERVICE

The pool and any connections are out of service and cannot be used for any conversation.

TARGETLIST(data-area)

specifies a contiguous array of 8-character target names. They must already be defined by FEPI INSTALL TARGETLIST.

TARGETNUM(fullword binary data-value)

specifies the number of names in TARGETLIST, in the range 0–256.

Conditions

INVREQ

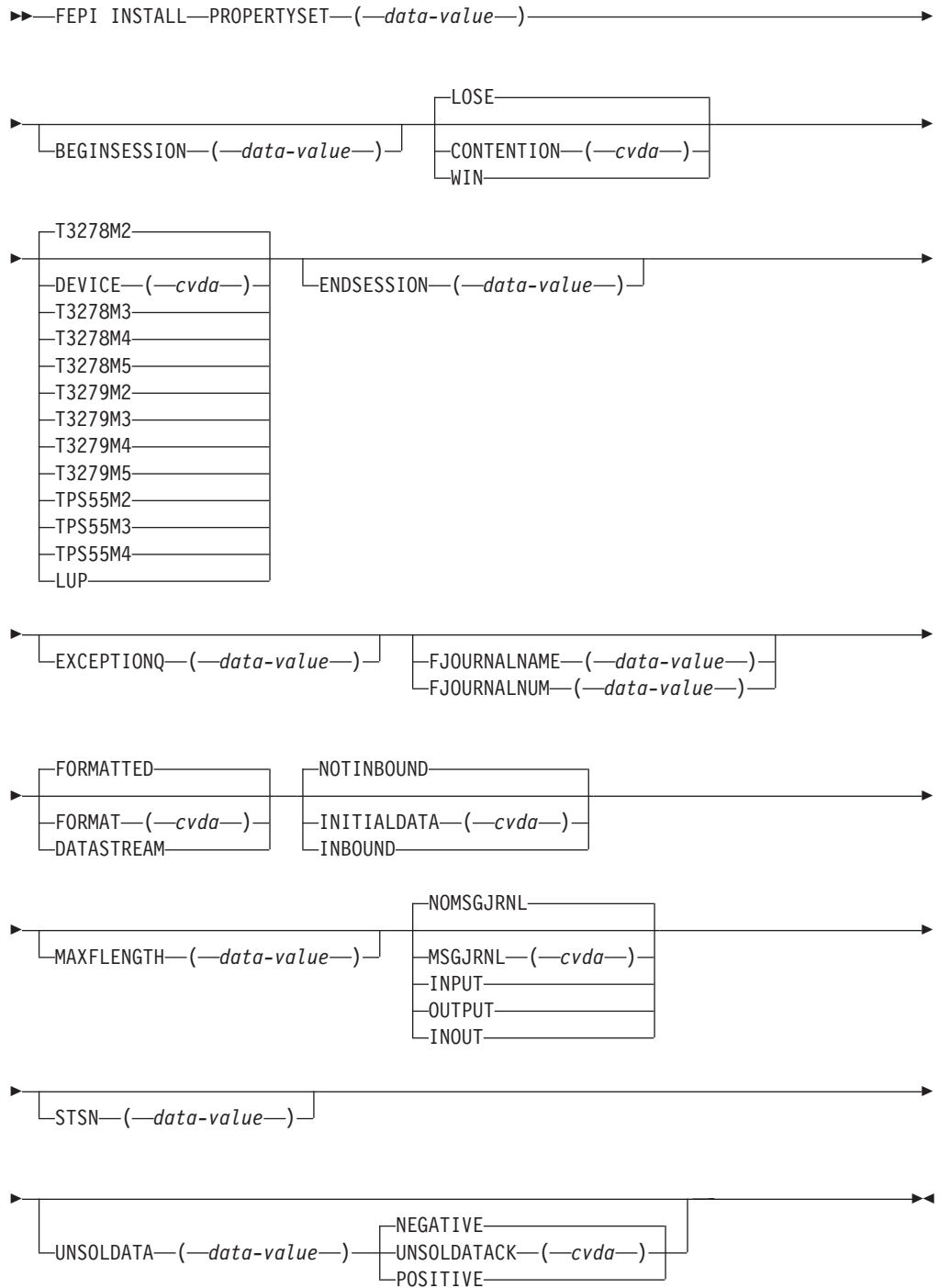
RESP2 values:

- 11 FEPI not installed or not active.
- 110 SERVSTATUS value not valid.
- 111 ACQSTATUS value not valid.
- 116 TARGET name unknown.
- 117 NODE name unknown.
- 119 The command failed for one or more items in the list.
- 130 TARGETNUM value out of range.
- 131 NODENUM value out of range.
- 162 POOL name not valid.
- 171 PROPERTYSET name unknown.
- 172 POOL name already exists.
- 175 The connection already exists.

FEPI INSTALL PROPERTYSET

Define a new set of properties.

FEPI INSTALL PROPERTYSET



Description

FEPI INSTALL PROPERTYSET defines a new set of properties to FEPI, which can be applied to any subsequently defined pool.

Options

Note: Specifying a blank value for BEGINSSESSION, ENDSSESSION, EXCEPTIONQ, STSN, or UNSOLDATA has the same effect as omitting the option.

BEGINSSESSION(4-character data-value)

specifies the name of the transaction to perform begin-session processing, immediately after a session has been established ('bound'). If omitted, there is to be no user-supplied begin-session processing.

CONTENTION(cvda)

specifies what happens when a FEPI SEND command is issued and there is inbound data with begin-bracket. The relevant CVDA values are:

LOSE The FEPI SEND command fails; a FEPI RECEIVE must be issued to get the inbound data.

WIN The FEPI SEND command succeeds; inbound data is rejected with a negative response.

DEVICE(cvda)

specifies the LU mode and device type that is to be simulated. The relevant CVDA values are:

T3278M2

SLU2 mode, 3278 Model 2

T3278M3

SLU2 mode, 3278 Model 3

T3278M4

SLU2 mode, 3278 Model 4

T3278M5

SLU2 mode, 3278 Model 5

T3279M2

SLU2 mode, 3279 Model 2B

T3279M3

SLU2 mode, 3279 Model 3B

T3279M4

SLU2 mode, 3279 Model 4B

T3279M5

SLU2 mode, 3279 Model 5B

TPS55M2

SLU2 mode, PS/55, 24 lines

TPS55M3

SLU2 mode, PS/55, 32 lines

TPS55M4

SLU2 mode, PS/55, 43 lines

LUP SLU P mode, all cases.

ENDESESSION(4-character data-value)

specifies the name of the transaction to perform end-session processing, when a conversation is ended (by a FEPI FREE command) or when a session is to be ended ('unbound'). If omitted, there is to be no user-supplied end-session processing.

EXCEPTIONQ(4-character data-value)

specifies the name of the TD queue to which pool-specific exceptional events are to be notified. If EXCEPTIONQ is omitted, there is to be no user-supplied exceptional event processing.

FJOURNALNAME(8-character data-value)

specifies the 1- to 8-character name of the journal where data is to be logged. You are not permitted to specify DFHLOG or DFHSHUNT, the primary and secondary system logs. If the value is zero or omitted, no journaling is done.

FJOURNALNUM(fullword binary data-value)

specifies the number of the journal where data is to be logged, in the range 1 through 99. Specifying a value here implies the journal name 'DFHJ*nm*', where *nm* is the journal number. If the value is zero or omitted, no journaling is done.

FORMAT(cvda)

specifies, for SLU2 mode, the data mode to be used. The relevant CVDA values are:

FORMATTED

Formatted operation. Character attributes are not supported on outbound data and ignored on inbound data.

DATASTREAM

Data stream operation.

This option is not valid for SLU P operation.

INITIALDATA(cvda)

specifies whether initial inbound data is expected when a session is started. The relevant CVDA values are:

NOTINBOUND

No inbound data is expected.

INBOUND

Inbound data is expected.

If the target is a back-end IMS system, you should specify INBOUND. See "Begin-session handler" on page 158.

MAXLENGTH(fullword binary data-value)

specifies the maximum length of data that can be returned on any FEPI RECEIVE, CONVERSE, or EXTRACT FIELD command for a conversation, or that can be sent by any FEPI SEND or CONVERSE command for a conversation. This value helps FEPI use storage more efficiently, so should be set no larger than is necessary. It must be in the range 128–1 048 576. If MAXLENGTH is not specified, 4096 is used.

MSGJRNL(cvda)

specifies the required journaling of data to and from the back-end system. The relevant CVDA values are:

NOMSGJRNL

No journaling

INPUT

Journal inbound data

OUTPUT

Journal outbound data

INOOUT

Journal inbound and outbound data.

PROPERTYSET(8-character data-value)

specifies the name of the set of properties to be defined. The name must not contain null characters (X'00'), leading blanks, or embedded blanks.

STSN(4-character data-value)

specifies the name of the transaction to be started to handle 'set and test sequence number' (STSN), for SLU P mode only. If omitted, there is to be no user-supplied STSN-handling; FEPI handles STSN automatically.

UNSOLDATA(4-character data-value)

specifies the name of the transaction to handle unsolicited data (data received outside a conversation). If omitted, there is to be no user-supplied unsolicited-data processing; FEPI treats unsolicited data as specified by UNSOLDATAACK.

UNSOLDATAACK(cvda)

if there is to be no unsolicited-data processing, this specifies what acknowledgment FEPI is to give to a BID. The relevant CVDA values are:

NEGATIVE

Negative response X'0813', BID not accepted

POSITIVE

Positive response, BID accepted and subsequent data is accepted and discarded.

Conditions**INVREQ**

RESP2 values:

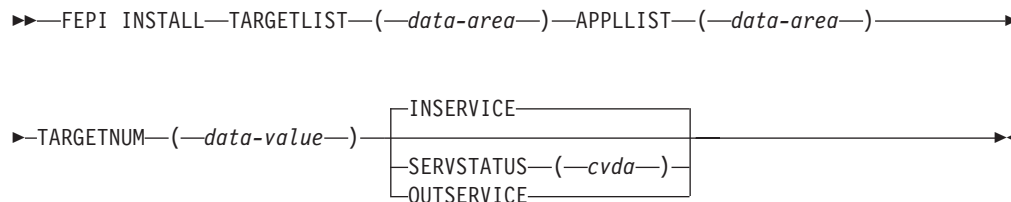
- 11 FEPI not installed or not active.
- 140 DEVICE value not valid.
- 141 CONTENTION value not valid.
- 142 INITIALDATA value not valid.
- 143 UNSOLDATAACK value not valid.
- 144 MSGJRNL value not valid.
- 150 FORMAT value not valid or is unsuitable for the LU mode and device type specified by the DEVICE value.
- 153 STSN name not valid or STSN is not allowed for the LU mode and device type specified by the DEVICE value.
- 154 BEGINSSESSION name not valid.
- 155 UNSOLDATA name not valid.
- 156 EXCEPTIONQ name not valid.
- 157 FJOURNALNUM value not valid.
- 158 MAXLENGTH value not valid.

- 159 ENDSSESSION name not valid.
- 160 PROPERTYSET name not valid.
- 170 PROPERTYSET name already exists.
- 178 FJOURNALNAME value not valid.

FEPI INSTALL TARGETLIST

Define new targets to FEPI.

FEPI INSTALL TARGETLIST



Description

FEPI INSTALL TARGETLIST defines new targets to FEPI. You can specify an initial service state for these new targets. A target cannot be used for a conversation until it has been put in service, and has been added to a pool so that it is connected to a node. The command completes when the targets have been installed without waiting for the requested states to be achieved.

Options

APPLIST (data-area)

specifies a contiguous array of 8-character primary logical unit (PLU) names. These are the VTAM application names (APPLID) of the back-end CICS or IMS systems with which FEPI applications are to communicate; they correspond one-to-one with the target names in TARGETLIST. The names must not contain null characters (X'00'), leading blanks, or embedded blanks. Each name must be unique within the list; duplicate names result in an INVREQ condition being returned.

If a target specified in TARGETLIST is a CICS terminal-owning region that is a member of a VTAM generic resource group, you can specify in APPLIST its generic resource name. This enables you to use the VTAM generic resource function to balance sessions across the available TORs. See "Workload balancing in a sysplex" on page 26.

SERVSTATUS (cvda)

specifies the initial service state of the targets being defined. All the targets in the list have the same state. The relevant CVDA values are:

INSERVICE

The target is in service and can be used in a conversation.

OUTSERVICE

The target is out of service and cannot be used for any conversation.

TARGETLIST (data-area)

specifies a contiguous array of 8-character target names to be defined. A target name is the logical FEPI front-end name of a back-end system. The names

must not contain null characters (X'00'), leading blanks, or embedded blanks. Each name must be unique within the list; duplicate names result in an INVREQ condition being returned.

TARGETNUM(fullword binary data-value)

specifies the number of names in TARGETLIST, in the range 1–256.

Conditions

INVREQ

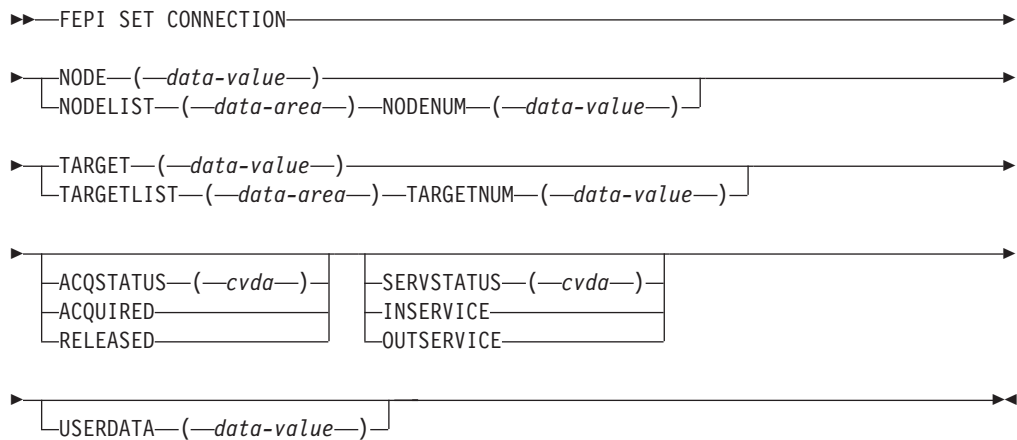
RESP2 values:

- 11 FEPI not installed or not active.
- 110 SERVSTATUS value not valid.
- 119 The command failed for one or more items in the list.
- 130 TARGETNUM value out of range.
- 164 TARGET name not valid.
- 167 Application name not valid.
- 174 TARGET name already exists.
- 177 Application name already exists.

FEPI SET CONNECTION

Set a FEPI connection.

FEPI SET CONNECTION



Description

FEPI SET CONNECTION controls the use of FEPI connections. Lists may be used to set more than one connection at a time; all connections in the list are set to the same state. The command completes immediately, although the requested settings may not be achieved until later.

Options

ACQSTATUS(cvda)

specifies the acquire state of the connection; that is, whether a session should be established ('bound') or not ('unbound'). The relevant CVDA values are:

ACQUIRED

The connection is to have a session established (that is, 'bound'). The state is ACQUIRING until this is completed.

RELEASED

The connection is to have its session ended (that is, 'unbound'), when usage of the connection by all owned conversations ends. (An unowned conversation on the connection is ended immediately. See the STATE option of "FEPI INQUIRE CONNECTION" on page 96.) The state is RELEASING until this is completed.

If this option is not coded, the acquire state is not changed.

NODE(8-character data-value)

specifies the node name that identifies a connection.

NODELIST(data-area)

specifies a contiguous array of 8-character node names identifying connections.

NODENUM(fullword binary data-value)

specifies the number of node names in NODELIST, in the range 1–256.

SERVSTATUS(cvda)

specifies the service state of the connection; that is, whether the connection can be used for a conversation or not. The relevant CVDA values are:

INSERVICE

Allows usage of the connection in a conversation.

OUTSERVICE

Stops usage of a connection for any new conversation, although existing conversations are unaffected. The service state is GOINGOUT until these conversations end.

If this option is not coded, the service state is not changed.

TARGET(8-character data-value)

Specifies the target name that identifies a connection.

TARGETLIST(data-area)

specifies a contiguous array of 8-character target names identifying a connection or connections.

TARGETNUM(fullword binary data-value)

specifies the number of target names in TARGETLIST, in the 1–256.

USERDATA(64-character data-value)

Specifies optional user data relating to the connections; it is not used by FEPI. It replaces any previous user data that was set.

Conditions

INVREQ

RESP2 values:

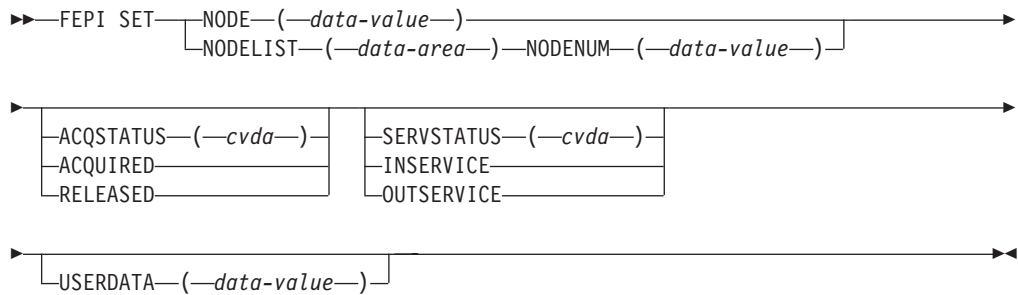
- 110** SERVSTATUS value not valid.
- 111** ACQSTATUS value not valid.
- 116** TARGET name unknown.
- 117** NODE name unknown.
- 118** Unknown connection (TARGET and NODE names are known but not connected in any pool).

- 119 The command failed for one or more items in the list.
- 130 TARGETNUM value out of range.
- 131 NODENUM value out of range.

FEPI SET NODE

Control the use of FEPI nodes.

FEPI SET NODE



Description

FEPI SET NODE controls the use of FEPI nodes. Lists may be used to set more than one node at a time; all nodes in the list are set to the same state. The function completes immediately, although the requested settings may not be achieved until later.

Options

ACQSTATUS (cvda)

specifies the acquire state of the node; that is, whether its VTAM ACB should be opened or closed. The relevant CVDA values are:

ACQUIRED

The VTAM ACB for the node is to be opened and 'set logon start' is to be done. The state is ACQUIRING until this is completed.

RELEASED

The VTAM ACB for the node is to be closed when usage of the node by any conversation ends. The state is RELEASING until this is completed.

If this option is not coded, the acquire state is not changed.

NODE (8-character data-value)

specifies the node to be set.

NODELIST (data-area)

specifies a contiguous array of 8-character node names to be set.

NODENUM (fullword binary data-value)

specifies the number of node names in NODELIST, in the range 1-256.

SERVSTATUS (cvda)

specifies the service state of the node; that is, whether the node can be used for a conversation or not. The relevant CVDA values are:

INSERVICE

Allows usage of the node in a conversation.

OUTSERVICE

Stops usage of a node for any new conversation, although existing conversations are unaffected. The service state is GOINGOUT until these conversations end.

If this option is not coded, the service state is not changed.

USERDATA(64-character data-value)

Specifies optional user data relating to the nodes; it is not used by FEPI. It replaces any previous user data that was set.

Conditions

INVREQ

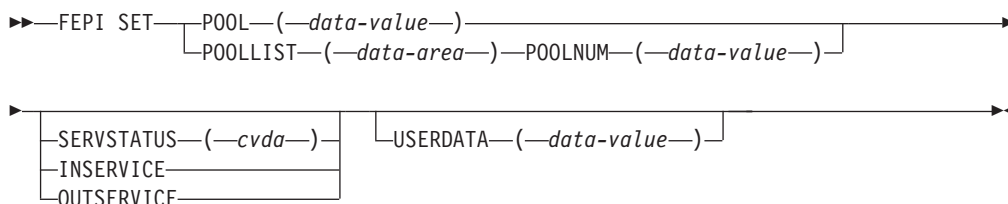
RESP2 values:

- 110 SERVSTATUS value not valid.
- 111 ACQSTATUS value not valid.
- 117 NODE name unknown.
- 119 The command failed for one or more items in the list.
- 131 NODENUM value is out of range.
- 174 The VTAM OPEN ACB failed.

FEPI SET POOL

Set the use of FEPI pools.

FEPI SET POOL



Description

FEPI SET POOL controls the use of FEPI pools. Lists may be used to set more than one pool at a time; all pools in the list are set to the same state. The function completes immediately, although the requested settings may not be achieved until later.

Options

POOL(8-character data-value)

specifies the pool to be set.

POOLLIST(data-area)

specifies a contiguous array of 8-character pool names to be set.

POOLNUM(fullword binary data value)

specifies the number of pool names in POOLLIST, in the range 1–256.

SERVSTATUS (cvda)

specifies the service state of the pool; that is, whether the pool can be used for a conversation or not. The relevant CVDA values are:

INSERVICE

Allows usage of the pool in a conversation.

OUTSERVICE

Stops usage of a pool for any new conversation, although existing conversations are unaffected. The service state is GOINGOUT until these conversations end.

If this option is not coded, the service state is not changed.

USERDATA (64-character data-value)

Specifies optional user data relating to the pools; it is not used by FEPI. It replaces any previous user data that was set.

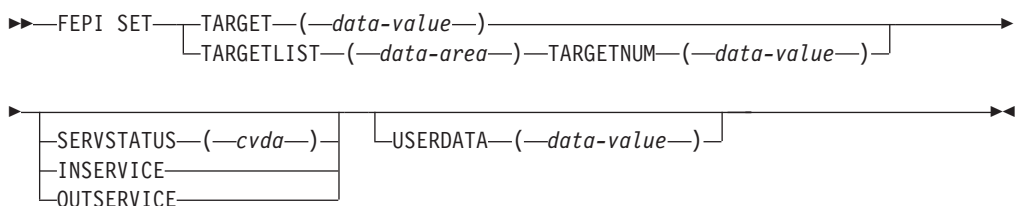
Conditions**INVREQ**

RESP2 values:

- 110 SERVSTATUS value not valid.
- 115 POOL name unknown.
- 119 The command failed for one or more items in the list.
- 132 POOLNUM value is out of range.

FEPI SET TARGET

Set the use of FEPI targets.

FEPI SET TARGET**Description**

FEPI SET TARGET controls the use of FEPI targets. Lists may be used to set more than one target at a time; all targets in the list are set to the same state. The function completes immediately, although the requested settings may not be achieved until later.

Options**SERVSTATUS (cvda)**

specifies the service state of the target; that is, whether the target can be used for a conversation or not. The relevant CVDA values are:

INSERVICE

Allows usage of the target in a conversation.

OUTSERVICE

Stops usage of a target for any new conversation, although existing conversations are unaffected. The service state is GOINGOUT until these conversations end.

If this option is not coded, the service state is not changed.

TARGET(8-character data-value)

specifies the name of the target to be set.

TARGETLIST(data-area)

specifies a contiguous array of 8-character target names to be set.

TARGETNUM(fullword binary data-value)

specifies the number of target names in TARGETLIST, in the 1–256.

USERDATA(64-character data-value)

Specifies optional user data relating to the targets; it is not used by FEPI. It replaces any previous user data that was set.

Conditions

INVREQ

RESP2 values:

- 110 SERVSTATUS value not valid.
- 116 TARGET name unknown.
- 119 The command failed for one or more items in the list.
- 130 TARGETNUM value is out of range.

FEPI SP NOOP

FEPI SP NOOP

▶▶—FEPI SP NOOP—◀◀

Description

FEPI SP NOOP has no effect.

Options

None.

Conditions

None specific to this command.

Transient data queue records

In response to various unexpected events, FEPI writes a record, describing the event and its circumstances, to a transient data (TD) queue.

Such events include:

- Errors in functions initiated by a system programming command
- Errors for items in a list on a system programming command

- Events unrelated to any command.

If the event relates to a specific pool, the record is written to the queue specified by EXCEPTIONQ for that pool; if EXCEPTIONQ was not specified, no record is written. If the event does not relate to a specific pool, the record is written to queue CSZX. In all cases, if the appropriate TD queue does not exist or if it is not defined as non-recoverable, the record is lost.

The format of the record is as follows. The copy books DFHSZAPA, DFHSZAPO, DFHSZAPC, and DFHSZAPP (according to your programming language) provide declarations for this record structure.

DATATYPE	Fullword binary data-area
EVENTTYPE	CVDA
EVENTVALUE	Fullword binary data-area
EVENTDATA	8-character data-area
Reserved	4-character data-area
POOL	8-character data-area
TARGET	8-character data-area
NODE	8-character data-area
CONVID	8-character data-area
DEVICE	CVDA
FORMAT	CVDA
Reserved	8-character data-area.

Fields

CONVID(8-character data-area)

the conversation ID for which the event occurred; null if not applicable.

DATATYPE(fullword binary data-area)

identifies the type and structure of the data. A value of 2 indicates FEPI TD queue data.

DEVICE(cvda)

the device type of the conversation for which the event occurred (the values are as for FEPI INQUIRE POOL); zero if not applicable.

EVENTDATA(8-character data-area)

contains data about the event:

Event	Data
ACQFAIL	2 fullword binary numbers: <ul style="list-style-type: none"> • VTAM reason code • Count
SESSIONFAIL	2 fullword binary numbers: <ul style="list-style-type: none"> • VTAM reason code • Count
SESSIONLOST	2 fullword binary numbers: <ul style="list-style-type: none"> • VTAM reason code • Count
Others	Nulls

If the count is nonzero, it indicates the number of times the node acquire or session start has failed; it will be tried again. A zero count indicates that several failures have occurred and that there will be no further attempts to acquire the node or start the session.

EVENTTYPE(cvda)

indicates what the event was.

Exceptional events queued to common TD queue CSZX:

ACQFAIL	A node could not be acquired (its VTAM ACB could not be opened).
DISCARDFAIL	A resource in a list could not be discarded by FEPI DISCARD.
INSTALLFAIL	A resource in a list could not be installed by FEPI INSTALL.
SESSION	An unsolicited bind was received.
SETFAIL	A connection or resource in a list could not be set by FEPI SET or FEPI INSTALL.

Exceptional events queued to pool-specific TD queue:

ADDFAIL	A connection in a list could not be added to the pool by FEPI ADD.
DELETEFAIL	A connection in a list could not be deleted from the pool by FEPI DELETE.
SESSIONFAIL	Session could not be started.
SESSIONLOST	Active session was lost.

EVENTVALUE(fullword binary data area)

provides further information about the event. Values are:

Event	Value
ACQFAIL	0
ADDFAIL	The RESP2 value describing the failure, as given in the description of the FEPI ADD command
DELETEFAIL	The RESP2 value describing the failure, as given in the description of the FEPI DELETE command
DISCARDFAIL	The RESP2 value describing the failure, as given in the description of the FEPI DISCARD command
INSTALLFAIL	The RESP2 value describing the failure, as given in the description of the FEPI INSTALL command
SESSION	0
SESSIONFAIL	The RESP2 value describing the communication failure; it can be any of the RESP2 values in the range 182–199.
SESSIONLOST	The RESP2 value describing the communication failure; it can be any of the RESP2 values in the range 182–199.
SETFAIL	The RESP2 value describing the failure, as given in the description of the FEPI SET command

FORMAT(cvda)

the data format of the conversation for which the event occurred (the values being as for FEPI INQUIRE POOL); zero if not applicable.

NODE(8-character data-area)

the name of the node for which the event occurred; nulls if not applicable.

POOL(8-character data-area)

the name of the pool for which the event occurred; nulls if not applicable.

TARGET (8-character data-area)

the name of the target for which the event occurred; nulls if not applicable. For the SESSION event, it is the VTAM application name of the back-end system, rather than the FEPI target name.

Reserved

nulls.

Chapter 12. FEPI problem determination

This section contains guidance information to help you identify the source of errors that affect your FEPI applications.

For information about using CICS debugging tools, trace, and dump, see the *CICS Problem Determination Guide*.

This section contains Diagnosis, Modification or Tuning information. It contains the following topics:

- “Debugging FEPI applications”
- “FEPI dump”
- “FEPI trace” on page 130
- “FEPI messages” on page 131
- “FEPI abends” on page 132
- “Reporting a FEPI problem to IBM” on page 133.

Debugging FEPI applications

The CICS execution diagnostic facility (EDF) helps users of the EXEC CICS interface to step through the EXEC CICS commands of an application program.

EDF can be used in just the same way to debug programs that use the EXEC CICS FEPI commands.

FEPI dump

CICS dump routines are available for FEPI. These routines are under the control of the usual CICS selection mechanisms.

You generate interpretation of the FEPI areas of a CICS dump by specifying the SZ keyword from within the interactive problem control system (IPCS). SZ can take the following values:

SZ value

What is printed

- | | |
|---|---|
| 0 | No FEPI areas are interpreted. |
| 1 | All FEPI areas are interpreted, excluding the stacks. |
| 2 | All FEPI areas are interpreted, including the stacks. |

If you are looking at a FEPI problem, first ensure the SZ TCB is active, and the FEPI Resource Manager is running. Look at the kernel and dispatcher prints to verify their presence.

If the SZ TCB is present, and the FEPI Resource Manager is running, the problem is probably caused by a wait or an abend. In the case of a wait, the dispatcher and kernel prints should show where it is located.

After looking at any FEPI trace entries, you should direct your attention to the output from the ‘SZ=2’ dump formatting keyword. This displays all known FEPI

control blocks. If you think a storage violation has occurred, use the dump storage manager options to display the contents of the FEPI storage subpools.

Here are some things that might help you identify a problem when you read the dump:

- Were any errors reported during interpretation? If so, this may indicate a corrupt address pointer or a broken chain.
- Follow all the pointers to associated control blocks (such as the conversation pointed to by the connection). Is this pointer correct? If not, this probably indicates corruption.
- Are there the expected numbers of nodes, targets, property sets, and pools? If not, this can indicate a broken chain or an unauthorized deletion.
- Does each pool contain the expected number of connections (that is, the number of nodes multiplied by the number of targets)? If not, this may indicate the failure of a FEPI ADD command.
- Has each node been successfully acquired? If not, there is the possibility of VTAM definition errors. The ACB and RPL may contain VTAM sense information—perhaps a VTAM major node is inactive.
- Is there successful communication with a target? If not, have APPLID and PASSWORD been correctly specified? If they are correct, is the back-end system running?
- Are there any queued ALLOCATE commands? If so, this indicates that there are not enough connections for the pool to process FEPI conversations without queuing. This may be acceptable, or not, depending on your configuration.
- Are the event handlers being run? If not, have they been correctly defined to CICS using RDO?
- Are the event handlers being recursively invoked? If so, this indicates a problem with a FEPI FREE command, a storage violation, or an internal logic error.
- Is information being correctly sent to the specified transient data queues? If not, are the queues defined as unrecoverable? Investigation of the DCT may help here.
- Are transactions being triggered from the TDQs? If not, are the transactions correctly defined to CICS?
- Is there a current conversation? If so, this conversation may be causing the error. Is the data correct? Is there any VTAM sense information in the RPL?
- Are the surrogate terminals correct? If not, the links between the nodes, pools, and targets may have become corrupted.
- Are FEPI SEND or FEPI RECEIVE commands failing due to state errors? If so, look at the conversation and see if the states are correct. If they are not, the conversation has become out of step with the VTAM flow.
- Is unexpected data being sent or received in formatted conversations? If so, there may be corrupt FEPI data. Look at FEPI's internal terminal character buffer.
- Look at the queues. Are there any requests that look as if they have got stuck? If so, the FEPI work chains may be corrupt. However, it may be that the flow to satisfy the requests has not yet happened. If you think it should have happened, there may be communication problems.
- Look at the FREE queue. The last VTAM event may be shown. If so, does it correspond with what you expected?

- Is the behavior of a pool correct? If not, it is possible that the property set used to define the pool is incorrect. However, if the property set is shown, it could have been re-created since the pool was defined—treat property set definitions with care.
- Are there any outstanding timer events that should have run? If so, this may indicate a chaining failure.
- Has a timer-dependent action been delayed? If so, this could indicate that the TIMEOUT parameter on the command was incorrect.
- Are you receiving all the data you expect? If not, have you set the correct end-of-flow condition on the FEPI RECEIVE (or CONVERSE) command?
- Are there many transactions waiting on FEPI? If so, either back-end systems are not responding, or the FEPI Resource Manager has failed.
- Has a VTAM dump been taken? If so, this may indicate a failure in one of the VTAM exits.

Using CICS dump facilities to investigate FEPI problems

This section describes how FEPI relates to the rest of CICS, and how its presence is revealed by the other CICS dump formatting commands.

The problem determination process for FEPI is driven from the usual CICS dump interpretation routines. The following sections describe what to look for in the major CICS areas.

Dispatcher

You should see a task (CSZI) running under the SZ task control block.

(However, note that CSZI can run under the QR TCB while executing certain CICS functions, such as starting transactions and writing to transient data queues.) If CSZI is not present, then either FEPI is not in the system, or the FEPI Resource Manager has failed.

Application programs waiting for responses from the FEPI Resource Manager are shown as waiting on FEPI. (For details of FEPI waits, see the *CICS Problem Determination Guide*.)

Interval control

Any transactions that have been started by the FEPI Resource Manager, but not yet run, appear in the interval control section.

Kernel

In the kernel, you should find a running task named KETCB SZ representing the SZ TCB that FEPI uses. If KETCB SZ is not present, then either FEPI is not in the system, or the TCB has abended.

You should find the CSZI task either running or waiting. If CSZI is not present, then either FEPI is not in the system, or the FEPI Resource Manager has failed.

If an abend has occurred, the usual information is available. The location of the abend is indicated by the failing module, as follows:

DFHESZ

The application programming EXEC stub

DFHEIQSZ

The system programming EXEC stub

DFHSZATR

The FEPI adapter

DFHSZRMP

The FEPI Resource Manager.

Storage manager

You can use the storage manager dump facilities to display the contents of the subpools used by FEPI.

If you suspect a storage violation, a comparison of the contents of these subpools with the areas interpreted by a FEPI dump might show where the corruption has occurred.

Table 9. FEPI storage subpools

Name	Type	Chained	Above or below 16MB line?	Usage
SZSPFCAC	Fixed	Yes	Below	ACBs
SZSPFCCD	Fixed	Yes	Any	Connections
SZSPFCCM	Fixed	Yes	Any	Common area
SZSPFCCV	Fixed	Yes	Any	Conversations
SZSPVUDA	VAR	Yes	Any	Various data areas
SZSPFCDS	Fixed	Yes	Any	Device support extensions
SZSPFCDT	Fixed	Yes	Any	Device-type control areas
SZSPFCNB	Fixed	Yes	Any	NIBs
SZSPFCND	Fixed	Yes	Any	Nodes
SZSPFCPD	Fixed	Yes	Any	Pools
SZSPFCPS	Fixed	Yes	Any	Property sets
SZSPFCRP	Fixed	Yes	Any	RPLs
SZSPFCRQ	Fixed	Yes	Any	Requests
SZSPFCSR	Fixed	Yes	Any	Surrogates
SZSPFCTD	Fixed	Yes	Any	Targets
SZSPFCWE	Fixed	Yes	Any	DQEs

FEPI trace

There are appropriate trace entries in the CICS trace table which are under the control of the usual CICS mechanisms.

FEPI trace entries are listed in the *CICS Trace Entries* manual.

FEPI generates exception and event trace entries - the latter under control of the 'SZ' component code. Points AP 1200 through AP 16FF are reserved for use by FEPI, although not all of these are used.

Taking and interpreting trace entries

FEPI supports only one level of tracing. At CICS initialization, you can specify the default levels of standard and special tracing using the **STNTR**, **SPCTR**, **STNTRSZ**, and **SPCTRSZ** system initialization parameters.

About this task

These parameters are described in the *CICS System Definition Guide*. Exception trace entries are always taken.

To take trace entries, you can either:

Procedure

1. You can start tracing using either of these methods:
 - Use the CETR transaction. The FEPI component code on the CETR “Component Trace Options” panel is ‘SZ’. Specify ‘SZ 1’ to turn on FEPI tracing.
You can use the selection features of the CETR transaction to limit tracing to specific transactions rather than using the SZ component. If you do this, you can control the tracing of application programs. The FEPI Resource Manager, running as the CSZI transaction, is unaffected, because trace selection is applied only at transaction start.
 - Use the following command: **SET TRACETYPE SZ**.
2. Check if there are any exception trace entries. Their presence indicates that a problem has been detected. Exception trace entries are either initialization errors or storage management errors.
 - a. Initialization errors result from checks made when CSZI starts, to prevent a second instance of the FEPI Resource Manager.
 - b. Storage errors result from GETMAIN or FREEMAIN errors, and are usually caused by a lack of CICS storage.
3. Other trace entries are the usual module entry and exit traces, together with a few points indicating that important processing events have occurred (such as the FEPI Resource Manager becoming idle).

What to do next

If you are using DFHTRAP under the guidance of IBM support, note that the FEPI Resource Manager runs under the SZ TCB. Therefore, do not do anything that could force an MVS task switch to any other TCB.

FEPI messages

Messages produced by FEPI have exactly the same format (DFHSZ...) as other CICS messages. They are all sent to the FEPI message log (the CSZL transient data queue); some are also sent to the operator.

FEPI messages are documented in the *CICS Messages and Codes* manual, and are also available through the CMAC transaction.

FEPI abends

FEPI does not (deliberately) issue either CICS transaction abends or MVS abends.

However, an unexpected failure can occur in the following places:

- In a FEPI application program when INVREQ is returned
- In the EXEC stubs
- In the FEPI adapter
- In the FEPI Resource Manager transaction (CSZI) code
- In a VTAM exit routine.

These abends have different results, as shown in Table 10.

Table 10. Types of abend issued by FEPI

Point of failure	Result
Application	The usual transaction abend for the error condition.
EXEC stubs	The usual transaction abend for a failure within CICS management modules. An example of this is an 'operation' program check, which generates a CICS AKEA abend, which in turn generates an ASRA abend.
FEPI adapter	The usual transaction abend for a failure within CICS management modules. An example of this is an 'operation' program check, which generates a CICS AKEA abend, which in turn generates an ASRA abend.
FEPI Resource Manager	No direct effect on the application program, because the abend occurs under the CSZI Resource Manager task. This probably results in a DFHSZ4099E message (see "Message DFHSZ4099E"), and the failure of the Resource Manager. An example of this is an 'operation' program check, which generates a CICS AKEA abend, which in turn generates an ASRA abend. Any CICS FEPI transactions are left waiting on the FEPI_RQE resource (for details of FEPI waits, see the <i>CICS Problem Determination Guide</i>).
VTAM exit	A VTAM abend; a VTAM dump is taken. Because the exit lies within the FEPI Resource Manager, the CICS abend handling routines are activated to process a "normal" failure in the Resource Manager.

Restart

An abend in an application program, an EXEC stub, or the FEPI adapter affects only the active CICS task that issued the FEPI command; other FEPI programs continue as normal.

If an abend affects the SZ TCB, CICS makes that TCB unavailable for use, while keeping the other CICS TCBs active and accessible. This means that FEPI functions can be restored only by restarting the CICS system.

Message DFHSZ4099E

This message indicates that the abend exit routine within the FEPI adapter has trapped an abend within the FEPI Resource Manager.

As soon as an abend within the Resource Manager is detected, the FEPI state (in the FEPI static area) is set to 'Failed'. If possible, message DFHSZ4099E is issued,

together with a SNAP dump, to indicate that FEPI has failed. However, in some circumstances it is not possible to issue DFHSZ4099E, and a system dump is generated instead.

Any FEPI transactions are left waiting on the FEPI_RQE resource (for details of FEPI waits, see the *CICS Problem Determination Guide*. These waits never get posted, so the transactions suspend. You must issue a CEMT FORCEPURGE command to remove these suspended transactions from the system.

Attention: It is strongly recommended that the CSZI transaction is initiated only as part of CICS system initialization. *Do not attempt to restart the CSZI transaction after a failure, other than by restarting CICS.*

Message DFHSZ4155I

This message indicates that a connection has ended, and gives a reason code taken from the VTAM control blocks. The reason code may be returned in the LASTACQCODE option of a CEMT or FEPI INQUIRE command, depending on the operation which generated DFHSZ4155I.

DFHSZ4155I does not always indicate a problem; if you took positive action to end the connection, DFHSZ4155I merely confirms that VTAM did as you requested. However, if the connection ended unexpectedly, the reason code tells you why.

To determine what the reason code means, refer to the *VTAM Programming* manual.

Reporting a FEPI problem to IBM

When reporting a problem to IBM Support, you must provide details of the CICS system in which FEPI is installed.

About this task

You must provide the following information:

- All listings from the CICS job, including the CICS job log and JCL
- A print of all reports sent to the CSZL transient data queue
- A full system dump (including the CSA and LSQA)
- Any relevant transaction dumps
- All trace entries (you may need to re-create the problem with SZ trace active)
- A listing of the application program that detected the problem
- Listings of the programs used to configure your FEPI system
- Listings of any active CICS global user exit programs (not only the FEPI ones)
- Prints of user journals, if FEPI journaling was active when the problem occurred.

The following materials might also be required:

- A VTAM trace showing the data flows
- A trace of the back-end system showing what data streams were received from FEPI application programs
- A VTAM status display showing the status of FEPI connections
- Any dumps or logs produced by the back-end system.

Part 3. FEPI application programming

This part of the book is primarily for application programmers and includes reference information for FEPI application programming commands.

- Chapter 13, “Basics of FEPI programming,” on page 137 introduces FEPI programming and the commands that are used.
- Chapter 14, “FEPI key stroke and screen-image applications,” on page 141 discusses the high-level interface for FEPI applications.
- Chapter 15, “FEPI data stream applications,” on page 149 describes the low-level interface for FEPI applications.
- Chapter 16, “FEPI application design,” on page 157 describes the programs comprising a FEPI application and various design aspects such as conversation ownership, handling errors, and specific requirements for CICS and IMS back-end systems.
- Chapter 17, “Specialized FEPI functions,” on page 173 describes control functions, normally handled by FEPI, that can be taken over by FEPI applications.
- Part 4, “FEPI application programming reference,” on page 175 contains reference information for the FEPI commands that are used in application programs.

Chapter 13. Basics of FEPI programming

This section introduces FEPI programming and the FEPI commands that you can use.

Before reading this section you should be familiar with the FEPI concepts and facilities described in Chapter 1, “Introducing FEPI,” on page 3 and Chapter 4, “FEPI functions and services,” on page 11.

To write FEPI front-end applications, you need to know how to write programs in at least one of the programming languages that CICS supports. More importantly, you also need knowledge of data communication and protocols. And, if you will be accessing IMS back-end systems, you must also be familiar with using IMS and writing IMS applications.

The applications that you write using FEPI are normal CICS transactions with the familiar EXEC CICS commands. These FEPI applications use the FEPI subset of EXEC CICS application programming commands to:

- Allocate a connection from a pool
- Communicate with a back-end application using this connection
- Free the connection when finished.

The section contains the following topics:

- “Communication and conversations”
- “Structure and design” on page 138.

Communication and conversations

A FEPI application runs in a **front-end** CICS system and accesses applications in a **back-end** CICS or IMS system. FEPI lets it do this by simulating a terminal connected to the back-end system; this means that it has to act just like a real terminal and terminal operator.

Note: The highlighted terms in this section are defined in Chapter 4, “FEPI functions and services,” on page 11.

The back-end systems are known as **targets** and the **connections** to them are arranged in **pools** that define the properties controlling communication. Targets, pools, and properties are defined by your system programmer, who can tell you which targets and pools to use and what properties they have.

When a connection has been established, on successful completion of a bind, the connection is **in session** and it can be allocated by FEPI for a **conversation** with the back-end system.

Conversations are the basis of all FEPI applications and, depending upon the needs of your application, may be used in several ways (see Chapter 16, “FEPI application design,” on page 157):

- A single conversation for all transactions on a back-end system
- A different conversation for each transaction or associated series of transactions
- A special conversation to handle unusual events.

The task that started the conversation owns it and other tasks cannot issue commands for it; however, the owning task can transfer ownership to another task. You can have as many conversations as you like at a time with various targets: they can be consecutive or, much more usefully, interleaved.

FEPI simulates a 3270-type terminal (SLU2 mode) for both CICS and IMS systems; it also supports the SLU P mode that is used by IMS for programmable terminals such as the 4700 family. The mode to be used, SLU2 or SLU P, is a property of the pool being used. Your application cannot change the mode of a conversation.

The data that you send and receive can be **formatted** or **data stream** and, as with mode, the data type is a property of the pool being used:

Formatted

A high-level data interface for SLU2 mode. The data sent by the FEPI application can be either **key stroke** format or **screen-image** format; data received by the application is in screen-image format.

Data stream

A low-level data interface for more sophisticated SLU2 mode applications and for use with SLU P mode. The data sent and received by the FEPI application is the data stream; applications using this format have access to some very specialized VTAM communication functions.

The same basic set of FEPI commands is used for all modes and data types and protocols, but the command options and keywords are generally different.

Structure and design

In addition to your main access program that handles communication with the back-end system, you might need to provide programs for other functions.

These functions are as follows:

Begin session

Handle begin-session processing.

Unsolicited data

Handle unsolicited inbound data that arrives when there is no conversation.

End session

Handle end of conversation and end of session processing.

These functions could be combined in one program or implemented in separate programs with individual transaction names. There may be any number of each function, again according to your requirements and preferences. Suggestions about the various possibilities are given later.

As the application programmer, you will always write the main access programs. Sometimes the system programmer provides any special functions that are required; otherwise you would be responsible for these. Even if you are writing only the main access program, you need to be aware of what these special functions do and how they affect how you communicate with the back-end system. Because the use of these special functions is controlled by the pools that you use, you need to liaise with the system programmers or administrators who set them up.

Several different styles of access program are possible:

One-out one-in conversational

One program performs the complete conversation with the target and each conversation has a single transmission to and from the back-end system.

Conversational

One program performs the complete conversation with the target with multiple transmissions to and from the back-end system, waiting each time for the inbound data.

Pseudoconversational

Here, one program sends data to the target and requests CICS to start another program when the inbound data arrives.

The section beginning with Chapter 14, "FEPI key stroke and screen-image applications," on page 141 and ending with Chapter 17, "Specialized FEPI functions," on page 173 describes the various features of writing application programs. A set of sample programs is available to help you to get started; these are supplied as source code on the distribution tape. For details, see Appendix A, "FEPI sample programs," on page 237.

Programming

FEPI programs are CICS applications, so all aspects of CICS programming apply. The FEPI application programming commands are an extension of the EXEC CICS commands. They have similar names and similar functions. The FEPI commands also have similar keywords, but they are distinguished by having FEPI as a prefix.

Your FEPI application programs can be AMODE(24) or AMODE(31) - that is, they can issue FEPI commands in either 24- or 31-bit addressing mode, and reside above or below the 16MB line.

The application programming commands are:

EXEC CICS FEPI ALLOCATE

Starts a conversation with a back-end system.

EXEC CICS FEPI FREE

Ends the conversation with a back-end system.

EXEC CICS FEPI REQUEST PASSTICKET

Requests the external security manager to supply a password substitute.

EXEC CICS FEPI SEND

Sends data to the back-end system.

EXEC CICS FEPI RECEIVE

Receives data from the back-end system.

EXEC CICS FEPI CONVERSE

Sends data to and receives data from the back-end system.

EXEC CICS FEPI ISSUE

Sends control data to the back-end system.

EXEC CICS FEPI EXTRACT

Gets field data and attributes, set-and-test sequence number (STSN) data, or information about a conversation.

EXEC CICS FEPI START

Schedules a CICS transaction to handle inbound data.

Note that, when translating your programs, you must specify the FEPI option; this instructs the translator to process FEPI commands.

For general information about writing CICS application programs, see the *CICS Application Programming Guide*. For programming information (including command formats, argument values, details on the translation of programs, and language considerations), see the *CICS Application Programming Reference*. Particularly relevant are the sections in the *CICS Application Programming Guide* about designing efficient applications and dealing with exception conditions.

Related information

CICS API commands

Designing efficient applications

Dealing with exception conditions

Exception conditions

As with all CICS commands, FEPI commands may produce exception conditions that you can check using the RESP option, or capture using HANDLE CONDITION.

Most FEPI command errors return INVREQ. The particular error in each case is uniquely identified by the RESP2 value. All the FEPI exception conditions and RESP2 values are listed in Part 4, “FEPI application programming reference,” on page 175. There are copy books that contain declarations for the RESP2 values:

- DFHSZAPA for Assembler language
- DFHSZAPO for COBOL
- DFHSZAPP for PL/I
- DFHSZAPC for C.

If there is an error, the command does nothing, and output values are not changed. Note, however, that commands such as FEPI SEND may have transferred data before the condition is recognized.

You can use EDF and CECI to debug FEPI programs. Because FEPI commands can be quite long, you will probably find the NAME field of CECI useful.

Chapter 14. FEPI key stroke and screen-image applications

This section discusses the key stroke and screen-image data interfaces for FEPI applications.

The examples given in this section are confined to simple conversational applications. However, you can use this data interface whatever the application structure. See Chapter 16, “FEPI application design,” on page 157 for further possibilities together with full details of conversations, error handling, and system considerations.

The key stroke and screen-image data interface is suitable for a wide range of applications, and is simpler to use than the alternative data stream interface. However, there are certain types of application for which you cannot use screen-image data. For more details, see Chapter 15, “FEPI data stream applications,” on page 149.

You can send both key stroke and screen-image data in the same conversation. The inbound data format is the same for both: a screen-image, that you can also access field-by-field.

You must have general knowledge of data communication and protocols.

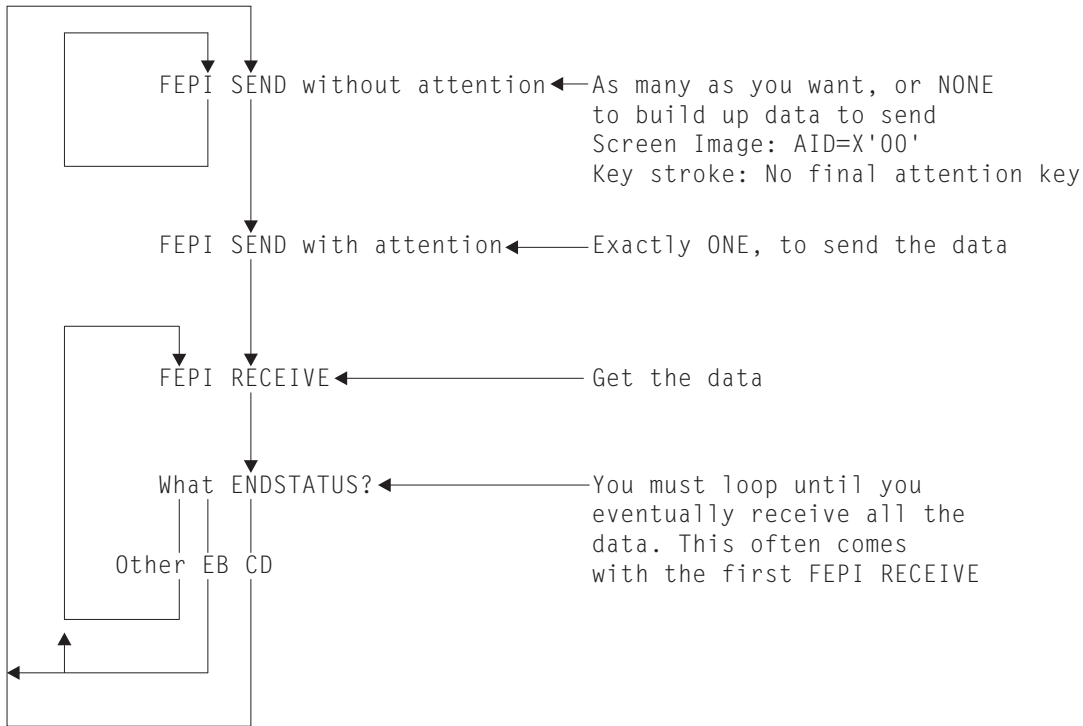
The section contains the following topics:

- “General sequence of commands”
- “Sending key stroke data” on page 142
- “Receiving field-by-field” on page 144
- “Multiple attentions” on page 145
- “Sending screen-image data” on page 146
- “Receiving screen-image data” on page 147
- “Extracting field data” on page 148
- “CONVERSE” on page 148.

General sequence of commands

The following diagram illustrates the general sequence of FEPI commands that you use with key stroke and screen-image data.

That is, a FEPI SEND, multiple FEPI RECEIVES that complete when all the data has been received, followed by another FEPI SEND.



Note: The diagram does not show any processing of the data, nor where you might enter, or leave, the loop. This information is explained more fully in Chapter 16, “FEPI application design,” on page 157.

Sending key stroke data

Sending key strokes is the easiest way of sending data.

Your program acts in the same way as the keyboard operator, with FEPI letting the program “press keys” just as the operator does.

A sample program illustrates the techniques used; see “Key stroke CONVERSE” on page 246.

The data can contain any combination of data characters together with manipulative, special, and attention key values representing almost every keyboard key. Data characters are represented as themselves. Manipulative, special, and attention key values are represented by *escape* sequences, comprising an escape character followed by a 2-character code. For example, using '&' for the escape character, you might send the following sequence to insert AB in one field, type IJKL into another field, and press PF7 to complete the input operation:

```
&H0&T2&R1&INAB&RS&N4IJKL&EF&07
Home
  Tab, twice
    Cursor right
      Insert
        AB
          Reset
            Newline, 4 times
              IJKL
                Erase–EOF
                  PF7
```

If the sequence were in a character string named KEY-SCRIPT, you would send it with:

```
EXEC CICS FEPI SEND FORMATTED
      CONVID(. . . .)
      KEYSTROKES
      FROM(KEY-SCRIPT)
      FLENGTH(30)
```

In full, the escape sequences are:

<i>Manipulative keys</i>	<i>Special keys</i>	<i>Attention keys</i>
&HO home	&IN insert	&AT attention
&Ln cursor left, n times	&DL delete	&An PAn (n = 1–3)
&Rn cursor right, n times	&RS reset	&nn PFnn (nn = 01–24, any leading 0 must be specified)
&Un cursor up, n times	&EF erase to end of field	
&Dn cursor down, n times	&EI erase input	&CL clear
&Tn tab, n times	&FM field mark	&CS cursor select (light pen)
&Bn backtab, n times	&DU DUP	&EN enter
&Nn newline, n times (n = 1–9)	&ES escape character	&ME end secure MSR
	&SO shift out	
	&SI shift in	
	&MS start secure MSR	

You can choose an alternative escape character.

Data characters must have values $\geq X'40'$, so nulls ($X'00'$) are not supported as such, although they can be generated using the erase or delete keys. Key strokes following an attempt to type into a protected field are ignored until RESET is keyed.

For magnetic stripe reader support, the sequence **&MS...data...&ME** represents passing a secure magnetic stripe card through the reader. Nonsecure cards have to be simulated by entering the data in the normal way.

The cursor position is set by your key strokes, rather than specifying where the cursor is placed. If your first key stroke is always the HOME key (**&HO**), you will have the cursor in a known starting position.

You can choose to send all the data with one command, or to use several commands to build up the data. The last (or only) command should have an attention key as its final key stroke, to send the data. There should be no other attention keys.

Alternatively, if you are not interested in the received data, you can ignore it by sending key strokes with multiple attention keys, as described in “Multiple attentions” on page 145.

Error handling

Apart from communication errors caused externally, there are two likely sorts of error that you might get.

These two sorts of errors are as follows:

- Bad command sequencing; that is, you have issued a FEPI SEND when one was not expected. A FEPI SEND must **not** follow a FEPI SEND with a final attention key, or a FEPI RECEIVE that did not indicate 'change direction'.
- Incorrect data; that is, your key strokes are improper. You may have:
 - Sent data, characters, or escape sequences that are not valid.
 - Got into an 'input inhibited' situation and not reset it.
 - Broken the rules for double-byte character set (DBCS) data.
 - Failed a validation test, if there are fields with one of the validation attributes.

Many of these data errors cannot be detected until the data is processed, because they depend on the previous data. This means that any key strokes preceding the error will already have taken effect—they cannot be removed by FEPI.

The FEPI SEND can also fail if, following end bracket, the back-end sends BID to send more data and your pool has CONTENTION(LOSE). You must then receive the new back-end data first.

Receiving field-by-field

Receiving data field-by-field is the easiest way of receiving data.

In the simplest case you would issue a FEPI RECEIVE command without specifying an INTO data area. FEPI gets the data from the back-end system and builds the resulting screen image internally. The cursor position is returned by the CURSOR option. Information about the number of lines, columns, and fields in the screen image is returned by the LINES, COLUMNS, and FIELDS options.

To get the data, you issue the FEPI EXTRACT FIELD command for each individual field that you want. As well as the data, you can find out the attribute settings for the field, and its length and position. The attribute values are defined in the DFHBMSCA copy book, as is used with BMS. You can issue as many FEPI EXTRACT FIELD commands as you need, for whichever fields you want. You can issue more than one for each field, for example, if you want to get the data and attributes separately. It is generally preferable to use the FIELDLOC option rather than FIELDNUM. There may be spurious attributes between each displayed field which make determining field numbers difficult.

A sample program illustrates the techniques used; see "Screen image RECEIVE and EXTRACT FIELD" on page 249.

Command completion

The FEPI RECEIVE command completes on 'end of chain'.

This normally coincides with 'change direction' or 'end bracket', meaning that all data has been received. In some cases, however, back-end applications may send data to you in several sections (chains), each causing a screen update, so you must keep on receiving data until 'change direction' or 'end bracket' is indicated.

In all cases, the ENDSTATUS option is set to indicate what the completion conditions were. Where several conditions occur together, ENDSTATUS shows the most significant one. The values of ENDSTATUS and their associated meanings are shown in Table 11 on page 145.

Table 11. ENDSTATUS values and associated meanings for formatted data

ENDSTATUS	End bracket	Change direction	End of chain	Next command expected
EB	Y	-	Y	Any
CD	-	Y	Y	FEPI SEND or CONVERSE
LIC	-	-	Y	FEPI RECEIVE
Note: Y=Condition indicated.				

When 'end bracket' is received, the session is in *contention state*, and either end may try to transmit data next. Some back-end systems use 'end bracket' in the middle of a series of transmissions to allow the terminal to break in if it wants, and they may use 'end bracket' instead of 'change direction' at the end of the flow. This is particularly true of IMS. CICS usually sends 'change direction' eventually, although it may send 'end bracket' indicators intermediately.

Using your knowledge of the back-end application and system, you must check the data that you have already received, to determine whether more data is to be expected or the transmission is complete. If more data is expected, you should issue another FEPI RECEIVE command; if the transmission is complete, it is the front-end application's turn to send data.

You should always use the TIMEOUT option on a FEPI RECEIVE command; see "Time-outs" on page 166.

Error handling

Apart from communication errors caused externally, the most likely error you might get is due to bad command sequencing. That is, you have issued a **FEPI RECEIVE** command when a **FEPI SEND** command is expected.

A **FEPI RECEIVE** command must not follow a **FEPI SEND** command without attention, or a **FEPI RECEIVE** command that indicated 'change direction'.

Another likely error is 'previous SEND failed'. This might be an external communication error, or the back-end system might have responded negatively—as IMS does, for example, if you try to run an unknown transaction. The sense data which you can get using **FEPI EXTRACT CONV** tells you which error it is, and, where the back-end system has responded negatively, you issue another **FEPI RECEIVE** command to get the data.

Multiple attentions

In certain circumstances you might not have any interest in the immediate result of the data you send, but only in a later result, after you have sent more data.

If this is the case, you can construct a single key stroke sequence, comprising all the sets of data to send, each with its own attention key, and then send the whole lot in one operation.

At each attention key, FEPI sends your data to the back-end system and receives the results internally, until 'change direction' or 'end bracket' is indicated. Then FEPI sends the next set of key strokes. Using multiple attentions improves performance but, if the intermediate results are not what you expect, FEPI has no

way of knowing this and carries on sending your key strokes. This can lead to unexpected effects, or to the failure of the command with a data error. In the latter case, all the key strokes and back-end system interactions preceding the error have already taken effect and you may find it difficult to determine the state of the back-end system. Further, no time-out can be specified for the intermediate receives, and so, if there is a communication problem, your application may be suspended indefinitely.

If the last set of key strokes ends with an attention key, you **must** issue a FEPI RECEIVE command to get the final result. If the last set of key strokes does not end with an attention key, you can issue another FEPI SEND command, with yet more key strokes.

Sending screen-image data

Sending screen-image data is an alternative to sending key stroke data. In general, this would be the screen image that you received modified to reflect the changes that would be the result of an operator action.

A sample COBOL program, DFH0VZTS, illustrates the techniques used; see “Screen image SEND and START” on page 247.

The data is exactly what you would expect: an image of the screen that you want to send. That is, 24 rows of 80 bytes (or whatever your screen size is) of data, corresponding byte-for-byte with the screen. For example, in a COBOL program containing this data description:

```
01 SCREEN-IMAGE PIC X(1920).
01 SCREEN-FIELDS REDEFINES SCREEN-IMAGE.
05 LINE-1 PIC X(80).
05 FILLER REDEFINES LINE-1.
10 FILLER PIC X(20).
10 CUST-NO PIC X(12).
10 FILLER PIC X(48).
05 LINE-2 PIC X(80).
05 LINE-3 PIC X(80).
05 LINE-4 PIC X(80).
05 FILLER REDEFINES LINE-4.
10 FILLER PIC X(12).
10 CUST-NAME PIC X(32).
10 FILLER PIC X(36).
```

you would put the required data into the fields and send the screen image using:

```
EXEC CICS FEPI SEND FORMATTED
      CONVID(...)
      FROM(SCREEN-IMAGE) FLENGTH(1920)
      AID(PF2)
```

where AID specifies which attention key was pressed on the simulated terminal.

Data bytes are represented as themselves; you must set any nulls (X'00') that are needed to fill a field. In a protected field, the data bytes must be the same as in the current, simulated terminal buffer that FEPI holds. In the case of attribute bytes, it does not matter what values you put, because you have no control over their positions or settings, any more than a terminal operator does. However, if the value is X'01', FEPI sets the modified data tag (MDT) for the field, even if its data has not changed. (If the data has changed, FEPI sets the MDT automatically.)

You do not have to send a complete screen image. If your changes are confined to the first few lines, you need only send those few lines. The data you send is taken as starting from the top left position of the screen.

Note: If you are using the C programming language, remember that a screen image probably contains null characters. Take care if you are handling the screen image as a string.

The cursor position can be set using the CURSOR option.

You can choose to send all the data with one command, or to use several commands to build up the data. The last (or only) command must have an attention identifier (AID) specified, using the AID option, to send the data. The other commands must have an AID value of X'00'. Definitions for the AID values are in the DFHAID copy book, as is used with BMS.

Note: The COBOL and assembler versions of the DFHAID copybook are different. Therefore, you cannot copy unmodified SEND commands from the DFH0VZTS sample program, which is supplied in COBOL only, to a user-written assembler program.

Error handling

The errors you can get are similar to those for key stroke data. Your screen-image data has other ways of being incorrect. In place of escape sequences not being valid, or 'input inhibited', you might have cursor or AID settings not valid, or changed data in a protected field. Many of these data errors cannot be detected until the data is processed. This means that some of the changes will have taken effect already - they cannot be removed by FEPI.

Receiving screen-image data

If you specify an INTO data area on a FEPI RECEIVE command, the data you receive is the screen image; 24 rows of 80 bytes (or whatever your screen size is) corresponding byte-for-byte with the screen.

Data bytes are represented as themselves. In positions corresponding to attribute bytes, X'FF' appears.

You need only get the first few lines of the screen if that is all that you are interested in.

After you have processed the data, you will probably use the same screen image, modified as required, on a subsequent screen-image send.

Even though you got a screen image, you can use the FEPI EXTRACT FIELD command as well if you want, for any particular fields that you require, just as described in "Receiving field-by-field" on page 144. In particular, the FEPI EXTRACT FIELD command is the only way you can determine the value of the field attributes.

A sample program illustrates the techniques you can use; see "Key stroke CONVERSE" on page 246.

Note: If you are using the C programming language, remember that a screen image probably contains null characters. Take care if you are handling the screen image as a string.

Command completion and errors

As far as completion and errors are concerned, a FEPI RECEIVE command with an INTO data area is just like one without.

So, if you do not get 'change direction' or 'end bracket', you have to issue another FEPI RECEIVE command before you can send your screen image back, and even 'end bracket' might require further FEPI RECEIVE commands.

Extracting field data

It is not only after a FEPI RECEIVE command that you can issue a FEPI EXTRACT FIELD command. You can issue this command anywhere in the conversation to find out about the current screen image that FEPI holds for the simulated terminal.

This can be particularly useful where a FEPI SEND command has failed or given unexpected results, to discover what happened.

CONVERSE

FEPI CONVERSE can be used instead of a FEPI SEND with attention and the first (or only) FEPI RECEIVE.

It is more efficient than issuing two separate commands and is allowed anywhere that FEPI SEND is allowed. The effects are exactly as if the two commands had been issued.

The ending conditions are identical to those for FEPI RECEIVE, unless you use the POOL option to get a temporary conversation. In this case, it ends on the first occurrence of either 'Change direction' or 'End bracket' and does not end at 'end of chain' alone.

Error handling

You must plan which command is expected next:

- If the receive part of the FEPI CONVERSE command fails, the send will have already been done, and so a FEPI RECEIVE command is expected next.
- If the send part fails, the receive is not done, and, if the initial send was expected, a FEPI SEND or CONVERSE command is expected next.

Chapter 15. FEPI data stream applications

This section discusses the low-level data stream interface for FEPI applications.

The examples it contains are confined to simple conversational applications. However, you can use this data interface whatever the application structure; see Chapter 16, “FEPI application design,” on page 157 for all the possibilities, together with details of conversations, error handling, and system considerations.

The section contains the following topics:

- “When to use the data stream interface”
- “General sequence of commands” on page 150
- “Receiving” on page 150
- “Sending” on page 153
- “CONVERSE” on page 153
- “SLU2 mode considerations” on page 154
- “SLU P mode considerations” on page 155.

When to use the data stream interface

You can use the data stream interface for the following types of applications:

- With passthrough; that is where the application passes data through, usually to the user’s terminal, without doing anything to it.
- With SLU P.
- Where the formatted interface does not provide the detailed function that you need.
- For handling non-3270 LU2 devices.
- With non-response mode IMS transactions.

The 3270 data stream interface is especially useful when creating FEPI applications that require little or no manipulation of the inbound (screen) data, because it is already in a form suitable for sending to a real terminal. If interpretation or reformatting of the inbound data is required, however, it can be significantly more difficult to operate on a 3270 data stream.

An example of an application suited to the 3270 data stream interface is a passthrough program, as illustrated by the sample program “3270 data stream passthrough” on page 250. Such programs can also be used to determine the flows and screen layouts of back-end systems when you are developing FEPI applications that, for example, drive signon or menu selection sequences and manipulate screens or dialogs.

You must be fully conversant with the data stream and data stream protocols as detailed in the books in the following list, and with how the back-end system uses them:

- *3270 Data Stream Programming Reference*
- *3274 Functional Description*
- *SNA Formats*

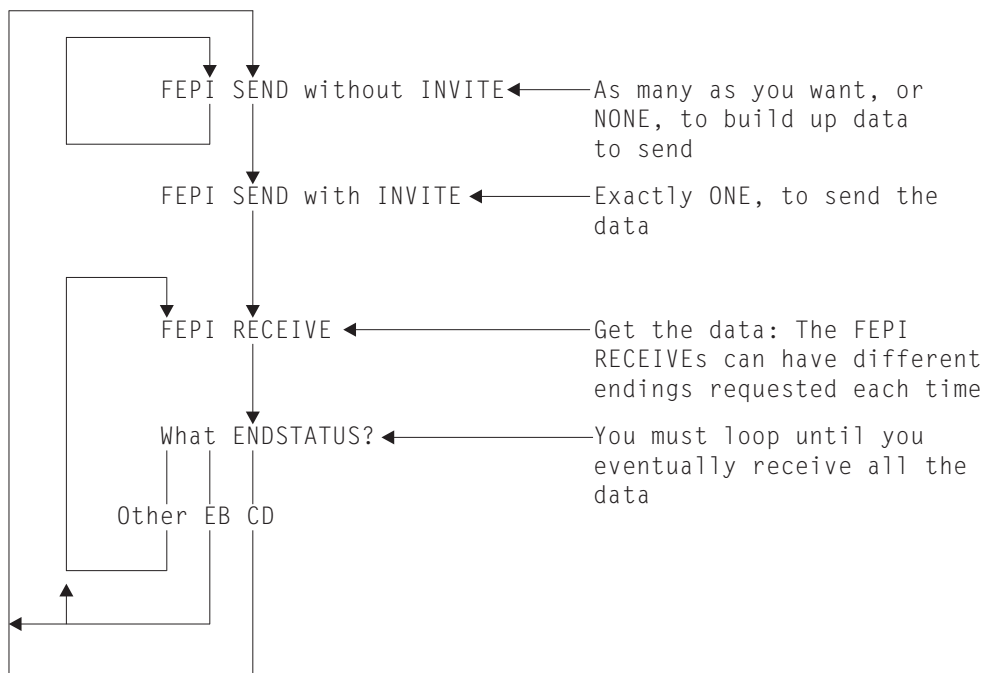
- *VTAM Programming*
- *IMS/ESA Programming Guide for Remote SNA Systems*
- *IMS/VS Version 2 Programming Guide for Remote SNA Systems.*

The application program is entirely responsible for the integrity of the data stream that uses this interface. FEPI performs no checking or interpretation on the data stream that is sent to or received from the back-end system, and makes no attempt to manipulate data into RU sizes that the sender or receiver can handle; the application program must be prepared to handle whatever data is presented to it. For example, with SLU2 mode, it must be prepared to handle READ commands, and WRITE STRUCTURED FIELD commands, in addition to the normal WRITE commands.

General sequence of commands

The following diagram illustrates the general sequence of FEPI commands that you use with data stream.

That is, a FEPI SEND, multiple FEPI RECEIVE commands that complete when all the data has been received, followed by another FEPI SEND.



Note: The diagram does not show any processing of the data, nor where you might enter, or leave, the loop. This information is explained more fully in Chapter 16, “FEPI application design,” on page 157.

Receiving

You can choose whether to process data in small segments or all at once.

Your choice depends upon various factors including:

- Processing convenience
- The amount of data that you expect

- The size of the data area that you can use
- What you are doing with the data
- How the back-end application operates
- Whether you want to handle responses (see Chapter 17, “Specialized FEPI functions,” on page 173 for this feature).

The data is a standard inbound data stream, exactly as is sent to the simulated terminal from VTAM. It is quite possible that there will be occasions on which you will receive no data; for example, when the back-end system needs to set a protocol indicator.

Command completion

The **FEPI RECEIVE** command can be specified, or defaulted, to end in one of several ways.

You can end the command using one of the following valid values:

RU On the first to occur of:

- INTO data area full
- End of request unit.

CHAIN

On the first to occur of:

- INTO data area full
- End of chain.

UNTILCDEB

On the first to occur of:

- INTO data area full
- End of chain with definite response request
- ‘Change direction’ indicated
- ‘End bracket’ indicated.

Note: Using UNTILCDEB is not recommended, because you may have the difficult task of splitting data back into its constituent chains in order to process it.

In all cases, the ENDSTATUS option is set to indicate what the completion conditions were. Where several conditions occur together, ENDSTATUS shows the most significant one. The values of ENDSTATUS and their associated meanings are shown in Table 12.

Table 12. ENDSTATUS values and associated meanings for data stream

ENDSTATUS	Command options	End bracket	Change direction	End chain	End RU	INTO area full	Next command expected
EB	RU, CHAIN, UNTILCDEB	Y	-	Y	Y	-	Any
CD	RU, CHAIN, UNTILCDEB	-	Y	Y	Y	-	FEPI SEND or CONVERSE
LIC	RU, CHAIN, UNTILCDEB	-	-	Y	Y	-	FEPI RECEIVE
RU	RU	-	-	-	Y	-	FEPI RECEIVE

Table 12. ENDSTATUS values and associated meanings for data stream (continued)

ENDSTATUS	Command options	End bracket	Change direction	End chain	End RU	INTO area full	Next command expected
MORE	RU, CHAIN, UNTILCDEB	-	-	-	-	Y	FEPI RECEIVE

Note: Y=Condition indicated.

FEPI RECEIVE commands must continue to be issued until 'change direction' or 'end bracket' is indicated. You cannot start sending data until all inbound data has been received. If an ENDSTATUS of MORE is indicated, the data stream is not necessarily self-contained and should not be processed until the remainder of the information is received. The value returned for REMFLENGTH might indicate how much more information is to come.

When 'end bracket' is received, the session is in *contention state*, and either end may try to transmit data next. Some back-end systems use 'end bracket' in the middle of a series of transmissions to allow the terminal to break in if it wants, and they may use 'end bracket' instead of 'change direction' at the end of the flow. This is particularly true of IMS. CICS usually sends 'change direction' eventually, although it may send 'end bracket' indicators intermediately.

Using your knowledge of the back-end application and system, you must check the data that you have already received, to find out whether more data is to be expected or the transmission is complete. If more data is expected, you should issue another FEPI RECEIVE command; if the transmission is complete, it is the front-end application's turn to send data.

A problem arises where the application is the passthrough type, because it does not look at the received data. There are various ways of handling this:

1. Request data conditionally from both ends—which cannot generally be done, and particularly not in the most typical case where the passthrough is directly to a front-end terminal.
2. Wait for data from both ends at once. This can be done where the passthrough is directly to a front-end terminal and the transaction is pseudoconversational for both CICS and FEPI. See "Started tasks" on page 160.
3. Ask each end at intervals if there is data waiting (for the back-end system by using FEPI RECEIVE with TIMEOUT); this is often not possible, as in the case where the passthrough is directly to a front-end terminal.
4. Forego a strict passthrough technique and check the data.
5. Assume that a transmission with 'end bracket' and no data means that more data is to come.
6. Issue another FEPI RECEIVE with TIMEOUT in case more data is to come, which has the disadvantage of introducing a delay.

Note: The last two cases involve an element of risk because the wrong assumptions can be made.

You should always use the TIMEOUT option on a FEPI RECEIVE command; see "Time-outs" on page 166.

Error handling

Apart from VTAM and back-end communication errors caused externally or, more probably, by errors in the outbound data stream that you sent previously, the most likely cause of an error condition is an incorrect sequence of commands.

That is, you have issued a FEPI RECEIVE when a FEPI SEND is expected. A FEPI RECEIVE must not follow a FEPI SEND without INVITE, or a FEPI RECEIVE that indicated 'change direction'.

Another likely error is 'previous SEND failed'. This may be an external communication error, or it may be that the back-end system has responded negatively—as IMS does, for example, if you try to run an unknown transaction. The sense data which you can get using FEPI EXTRACT CONV tells you which error it is, and in the latter case you issue another FEPI RECEIVE to get the data.

See "3270 data stream passthrough" on page 250 and "SLU P pseudoconversational" on page 252 for sample programs illustrating some of the programming techniques.

Sending

You can choose to send an entire stream of data, or you can break it up into smaller units, finishing with a FEPI SEND with INVITE.

INVITE indicates that this is the last data to send, and that inbound data should be expected next. The data is sent with 'last in chain' and 'change direction'. Otherwise, further FEPI SENDS are to be expected. It is the application program's responsibility to ensure that the amount of data sent on a request does not exceed the capacity of the receiving LU.

Error handling

Apart from VTAM errors caused, most probably, by errors in the outbound data stream that you sent previously, the most likely cause of an error condition is an incorrect sequence of commands. That is, you have issued a FEPI SEND when one was not expected. A FEPI SEND must not follow a FEPI SEND with INVITE, or a FEPI RECEIVE that did not indicate 'change direction'.

The FEPI SEND can also fail if, following 'end bracket', the back-end system sends BID to send more data and your pool has CONTENTION(LOSE). You must then receive the new back-end data first.

See "3270 data stream passthrough" on page 250 and "SLU P pseudoconversational" on page 252 for sample programs illustrating some of the programming techniques.

CONVERSE

FEPI CONVERSE can be used instead of a FEPI SEND with INVITE and the first (or only) FEPI RECEIVE. It is more efficient than issuing two separate commands and is allowed anywhere that FEPI SEND is allowed. The effects are exactly as if the two commands had been issued.

The ending conditions are identical to those for FEPI RECEIVE, unless you use the POOL option to get a temporary conversation. In this case, it ends on the first to occur of:

- INTO data area full
- 'Change direction' indicated
- 'End bracket' indicated,

and **not** at 'end of chain' alone. Further, if there is any residual data to receive, it is lost.

With regard to errors, you need to take into consideration which command is expected next:

- If the receive part of the FEPI CONVERSE command fails, the send will have already been done, and so a FEPI RECEIVE command is expected next.
- If the send part fails, the receive is not done, and, if the initial send was expected, a FEPI SEND or CONVERSE command is expected next.

SLU2 mode considerations

It is necessary, when sending outbound 3270 data streams, to ensure that a three-byte prefix containing the attention identifier (AID) and cursor address is inserted at the front of the data.

Similarly, the first two bytes of inbound data typically contain the 3270 command code and write control character (WCC). The lengths supplied or returned on the FEPI SEND, RECEIVE, or CONVERSE DATASTREAM commands include the length of the prefix.

AID values are the same as the CICS values and, in passthrough applications, can be taken from EIBAID. The cursor address however is a buffer address and cannot be taken from EIBCPOSN. 3270 buffer addresses can be 12-, 14-, or 16-bit addresses depending on the device. 12-bit addressing is the most difficult to convert to or from, but it is very common; an address conversion table and an algorithm are contained in the *3270 Information Display System 3274 Control Unit Reference Summary*.

The inbound 3270 command is most likely to be a WRITE or ERASE WRITE and is, therefore, followed by a WCC then orders and data. However, this is not guaranteed and the inbound command should be inspected to determine what it is, what, if anything, should follow it, and how it should be handled. For example, the application may choose to perform an EXEC CICS SEND TEXT from the inbound data and may, therefore, require to know whether to append the ERASE keyword. The various READ commands (such as READ BUFFER and READ MODIFIED) and all the WRITE STRUCTURED FIELD commands (a common one being READ PARTITION with QUERY) need special handling.

If you receive more than one chain (using the UNTILCDEB option), you have to find each inbound command yourself, so this is not recommended unless you know that the back-end system only sends a single chain.

For further information, refer to the *3270 Information Display System Data Stream Programmer's Reference*.

SLU P mode considerations

Two sample programs illustrate some of the programming techniques for SLU P mode.

For details, see “SLU P pseudoconversational” on page 252 and “SLU P one-out one-in” on page 251.

Chapter 16. FEPI application design

This section describes the programs comprising a FEPI application and the basic design aspects. It also discusses signon security, error handling, and system considerations, including performance.

The section contains the following topics:

- “Programs”
- “Application organization” on page 159
- “Signon security” on page 164
- “Error handling” on page 165
- “System considerations” on page 168.

Programs

A FEPI application consists of a number of programs.

These programs are as follows:

- Access
- Begin-session handler
- Unsolicited-data handler
- End-session handler.

Access program

The main purpose of an access program is to start a conversation, communicate with the back-end application, and end the conversation.

It must also be able to handle exception cases such as edit errors, transactions that are not valid, or security violations, and it might need to manage signon/signoff sequences. It might also need to handle begin-session and end-session requirements, if special handlers are not provided. The `SESSNSTATUS` option of `FEPI ALLOCATE` tells you if a new session has been started, or if you are reusing an existing session.

For many FEPI applications, particularly where formatted data is used, the access program is not complex. However, you do need to be fully conversant with everything that the back-end application might do. Your application must behave just like the real terminal and operator, and you must send and receive data in the correct sequence. Within a conversation, any data received is passed to the application that owns the conversation; FEPI cannot determine whether it is the data or screen image that was expected or, for example, a message reporting an abnormal end. Although the FEPI application needs to handle these cases, the access program need not test for all possibilities. The suggested method is to test only for the expected data or screen image and use a special error-handling program if the test fails.

Other applications may require more sophisticated programming. In some cases, you not only have to understand all the displays and protocols of the back-end

application, but must also be conversant with the detailed data stream protocols. Applications may have to be custom-written for each device and type of target that is to be supported.

Syncpoints are not needed and not applicable in FEPI because communication environments do not provide any recoverable units of work. It is up to you to provide the syncpoints and any recovery of data that you need. For particularly critical operations with the back-end applications, you may find that using “definite responses” is helpful; see “DRx responses” on page 174.

Begin-session handler

The begin-session handler transaction is started by FEPI when a connection is acquired. This transaction handles any functions that are required to initialize the session.

Typical tasks are as follows:

- Handling device queries.
- Handling any initial inbound data, or “good morning” message, following the bind.
- Signing on to the back-end system.

Device queries are sent by the back-end system (particularly CICS) if the terminal definitions so demand. You would normally reply ‘null’ (as illustrated by the begin-session sample program), or with some particular terminal properties that you want. Note, if you want to match the terminal properties to those of the real front-end terminal that an application is using, you cannot use a begin-session handler; each application will have to do its own begin-session handling.

When a back-end system sends a message after a successful bind, the connection should be in a pool where the INITIALDATA property is set to INBOUND. For SLU2, IMS always sends such a message; CICS may or may not do so depending on the way your system is defined. This extends the process of acquiring a connection to include receiving the data. Note that, if INBOUND is specified, the begin-session handler (or each application program, if there is no begin session handler) must issue a FEPI RECEIVE command to get the data and then send a suitable reply to the back-end system.

Remember that handling this initial data is just like handling any other back-end data: you must cope with whatever the back-end system may send, and handle and reply to it accordingly.

Security requirements in the back-end system might make it more appropriate for sign-on to be part of the access program. (Information about implementing signon security is in “Signon security” on page 164.)

There is a sample begin-session handler program; see “Begin session” on page 245.

Unsolicited-data handler

The unsolicited-data handler transaction is started by FEPI if inbound data arrives on a connection for which there is no current conversation.

Unsolicited data can occur when:

- A target sends more data than the application expected.

- The access program times out, or the conversation is ended, before the data arrives.
- Asynchronous IMS output such as:
 - Message from previous input that could not be processed at the time of receipt by IMS
 - Reassignment of a logical terminal that has a message queued.

With IMS, this type of unsolicited data does not usually occur in SLU2 mode because IMS only sends messages in reply to explicit requests from the terminal.

- Asynchronous CICS output such as that sent by ATI.

The unsolicited data should all be received by the handler, even if it is only to be discarded. Otherwise, although FEPI eventually discards the data, it also ends and restarts the session, which is inefficient.

There is a sample unsolicited-data handler program; see “Monitor and unsolicited data-handler” on page 243.

End-session handler

The end-session handler transaction is started by FEPI when a conversation ends or a session is to be unbound.

This could be used as follows:

- To set the session to a known state, perhaps by signing off from the back-end system, ready for the next conversation.
- When the conversation ends, to force (or prevent) unbind and the subsequent starting of a new session (overriding what the access program specified).
- To perform special action on CICS shutdown in the front-end system.

There is a sample end-session handler program; see “End-session handler” on page 250.

Note: The end-session handler transaction runs under the CICS region userid.

Application organization

This section discusses application styles, started tasks, and conversations.

The three application styles can be mixed as desired. If there are enough connections available, you can have as many conversations as you like at a time with various targets: they can be consecutive or, much more usefully, interleaved. For example, if you need data from four different applications, you could overlap the processing by sending all four requests for data before you start waiting for a response.

Application style

Your FEPI application can have one of three conversational styles.

One-out one-in conversational

One transaction performs the complete conversation with the back-end application in a single send and receive operation. This is the simplest style, if the required

data can be obtained from the back-end application in this way. The transaction can be reduced to a single FEPI CONVERSE command using a temporary conversation.

By freeing the connection between transmissions, the capacity of the connection is increased. However, this style only works where no setup is needed to run the back-end transaction and it does not depend on any prior communication. This is because, unless you have a very strict pool regime, you cannot generally guarantee which simulated terminal FEPI will use—it may not be the same one as in a previous conversation—or that you were the last user of the terminal. Further, if you receive unexpected results from the back-end transaction, you may not be able to recover. Therefore, you should only use this style where it does not matter if the back-end transaction runs or not, for example, for a simple inquiry. A one-out one-in conversational program is unlikely to be suitable for accessing CICS transactions or IMS conversational transactions.

See the sample program “SLU P one-out one-in” on page 251.

Conversational

One transaction performs the complete conversation with the back-end application using multiple send and receive operations and waiting for the inbound data to arrive. This style is used for a back-end application that requires several transmissions or complex setup. This style is simple, and if the network performance is good, the time spent waiting for inbound data may not be a problem.

See the sample program “Key stroke CONVERSE” on page 246.

Pseudoconversational

One transaction sends data to the back-end application, identifies another transaction that is to be started when the inbound data arrives, and ends. When inbound data arrives, FEPI starts the specified transaction which then receives the data. A typical technique is to have a transaction that, when started to receive inbound data, receives the data, sends the next piece of outbound data, issues FEPI START to start itself, and then ends.

The pseudoconversational style (use of FEPI START commands) results in significant CPU overheads in the front-end region. Further, since the use of FEPI START generates additional flows to and from the real terminal, response times are also significantly increased. As a consequence, FEPI START should be used sparingly when, for example, the receipt of the data from the back-end application takes a long time.

See the sample programs “Screen image SEND and START” on page 247 and “Screen image RECEIVE and EXTRACT FIELD” on page 249.

Started tasks

In the pseudoconversational case, the ‘receive’ program is started by FEPI as a CICS started task, with a start code of 'SZ' (for FEPI) which can be checked using EXEC CICS ASSIGN STARTCODE.

FEPI supplies **start data** that identifies the reason for starting the task and gives information about the FEPI resources, such as the node-target connection, the data

mode and format, and the conversation ID involved. The program that processes the transaction issues EXEC CICS RETRIEVE to get this data (the CICS rules relating to transactions and start data apply; in particular, you must retrieve all of the start data to prevent multiple initiations). Copy books DFHSZAPA, DFHSZAPO, DFHSZAPC, and DFHSZAPP contain declarations of the start data structure. You can provide your own data to be included in the start data, so that your programs can communicate with each other about their processing state and so on.

The first thing such a program must do is get ownership of the conversation using the conversation ID from the start data; it should then use FEPI RECEIVE to get the actual data from the back-end. Then it can do whatever it likes: end the conversation, send more data to the back-end system (and start itself or a new task to receive the reply), and so on.

In addition to inbound data arriving, anything else that would cause a FEPI RECEIVE command to complete causes the 'receive program' to be started. This includes a 'previous SEND failed' error, and a response from the back-end system without any data. The FEPI RECEIVE that you issue shows these cases, as if FEPI START had not been used.

The program is also started if the time limit set by the FEPI START command expires, or if the session is lost. These cases are indicated by the value of EVENTTYPE, in the start data, being TIMEOUT or SESSIONLOST rather than DATA. They should be handled as if a FEPI RECEIVE command had caused the error.

If your 'send' program is associated with a front-end terminal, your FEPI START command would normally specify that the 'receive' program uses the same terminal. You should be aware that it is not possible for FEPI to guarantee that another transaction will not use the terminal while the inbound data is awaited. In the majority of cases, this does not happen or does not matter. If it does happen and it is critical (perhaps for security reasons), you can prevent user input at the terminal by issuing an EXEC CICS SET TERMINAL command specifying NEXTTRANSID(itran) before issuing FEPI START; remember to reset NEXTTRANSID to blank in the started task. *itran* is the name of a transaction that you provide which rejects any user input, and sets NEXTTRANSID(itran) again. If this is unacceptable, you must avoid using pseudoconversational applications.

The handlers mentioned on pages "Begin-session handler" on page 158, "Unsolicited-data handler" on page 158 and "End-session handler" on page 159—begin-session, unsolicited data, end-session—are also CICS started tasks. Again, the start data (obtained with EXEC CICS RETRIEVE) tells you why the task was started and the identity of the conversation. The started task must get ownership of the conversation so that it can continue the conversation and so that FEPI knows that the event is being handled.

Conversations

Your entire communication with a particular back-end transaction should be contained in a single FEPI conversation.

This means that you remain in control of the communication; no other program can break in and you keep using the same simulated terminal. Only the task that

started the conversation with FEPI ALLOCATE can use the conversation. It “owns” it and no other task can issue any command for it, not even FEPI EXTRACT CONV.

Conversational applications

In the simplest case, an access program starts a conversation with a FEPI ALLOCATE command specifying the pool of connections that is to be used.

The command returns an identifier, the conversation ID, that is used to refer to the conversation subsequently. The program then issues a series of FEPI SEND, RECEIVE (and possibly other) commands for the conversation, each specifying the identifier, so that FEPI knows which conversation—and therefore which connection and target—the command is for. Finally, it ends the conversation with a FEPI FREE command. If it does not, the conversation is ended by FEPI when the task ends.

The FEPI FREE command should normally specify the HOLD option, so that the connection remains ready for use by another conversation. If the RELEASE option is used, or you leave the conversation to be freed by FEPI at the end of task, the session is ended, and a new one must be started for the next conversation; this is inefficient and, therefore, not recommended.

Started tasks

If the access program is pseudoconversational, after sending data it issues a FEPI START command to name the transaction that FEPI is to start when inbound data arrives.

At this point the conversation becomes “unowned” and the first task can no longer use it. However, the conversation does not end; when data arrives, the conversation ID is passed to the started task and that task issues FEPI ALLOCATE with the PASSCONVID option to get ownership of the conversation. Only then can the started task use the conversation to receive the inbound data.

While the conversation is unowned, it can be acquired by any task that knows the conversation ID. Acquiring the connection cancels the pending start request, and the task that acquired ownership has to continue the conversation as if no FEPI START had been issued. This technique is useful in a pass-through application to a front-end terminal to handle contention between inbound data and terminal input. The application issues a FEPI START command, specifying the front-end terminal, and then returns to CICS specifying a ‘next’ transaction. Inbound data arriving first causes FEPI to start the transaction on the front-end terminal, which causes CICS to cancel its wait for terminal input; if terminal input arrives first, the application, after using EXEC CICS ASSIGN STARTCODE to determine why it was started, issues FEPI ALLOCATE with PASSCONVID which cancels the FEPI START request.

Getting ownership also applies to the tasks started by the various handlers. The conversation may have been started by some access program (end-session), or by FEPI itself (begin-session, unsolicited-data). Either way, you must still issue a FEPI ALLOCATE command with PASSCONVID, quoting the conversation ID, to get ownership and continue the conversation.

When a handler has finished processing, it must tell FEPI by issuing a FEPI FREE command for the conversation. For the begin-session handler, this should specify the HOLD option to indicate that the session is ready to be used; if RELEASE is

used, the session is ended. The end-session and unsolicited-data handlers can use any of the options according to requirements.

Passing conversations

Besides using FEPI START to have a task for receiving data, any program or handler can explicitly give up ownership of its conversations so that another task can use them.

You do this with the FEPI FREE command and the PASS option. Any task can then get ownership by using FEPI ALLOCATE with PASSCONVID and, if it maintains the command sequence, continue the conversation (for example, if the first task has issued a FEPI SEND with INVITE, the second task would have to issue a FEPI RECEIVE or, perhaps, a FEPI START). It is up to the two tasks to communicate between themselves, using the standard CICS methods (TS queue, COMMAREA, and so on), about the state of the conversation and its ID. FEPI does not offer any application programming facilities for this except that the new task can use FEPI EXTRACT CONV to determine details such as the data format.

If you do not employ a method of passing and saving the conversation across invocations of a pseudoconversational front-end transaction, and instead issue the default FREE command, you lose your connection to the back-end transaction, making it possible for another program to start a conversation and effectively “break into” the active transaction. This can cause the back-end application to abnormally end.

The only other method that can be used to ensure a unique relationship between front-end and back-end transactions, is to have FEPI pools containing a single FEPI node for each user. This ensures that you always get connected to the back-end transaction on the same terminal (FEPI node) to continue your conversation. However, this method can cause administrative problems where there are a large number of end users.

Temporary conversations

In a one-out one-in conversational application you can use a single FEPI CONVERSE command that combines an ALLOCATE–SEND–RECEIVE–FREE command sequence.

This combination is selected by using the POOL option of FEPI CONVERSE rather than the CONVID option. In this case, the conversation is a *temporary* conversation that lasts only for the duration of the FEPI CONVERSE command. No conversation ID is returned by FEPI and no other commands can be issued for the conversation; you cannot even use FEPI EXTRACT FIELD to process the returned data.

As with all one-out one-in conversational applications, temporary conversations should be used with care. If more data is received than can be returned on the FEPI CONVERSE command (because, for example, the data is not what you expect), the excess is discarded and cannot be retrieved by the application. Data may be lost if the command fails and, because you cannot receive any more data or guarantee that your next conversation will use the same simulated terminal, it may be difficult to determine the state of the back-end system.

Note:

1. Every conversation started with FEPI ALLOCATE has a unique conversation ID, as does every conversation started for a handler, except in the case of end-session when started after a FEPI FREE. In this case, the ID is the same as in the task issuing the FEPI FREE.

A task started when inbound data arrives gets the same conversation ID as the task that issued the FEPI START command.

2. The state of a conversation (whether, for example, it is owned by an access program, in a begin-session handler, waiting for inbound data, or being passed) is shown by the STATE option of the CEMT INQUIRE FECONNECTION command (see State) or the FEPI INQUIRE CONNECTION command (page “FEPI INQUIRE CONNECTION” on page 96). This may be useful when you are debugging applications.
3. If your programs are written in C, do not handle conversation identifiers as strings; they may contain null characters.

Signon security

When signing on to a back-end system, FEPI applications can ask the external security manager (ESM) to supply a password substitute, or *PassTicket*.

(For an explanation of why PassTickets are necessary, see page “Signon security” on page 16.)

How to use PassTickets

This section is an overview of how PassTickets work, and describes what you need to do to use them.

About this task

Procedure

1. To process PassTickets, the ESM uses keys, known as Secure Signon keys, that are shared by the front- and back-end systems. You must define a Secure Signon key for each target system with which FEPI communicates. For information about how to do this, RACF users should refer to the *z/OS Security Server RACF System Programmer's Guide*. Users of other ESMs should refer to the documentation for their product.
2. The end-user is verified by signing on to the front-end CICS in the usual way.
3. When the end-user runs a transaction that uses FEPI, your application issues a **FEPI REQUEST PASSTICKET** command to obtain a PassTicket. If EDF is being used, the PassTicket is not displayed. A PassTicket is a secure representation of a password that can be used to sign on to the back-end system. It is valid for one use only, and is time-stamped. The userid for which the PassTicket is generated is that of the currently signed-on user. Your FEPI application can use an **EXEC CICS ASSIGN** command to check the userid of the currently signed-on user.
4. Your FEPI application uses the PassTicket and userid to perform a sign-on in the back-end system, just as if it were sending a password and userid. For example:

```
EXEC CICS FEPI SEND FORMATTED
                        CONVID(convid) FROM(CESN userid PassTicket)
                        FROMLENGTH(length_of_data)
```

It is the application's responsibility to provide the signon processing, because CICS cannot know either the type of back-end (CICS or IMS) or the back-end program being used for signon processing.

5. The back-end system uses an unchanged interface to perform the sign-on. Thus, a CICS system receiving a userid and a PassTicket can use its existing procedures to sign on the userid. RACF takes care of the fact that a PassTicket, rather than a password, is passed to it.

Results

Note: If the PassTicket times out (because, for example, of a session failure), your application should generate another and try to sign on again. If signon continues to fail and the front- and back-ends are in different MVS systems, check that the TOD clocks are suitably synchronized. Too many failed signon attempts could result in the user id being revoked.

What to do next

For detailed information about PassTickets, see the *z/OS Security Server RACF Security Administrator's Guide*.

For information about using RACF with CICS, see the *CICS RACF Security Guide*.

Related concepts

CICS RACF Security Guide

Benefits

There are a number of advantages to using PassTickets.

These advantages are as follows:

- They provide a secure way of signing on to back-end systems. This is because:
 - They are valid for one use only and are timestamped—therefore, the potential damage caused by their being intercepted is minimal.
 - Passwords are not transmitted across the network.
- FEPI applications do not have to store passwords (or ask users to reenter them) in order to sign on to back-end systems.
- No changes are required in the CICS or IMS back-end systems.
- System clocks in the front- and back-end systems do not need to be precisely synchronized (RACF compensates for variations up to plus or minus 5 minutes).

Requirements

To run a FEPI application in CICS, you must meet the following requirements.

- The front-end must be a CICS Transaction Server for z/OS region. The back-end can be a CICS region or IMS system.
- RACF Version 2 Release 1 or later, or a functionally-equivalent external security manager, on both the front- and back-end systems.
- End-users must use the same user ID in the back-end systems as in the front-end system.

Error handling

FEPI does not recover any user data when an error condition is raised. Data recovery, if needed, must be performed by the application program. In addition, the output option values on a command are not set if the command fails; your program should not be using these values in such cases.

The recommended method is errors raised by FEPI commands should be handled by your application rather than letting CICS terminate the transaction abnormally. Errors and exceptions can be detected by using the RESP and RESP2 command options, or trapped using HANDLE CONDITION.

Time-outs

You should use time-outs with FEPI commands. If there is a problem with the connection to the back-end application, a program without time-outs may wait for ever, you may stop other applications running, and operator intervention may be needed.

Time-outs can be used with FEPI ALLOCATE, RECEIVE, START, and CONVERSE commands. In all cases, the timing applies only to the period that FEPI waits for a reply from the back-end system. As soon as anything is received from the back-end, FEPI stops the timer, and then waits for as long as is necessary to receive all the data that is required to complete the command. You **cannot** specify a time-out for FEPI SEND, because the command always completes immediately, without waiting for any data to be transmitted. Any delay or other problem is handled by the following FEPI RECEIVE command. The action to take on a time-out depends on the command that was used:

- For FEPI ALLOCATE, you could retry the initial command and then retry using a different pool or target before going into your error-handling routine.
- For FEPI RECEIVE, you can retry the command and, if that fails, handle the error as if the session with the back-end application had been lost.
- For FEPI START, the time-out is reported to the started task, and not as an error on the command. In other respects, however, it is the same as a FEPI RECEIVE time-out.
- For FEPI CONVERSE with a previously allocated conversation, it is exactly as if a FEPI SEND command and then a FEPI RECEIVE command were issued. That is, the time-out that you specify applies only to the 'receive' part of the command, and is treated and handled just like that for a FEPI RECEIVE.

For a temporary conversation, it is as if the command were preceded by a FEPI ALLOCATE and followed by a FEPI FREE, so in this case the time-out is applied to both the 'allocate' and 'receive' parts of the command. In this situation, if a time-out occurs, there is no indication as to which part caused it.

Lost session

If a FEPI application loses the session with the back-end application, it should free the conversation. Having done that, the application can take whatever action is required. A typical action would be to recover any data and restore the initial state before retrying the conversation or sending a message to the user.

The loss of a session can also occur because of CLSDST(PASS) processing (as discussed in "Handling CLSDST(PASS)" on page 48). If this is the case, you can find out when the session has been reestablished using the FEPI EXTRACT CONV command. You can then continue processing as required.

Previous SEND failed

This occurs on a FEPI RECEIVE and is indicated by RESP2=216.

It may be an external communication error, or it may be that the back-end system has responded negatively (as IMS does, for example, if you try to run an unknown transaction). Use the FEPI EXTRACT CONV command to get the sense data

describing the failure. If this indicates a negative response, you should reissue the FEPI RECEIVE to get the data. If it was not a negative response, it is equivalent to a lost session and the session cannot be recovered.

Communication errors

It is simplest to treat communication and network errors as a lost session, which avoids the need for detailed SNA error protocol handling. However, sophisticated applications may want to handle certain recoverable conditions, for example, SNA CLEAR received (RESP2=230).

Bypass by user exit

A command can be rejected by the FEPI global user exits (RESP2=10). Typically this would be because it violates the rules imposed by your system programmer. Check the rules with your system programmer.

Unknown conversation ID

An unknown conversation ID is most likely to occur because the ID is specified incorrectly.

Alternatively this problem could be caused by the task that issued the command not owning the conversation, because:

- The conversation has been ended
- The conversation has been passed to another task
- FEPI ALLOCATE with PASSCONVID has not been issued.

If the error occurs on a **FEPI ALLOCATE** command with PASSCONVID, the conversation was probably not “unowned”. Where the CONVID was obtained from FEPI start data, it is possible that between FEPI scheduling the task and it starting, a resource used by the conversation has been discarded, or CICS has started shutdown.

Operator/system action

An operator/system error occurs when the operator tries to cancel a FEPI transaction. If, as is likely, it is waiting for a FEPI command to be processed, it is the ‘wait’ for FEPI processing that is canceled, not the transaction.

When a FEPI command fails with an ‘operator action’ error (RESP2=18), first end all the active conversations and then end the transaction as soon as possible.

Shutdown

A normal CICS shutdown waits for currently active tasks to end, but does not allow new tasks to start. FEPI allows existing conversations to continue within a task but does not allow them to be passed to another task (because that task would never be started), nor does it allow new conversations to be started.

Conversations that are “unowned” are ended immediately, because the tasks that would subsequently handle them would never be started. Therefore, **FEPI START** or **FREE PASS** commands issued during shutdown fail (RESP2=214); in this case the error-handling routine, after doing whatever housekeeping is required, should issue **FEPI FREE** to end the conversation. **FEPI ALLOCATE** commands issued during shutdown fail with RESP2=12.

You might need to take special action on the back-end system, for example, signing off, when the front-end application is going to shut down. For this reason, when conversations end during shutdown, the end-session handler is invoked with SHUTDOWN indicated in the EVENTVALUE field of the start data, so that the back-end system can be restored to a known state before FEPI ends; the FEPI FREE issued by the handler is treated as if RELEASE is specified. If you require this function, make sure the end-session handler is defined in the transaction list table (XLT), so that it can be started, and so that it does not adversely affect the performance of CICS shutdown. (The XLT is described in the *CICS Resource Definition Guide*.) Using an end-session handler is the only way to perform special processing on shutdown, because no notification of shutdown is given to normal active transactions and conversations.

An immediate CICS shutdown ends all conversations immediately, and commands in progress fail. No further FEPI commands can be issued, and no end-session handlers are started.

Related reference

XLT—transaction list table

System considerations

You can think of FEPI as a “pipe” through which users access back-end transactions; any peculiarities that exist in the back-end system have to be allowed for in the FEPI application. IMS has special considerations and these are explained in the following text.

This section concludes with some notes about performance.

IMS considerations

It is essential that you are familiar with using IMS and writing IMS applications.

When designing access programs that have IMS as a target back-end system, careful consideration must be given to the differences between CICS and IMS under certain circumstances:

- Message protocols
- Use of response mode
- Beginning and end of session
- Effects of IMS restart and recovery features in a FEPI environment. (Because IMS is almost totally recoverable, this can present problems in the design of the FEPI application and some event handlers.)

Message protocols

In SLU2 mode, IMS sends messages only in reply to explicit requests from the terminal.

- Unsolicited data will not usually occur; rather it will be available for the next FEPI conversation to receive. At the start of a FEPI conversation, you should first dequeue all such messages. However, unsolicited data can occur when requested data arrives after a FEPI conversation has been ended by, for example, a time-out.
- Take care if you use the IMS /SET command to preset a destination or put the transaction ID in the SPA to specify which IMS transaction to use next.
- If you are using Message Format Services (MFS), consider the following:
 - Physical paging or operator logical paging:

- Whether paged output is deleted automatically by an input message or not.
- For SLU2 mode, sending PA1 to request additional pages of paged output, and sending PA2 to remove paged output from the queue.
- Unlocking the keyboard after MFS bypass.

Response mode

You are strongly recommended to run all your back-end IMS transactions in response mode where messages to IMS from the simulated terminal are handled synchronously; each message from the terminal is processed by IMS and the reply is queued before a further message from the terminal is allowed.

Using response mode simplifies the front-end application because the data received is the reply to the data just sent and because the data stream flows from IMS are more straightforward; further, a separate FEPI conversation can be used for each IMS transaction and this allows much better use of the network. You must use the same FEPI conversation throughout an IMS conversational transaction.

If you use non-response mode, the data stream flows might be more complex. If you send multiple messages to IMS, the application has to handle asynchronous messages from IMS and, to keep the same simulated terminal, has to use the same FEPI conversation all the time.

Check with your system programmer that the transactions to be used by FEPI are defined to run in response mode. This requires the terminals for FEPI to be defined either to force response mode or to use the setting for the transaction (which in turn should be defined as response mode).

Beginning of session

For SLU2, there is always initial data.

You should:

- Dequeue all output messages by sending CLEAR and PA1 after each FEPI RECEIVE, until there are no more messages (there may be 'unsolicited' data as well as the initial data).
- If there is an IMS error message, end the session using FEPI FREE with the RELEASE option.

End of session

When you are designing an application program you must plan what happens when a session ends.

You must ensure the following:

- An IMS conversation is not left active.
- An IMS /RCLSDST command is issued if appropriate.
- An IMS MFS bypass application is not left in bypass mode.
- Any preset destination has been reset.
- Any used test mode has been ended.
- No paged output message is left on the IMS message queue.
- All messages have been received.

Physically paged messages are removed from the queue automatically when the last page has been sent and, if they are recoverable, acknowledged. Operator logically-paged messages are not removed and require a PA2 (for SLU2 mode) or a

NEXTMSG/NEXTMSGP control function (for SLU P mode) to be sent to IMS to remove the message from the queue if no input message is due.

IMS recovery

After a system failure, IMS recovers following a restart from the last checkpoint it took.

This means that, if the failure occurs when IMS has committed a message to the input queue then, on restart, IMS requeues that message and schedules a transaction to process it. Similarly, IMS will requeue all output messages that it has committed to its output queues and not successfully sent.

When IMS fails, all sessions between FEPI and IMS are ended. This is reported to the FEPI application as a command error ('session lost'). A FEPI application should check this so that it can tidy up before ending and take the appropriate action (such as informing the operator).

FEPI attempts to regain lost connections and, therefore, when IMS restarts, any previously acquired connections are reestablished. If IMS has committed an input or output message, eventually there is going to be an output message to send. With the connection reacquired, IMS attempts to recover its position and ultimately to send any queued output messages to the FEPI node that carried the original FEPI conversation. The process of recovery in this situation is different for each of the two modes:

- **SLU P recovery** When IMS tries to recover SLU P connections, it uses 'set and test sequence numbers' (STSN) in an attempt to resynchronize failed conversations. The STSN flow from IMS carries its version of the sequence numbers for the node being resynchronized. If there is an STSN handler specified, it is started. If not, FEPI responds POSITIVE, which effectively tells IMS that FEPI is satisfied with the sequence numbers sent. On receiving this, IMS sends all messages queued for the node. FEPI receives the messages, discards them and responds to IMS, completing the resynchronization.
- **SLU2 recovery** The queued message is sent by IMS until there is a request from a front-end application, that application will receive the message as unexpected data interleaved with the data that it expects to receive. This problem can be handled in either of two ways:
 1. By the application issuing a FEPI RECEIVE, with TIMEOUT, before starting its intended task or by dequeuing all output messages using CLEAR and PA1.
 2. By the begin-session handler.

This situation becomes more complex if the back-end transaction is IMS conversational, because the front-end transaction has no way of knowing this, and the IMS conversation will still be active in the back-end system awaiting input.

The potential therefore exists for a front-end FEPI application to allocate a FEPI conversation on a node where an IMS conversation still exists on the back-end system. Any data flowing on this FEPI conversation is viewed by the front-end application as an exchange with a new back-end transaction, but it is viewed by IMS as the next input message to the existing conversation. To prevent this situation occurring, you can use the begin-session handler to issue the IMS /EXIT command, which has the effect of ending an active IMS conversation.

Where the possibility exists of a number of nodes with active IMS conversations following a restart, it is possible to use FEPI to obtain a connection to IMS and control the cleanup operation, from a single point. You do this by issuing, again from the appropriate handler:

- An IMS /DISPLAY command to display all active conversations
- The IMS /EXIT command to end all those attached to FEPI nodes.

In the event of a failure that unbinds all the FEPI connections to IMS, the recovery procedure is identical to that described here.

Performance

Use the following techniques to get the best performance from your FEPI applications; the main principles are to minimize the number of commands issued and the amount of data transmitted.

Remember, however, that some of these techniques have drawbacks and some conflict with each other; you must choose the best balance to meet your needs.

- Use data area sizes that allow a send or receive to be completed with a single FEPI command.
- Use FEPI CONVERSE where possible. But remember that the send part of CONVERSE can fail for various reasons, so be sure to write your program so that it can issue a subsequent FEPI RECEIVE if necessary.
- The pseudoconversational style (use of FEPI START commands) results in significant CPU overheads in the front-end region. Further, since the use of FEPI START generates additional flows to and from the real terminal, response times are also significantly increased. As a consequence, FEPI START should be used sparingly when, for example, the receipt of the data from the back-end application takes a long time.
- Avoid ending sessions unnecessarily. Use the begin-session and end-session handlers to manage usage of the connections.
- Try to avoid operator dependency in exchanges with a back-end system.

Formatted data

- Unformatted screens (where the terminal character buffer contains no field attributes) require more processing than formatted screens. Where possible use formatted screens from the back-end systems.
- Not clearing a screen results in unnecessary data being transmitted to the back-end system.
- If, when data is received, only a small portion of the resultant screen is of interest, use FEPI EXTRACT FIELD to minimize the amount of data that needs to be transferred to the application.
- When using key stroke data, avoid issuing a FEPI CONVERSE, SEND, or RECEIVE for each attention operation; combine all the operations into one long string.
- When using key stroke data with an unformatted screen, use the HOME and ERASE-EOF keys to clear the screen rather than CLEAR, because the latter requires a network transmission.
- Use key stroke rather than screen-image data where possible, because much less data needs transferring from the application.

Chapter 17. Specialized FEPI functions

This section describes specialized control functions that are handled by FEPI but can be taken over by a FEPI application.

It contains the following topics:

- “Set and test sequence number (STSN)”
- “DRx responses” on page 174
- “SNA commands” on page 174.

Set and test sequence number (STSN)

In SLU P mode, message sequence numbers are available in the data stream to allow message resynchronization. This can be demanded by a ‘Set and Test Sequence Number’ (STSN) request when a session is started.

The response that IMS requires, and that FEPI supplies if the system programmer has not defined a transaction to handle the STSN request, depends upon whether the STSN request showed ‘SET’ or ‘TEST and SET’:

- For ‘SET’, the response is always ‘TEST POSITIVE’.
- For ‘TEST and SET’, the response is ‘TEST POSITIVE’ or ‘TEST NEGATIVE’.

Any other response to STSN will cause the session to be unbound.

If an STSN handler is defined, it is started when session resynchronization is requested by the back-end system through an SNA STSN or SDT command. The back-end system sends an SNA STSN command indicating whether the last inbound message was in doubt or not; that is, whether a message had been sent by the back-end system but it had not logged the receipt of a response. The back-end system does not send an SNA STSN command if no traffic has been on the session since the latest cold start of the back-end system, but sends an SNA SDT command directly.

Like other handlers, the STSN handler is a CICS started task that uses EXEC CICS RETRIEVE to get the start data and FEPI ALLOCATE with PASSCONVID to get ownership of the conversation identified in that data. The STSN handler, which can use the FEPI EXTRACT STSN command to determine what response is needed, must use the FEPI ISSUE command to respond to the STSN.

FEPI normally does all the necessary STSN handling automatically, so an STSN handler is required only where you need to handle the sequence number information yourself. The FEPI SEND, FEPI RECEIVE, and FEPI CONVERSE commands return the current sequence numbers for you.

A sample program illustrates the techniques used. See “STSN handler” on page 254.

DRx responses

In all cases except those mentioned in the next paragraph, FEPI automatically gives a positive DRx response when the inbound data indicates that a response is required. This response flows on the next FEPI command (SEND, RECEIVE, CONVERSE, FREE, or START).

The automatic response is not issued if the next command for a conversation is a FEPI ISSUE CONTROL or a FEPI FREE PASS. Thus, if you want to send your own response, perhaps for added certainty or confirmation of particularly sensitive changes, you would do so using FEPI ISSUE CONTROL. The response type that is required can be determined from the RESPSTATUS option of FEPI RECEIVE and FEPI CONVERSE.

You can send your own responses with either formatted data or data stream. But do not use the following because they can cause FEPI to send responses automatically:

- Key stroke formatted data containing an attention key that is not the final key stroke
- FEPI CONVERSE with the POOL option to use a temporary conversation.

If you respond negatively a back-end CICS system will discard the data but an IMS system will resend it.

SNA commands

The FEPI ISSUE command allows you to send various other SNA commands yourself. You should do this only if you have a particular requirement.

Part 4. FEPI application programming reference

This section defines the FEPI application programming commands.

(System programming commands such as INSTALL, INQUIRE, and SET are defined in Chapter 11, "FEPI system programming reference," on page 89.) The section contains the following topics:

- Chapter 18, "The FEPI API commands," on page 177
- Chapter 35, "Start data," on page 229
- Chapter 36, "Data formats," on page 231
- Chapter 37, "Ending status," on page 233.

Chapter 18. The FEPI API commands

The FEPI application programming commands are additions to the set of **EXEC CICS** commands that are available to application programmers and they have the same features and properties as those commands.

The notation used to describe these commands is the same as that used to describe all application programming commands in CICS. The FEPI application programming commands are:

- **ALLOCATE**
- **AP NOOP**
- **CONVERSE**
- **EXTRACT**
- **FREE**
- **ISSUE**
- **RECEIVE**
- **REQUEST PASSTICKET**
- **SEND**
- **START**

You must specify the 'FEPI' translator option when you use FEPI commands. FEPI commands can be issued in either 24-bit or 31-bit addressing mode, by programs that reside either above or below the 16MB line.

No information is passed through the EXEC interface block (EIB) except that, as for all CICS commands, the EIBRESP, EIBRESP2, EIBFN, and EIBRCODE fields are set.

Arguments and data types

The text used to identify arguments indicates the type of data represented by the argument and whether it is a value used by the command, or an area in which the command returns data. For example:

- **POOL(8-character data-value)** indicates that the argument is, or identifies, a string of eight characters, and that the string is passed to the command as an input value.
- **ACQNUM(fullword binary data-area)** indicates that the argument is a user-defined fullword data area in which the command can return a binary number as an output value.

Exceptionally, arguments that are lists have to be data areas, even though they are input values.

Command format

The general format of a command is:
EXEC CICS FEPI command option(argument)...

where:

command

Is the command name (for example, ADD)

option Is an option name (for example, POOL)

argument

Is the source or destination for data, as required for the specified option, that is passed to or returned from the command.

The way that you terminate the command is determined by the programming language that you use—COBOL, for example, requires an END-EXEC statement.

Errors and exception conditions

All FEPI commands support the RESP and RESP2 options to signal successful completion or an exception condition. Alternatively, you can use HANDLE CONDITION to trap errors.

Most FEPI command errors give the 'INVREQ' exception condition. The particular error in each case is uniquely identified by the RESP2 value.

If there is an error, the command does nothing and the output arguments are not changed. Note, however, that commands such as FEPI SEND may have transferred data before the condition is recognized.

Both RESP and RESP2 take, as an argument, the name of a user-defined fullword binary data area. Possible values of the RESP2 option are given in the description of each of the commands and a full list is given in "FEPI RESP2 values" on page 260. The following copy books provide declarations for the RESP2 values:

- DFHSZAPA for Assembler language
- DFHSZAPO for COBOL
- DFHSZAPP for PL/I
- DFHSZAPC for C.

The INVREQ condition and the following RESP2 values can occur for any application programming command:

RESP2

Meaning

- | | |
|----|--|
| 10 | Command bypassed by user exit. |
| 11 | FEPI not installed, or not active.. |
| 12 | CICS shutting down, command not allowed. |
| 13 | FEPI unavailable. |

- 14 FEPI busy or cannot get storage..
- 15 Unknown command..
- 16 Internal error.
- 17 FEPI cannot get storage for user exit..
- 18 Command failed through operator or system action..

Chapter 19. FEPI ALLOCATE PASSCONVID

FEPI ALLOCATE PASSCONVID acquires ownership of an existing unowned conversation.

Syntax

FEPI ALLOCATE PASSCONVID

►►—FEPI ALLOCATE—PASSCONVID—(*—data-value—*)——————►◄

Options

PASSCONVID(8-character data-value)
specifies the ID of the conversation.

Conditions

RESP2

Meaning

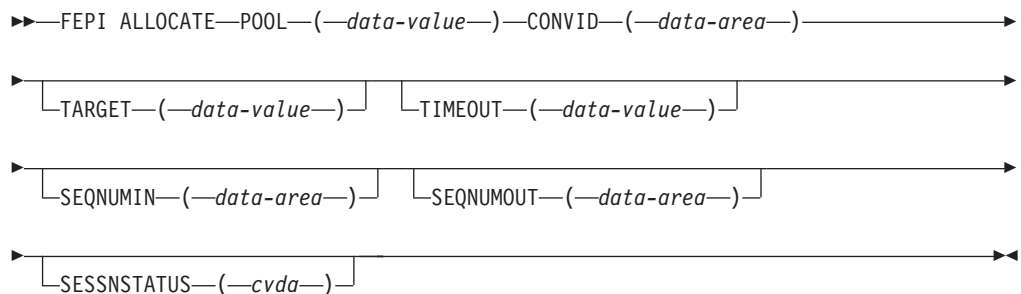
- 216 Error occurred on previous FEPI SEND.
- 240 Unknown conversation ID.

Chapter 20. FEPI ALLOCATE POOL

FEPI ALLOCATE POOL establishes a new FEPI conversation with a target application, acquiring a session from the named pool to use for the conversation. The conversation has the properties, particularly the mode (SLU2 or SLU P) and data format (data stream or formatted), specified for the pool that is used: some of the properties can be queried using FEPI EXTRACT CONV. The command completes immediately if, in the named POOL, a suitable session has been established and is not in use. Otherwise the request waits for a session to become available. A time limit can be set for this wait.

Syntax

FEPI ALLOCATE POOL



Options

CONVID(8-character data-area)

returns a unique identifier for the new conversation; this is the ID that must be quoted on all subsequent commands for the conversation.

POOL(8-character data-value)

specifies the name of the pool containing the target for the conversation.

SEQNUMIN(fullword binary data-area)

in SLU P mode, returns the current sequence number for inbound data. (SEQNUMIN has no significance in SLU2 mode.)

SEQNUMOUT(fullword binary data-area)

in SLU P mode, returns the current sequence number for outbound data. (SEQNUMOUT has no significance in SLU2 mode.)

SESSNSTATUS(cvda)

returns a value that indicates whether the session being used for the conversation was newly-bound or not. The relevant CVDA values are:

- NEWSESSION
- OLDESSION

TARGET(8-character data-value)

specifies the name of the target. TARGET can be omitted if there is only one target in the pool or if all targets are suitable for the desired conversation.

TIMEOUT(fullword binary data-value)

specifies the maximum time in seconds that the command is to wait for a

suitable session to become available. If TIMEOUT is not specified or the specified time is zero, the command is not timed out.

Conditions

If an INVREQ condition is returned, it can have the following RESP2 values:

RESP2

Meaning

30	Pool name unknown.
31	Pool name out of service.
32	Target name unknown..
33	Target name out of service..
34	Target name required but not specified.
36	No suitable session available and in service.
213	Command timed out.
241	TIMEOUT value negative or not valid.

Chapter 21. FEPI AP NOOP

FEPI AP NOOP has no effect.

Syntax

FEPI AP NOOP

▶—FEPI AP NOOP—▶

Options

None

Conditions

None specific to this command.

Chapter 22. FEPI CONVERSE DATASTREAM

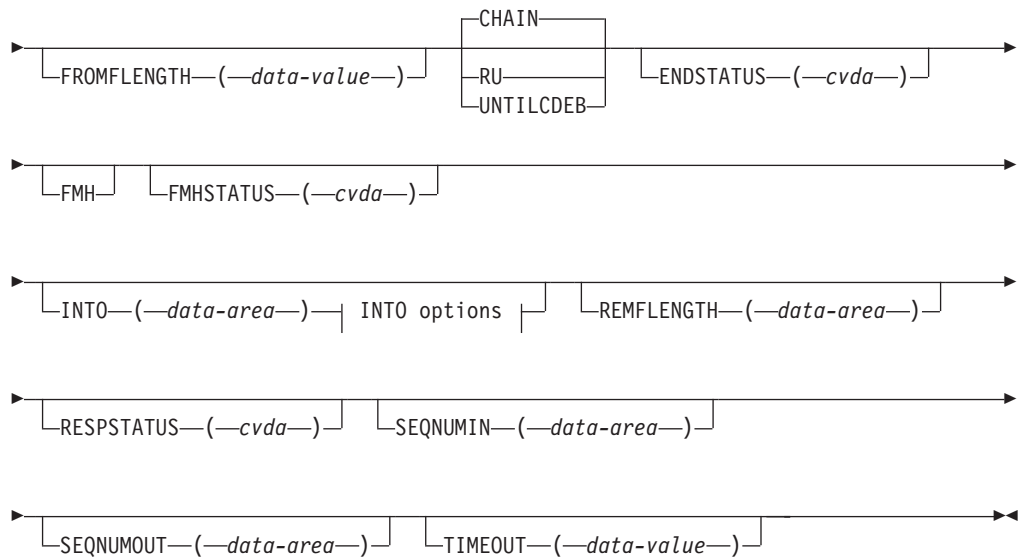
FEPI CONVERSE DATASTREAM sends application data to and receives a reply from a target.

The data supplied by the application must be a currently valid data stream appropriate to the mode of the conversation (SLU2 or SLU P); the data received into the application's data area is also data stream. Full details about the data are given in Chapter 36, "Data formats," on page 231

The conversation with the target can be one of two types. A time limit can be set for this command. For more details of ending conditions, see Chapter 37, "Ending status," on page 233.

Previously allocated conversation syntax

►►—FEPI CONVERSE DATASTREAM—CONVID—(*—data-value—*)—FROM—(*—data-value—*)—►►

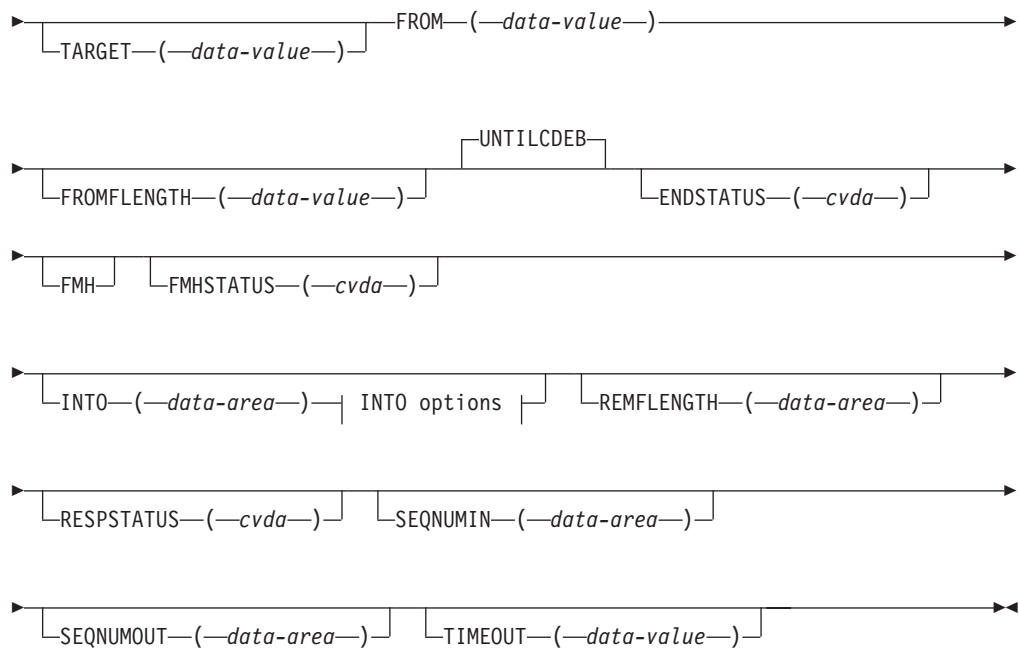


INTO options:



Temporary conversation syntax

►►—FEPI CONVERSE DATASTREAM—POOL—(*—data-value—*)—►►



INTO options:



Options

CHAIN

specifies that the command should complete when a whole chain has been received. CHAIN is not allowed if the POOL option is specified.

CONVID(8-character data-value)

specifies the ID of the conversation to use. The conversation must be owned by the task issuing the command.

ENDSTATUS(cvda)

returns a value that indicates the ending status for the received data. The relevant CVDA values are:

Value Meaning

CD 'Change direction' received.

EB 'End bracket' received.

LIC 'Last in chain' received.

RU RU received.

MORE

The data area identified by the INTO option was too small to receive all the requested data.

For more details of ending status and how additional data is handled, see Chapter 37, "Ending status," on page 233.

FMH

indicates that the data to send includes a function management header.

FMHSTATUS(cvda)

returns a value that indicates whether the received data contains a function management header. The relevant CVDA values are:

- FMH
- NOFMH

FROM(data-value)

specifies the data to send to the back-end application. Its length is specified by the FROMLENGTH option.

FROMLENGTH(fullword binary data-value)

specifies the length of the data to send; that is, the length of the data area identified by the FROM option. It must not be zero or more than the maximum length allowed for the pool.

INTO(data-area)

specifies the data area in which the received data is to be returned. The length of the area is specified by the MAXLENGTH option, and the actual length of data written into the area is returned by the TOLENGTH option.

MAXLENGTH(fullword binary data-value)

specifies the maximum amount of data that can be returned; that is, the length of the data area identified by the INTO option. It must not be more than the maximum length allowed for the pool.

POOL(8-character data-value)

specifies the name of the pool containing the target for the conversation. Specifying POOL means that the conversation is a temporary one, that exists only for the duration of the FEPI CONVERSE. The CHAIN and RU options are not allowed, and the command completes when 'change direction' or 'end bracket' is received. If there is more data to receive than fits into the data area identified by the INTO option, the additional data is discarded.

REMLENGTH(fullword binary data-area)

returns the length, if known, of data remaining after filling the data area identified by the INTO option.

RESPSTATUS(cvda)

returns a value that indicates the type of response that is required at the back-end system. The relevant CVDA values are:

Value Meaning**DEFRESP1**

Definite response 1 required.

DEFRESP2

Definite response 2 required.

DEFRESP3

Definite response 1 and definite response 2 required.

NONE

No response required.

RU specifies that the command should complete when a request unit has been received. RU is not allowed if the POOL option is specified.

SEQNUMIN(fullword binary data-area)

in SLU P mode, returns the current sequence number for inbound data, as at the completion of the command. (SEQNUMIN has no significance in SLU2 mode.)

SEQNUMOUT(fullword binary data-area)

in SLU P mode, returns the current sequence number for outbound data, as at the completion of the command. (SEQNUMOUT has no significance in SLU2 mode.)

TARGET(8-character data-value)

specifies the name of the target. TARGET can be omitted if there is only one target in the pool or if all targets are suitable for the desired conversation.

TIMEOUT(fullword binary data-value)

specifies the maximum time in seconds that the command is to wait for the requested data to begin to arrive. If TIMEOUT is not specified or the specified time is zero, the command is not timed out.

TOFLENGTH(fullword binary data-area)

returns the actual length of data received in the data area identified by the INTO option.

UNTILCDEB

specifies that the command should complete when 'change direction' or 'end bracket' is received.

Conditions

The INVREQ condition can have the following RESP2 values:

RESP2**Meaning**

- | | |
|-----|---|
| 30 | Pool name unknown. |
| 31 | Pool name out of service. |
| 32 | Target name unknown. |
| 33 | Target name out of service. |
| 34 | Target name required but not specified. |
| 35 | POOL name is unsuitable for temporary conversations. It has CONTENTION(LOSE) or it has INITIALDATA(INBOUND) and no begin-session handler. |
| 36 | No suitable session available and in service. |
| 40 | FROMLENGTH value negative, zero, or more than the maximum allowed for the current pool. |
| 50 | Inbound data with 'begin bracket' to be received. |
| 58 | VTAM SEND failed. |
| 60 | MAXFLENGTH value negative, zero, or more than the maximum allowed for the current pool. |
| 71 | VTAM RECEIVE failed. |
| 212 | Conversation has wrong data format. |
| 213 | Command timed out. |
| 215 | Session lost. |
| 216 | Error occurred on previous FEPI SEND. |
| 220 | FEPI CONVERSE not allowed at this point in the conversation. |

- 224 Only FEPI ISSUE or FEPI FREE commands allowed at this point in the conversation.
- 230 SNA CLEAR command received. For an explanation of this SNA command, see the *SNA Formats* manual.
- 231 SNA CANCEL command received. For an explanation of this SNA command, see the *SNA Formats* manual.
- 232 SNA CHASE command received. For an explanation of this SNA command, see the *SNA Formats* manual.
- 233 Exception response received.
- 234 Exception request received.
- 240 Conversation ID not owned by this task.
- 241 TIMEOUT value negative or not valid.

Chapter 23. FEPI CONVERSE FORMATTED

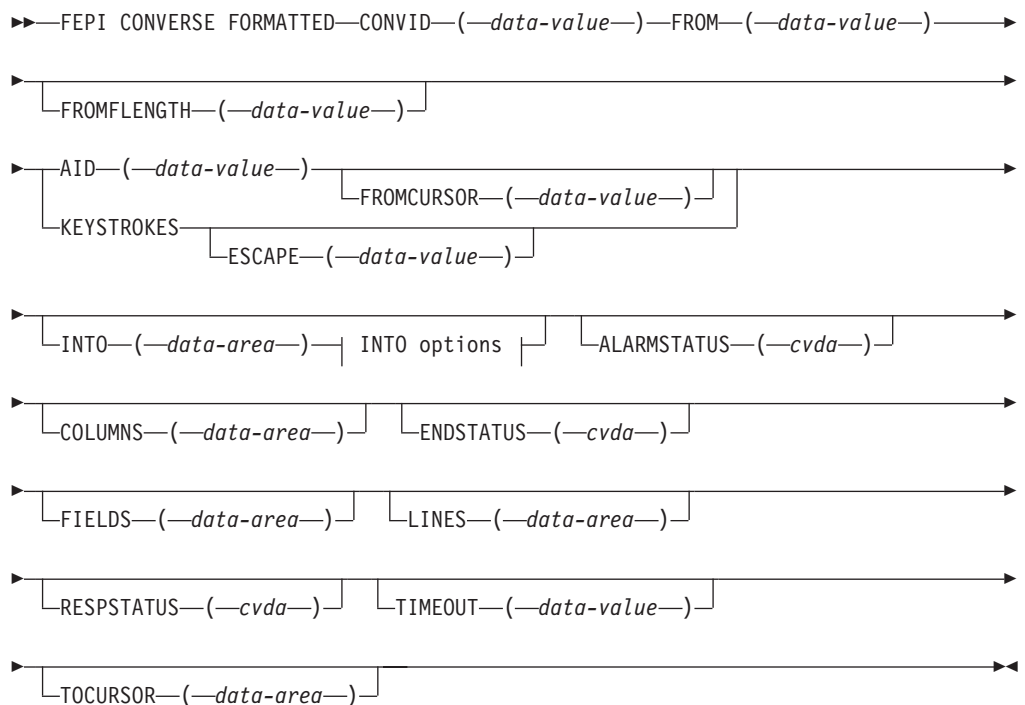
FEPI CONVERSE FORMATTED sends application data to and receives a reply from a target.

The data supplied by the application must be formatted data, as key strokes (with a final attention character) or a screen image; the data received into the application's data area is a screen image.

This command is for SLU2 mode only.

Full details about the data are given in Chapter 36, "Data formats," on page 231. The conversation with the target can be one of two types. A time limit can be set for this command. For more details of ending conditions, see Chapter 37, "Ending status," on page 233.

Previously allocated conversation syntax

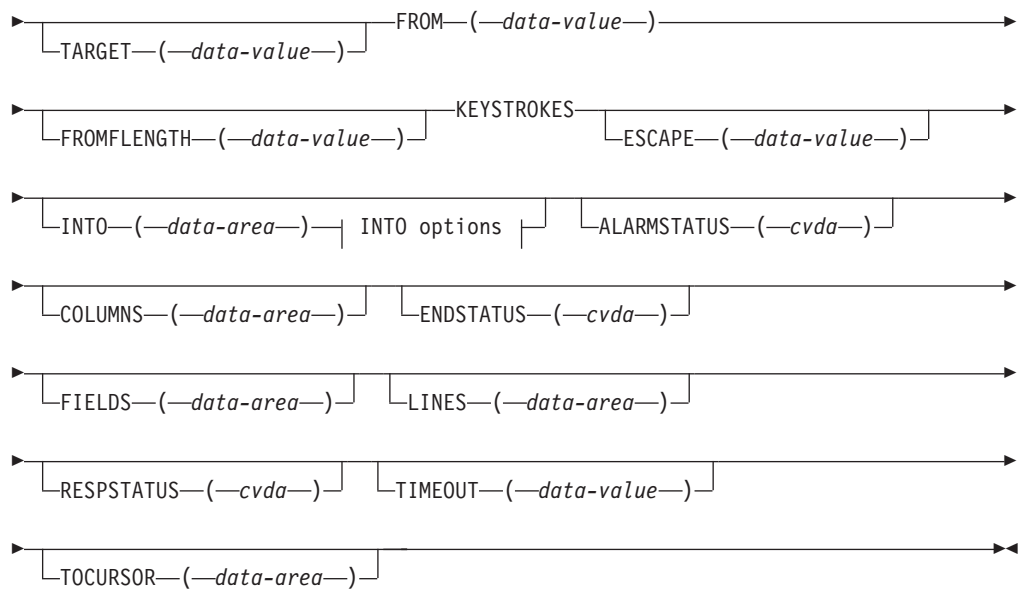


INTO options:



Temporary conversation syntax





INTO options:



Options

AID(1-character data-value)

specifies the attention identifier value to send with the data. Specifying AID also indicates that the data to send is in screen-image format, as described in Chapter 36, "Data formats," on page 231. The value must not be null (X'00'). AID, and therefore screen-image format data, is not allowed if POOL is specified.

Symbolic names for the AID values are available for the supported languages in the language-specific DFHAID copybooks.

ALARMSTATUS(cvda)

returns a value that indicates whether the received data sounded the alarm. The relevant CVDA values are:

- ALARM
- NOALARM

COLUMNS(fullword binary data-area)

returns the number of columns in the screen image.

CONVID(8-character data-value)

specifies the ID of the conversation to use. The conversation must be owned by the task issuing the command.

ENDSTATUS(cvda)

returns a value that indicates the ending status for the received data. The relevant CVDA values are:

Value Meaning

CD 'Change direction' received.

EB 'End bracket' received.

LIC 'Last in chain' received.

For more details of ending status and how additional data is handled, see Chapter 37, "Ending status," on page 233.

ESCAPE(1-character data-value)

for send data in key stroke format, specifies the escape character used to indicate character combinations representing special keys. You can use any value in the range X'40' through X'FE'. The default escape character is & (X'50').

FIELDS(fullword binary data-area)

returns the number of fields in the screen image.

FROM(data-value)

specifies the data to send to the back-end application. Its length is specified by the FROMLENGTH option. For send data in screen-image format, if the length is more than the screen image, the additional data is ignored; if it is less, the data is the first part of the screen image, and the last part of the screen image is not changed.

FROMCURSOR(fullword binary data-value)

for send data in screen-image format, specifies the position of the cursor, expressed as an offset from the start of the screen image; offset zero is the top left-hand corner of the screen. If FROMCURSOR is not specified, the cursor remains where it was positioned by the last inbound data.

FROMLENGTH(fullword binary data-value)

specifies the length of the data to send; that is, the length of the data area identified by the FROM option. It must not be zero or more than the maximum length allowed for the pool.

INTO(data-area)

specifies the data area in which the received data is to be returned. The length of the area is specified by the MAXLENGTH option, and the actual length of data written into the area is returned by the TOLENGTH option.

KEYSTROKES

specifies that the data to send is a sequence of key strokes (see Chapter 36, "Data formats," on page 231).

LINES(fullword binary data-area)

returns the number of lines in the screen image.

MAXLENGTH(fullword binary data-value)

specifies the maximum amount of data that can be returned; that is, the length of the data area identified by the INTO option. It must not be more than the maximum length allowed for the pool.

POOL(8-character data-value)

specifies the name of the pool containing the target for the conversation. Specifying POOL means that the conversation is a temporary one, that exists only for the duration of the FEPI CONVERSE. You must also specify the KEYSTROKES option. If the length of the data area identified by the INTO option is less than the size of the screen image, the additional data is discarded.

RESPSTATUS(cvda)

returns a value that indicates the type of response that is required at the back-end system. The relevant CVDA values are:

Value Meaning

DEFRESP1

Definite response 1 required.

DEFRESP2

Definite response 2 required.

DEFRESP3

Definite response 1 and definite response 2 required.

NONE

No response required.

TARGET(8-character data-value)

specifies the name of the target. TARGET can be omitted if there is only one target in the pool or if all targets are suitable for the desired conversation.

TIMEOUT(fullword binary data-value)

specifies the maximum time in seconds that the command is to wait for the requested data to begin to arrive. If TIMEOUT is not specified or the specified time is zero, the command is not timed out.

TOCURSOR(fullword binary data-area)

returns the position of the cursor in the received screen image, expressed as an offset from the start of the screen image; offset zero is the top left-hand corner of the screen.

TOLENGTH(fullword binary data-area)

returns the actual length of data received in the data area identified by the INTO option.

Note: On a FEPI CONVERSE FORMATTED command, if MAXLENGTH is less than the presentation space size, TOLENGTH returns the value defined in MAXLENGTH. If MAXLENGTH is greater than the presentation space size, TOLENGTH returns the presentation space size.

Conditions

The INVREQ condition can have the following RESP2 values:

RESP2**Meaning**

- 30 Pool name unknown.
- 31 Pool name out of service.
- 32 TARGET name unknown.
- 33 TARGET name out of service.
- 34 TARGET name required but not specified.
- 35 POOL name is unsuitable for temporary conversations. It has CONTENTION(LOSE) or it has INITIALDATA(INBOUND) and no begin-session handler.
- 36 No suitable session available and in service.
- 40 FROMLENGTH value negative, zero, or more than the maximum allowed for the current pool.
- 41 ESCAPE value not valid.
- 50 Inbound data with 'begin bracket' to be received.

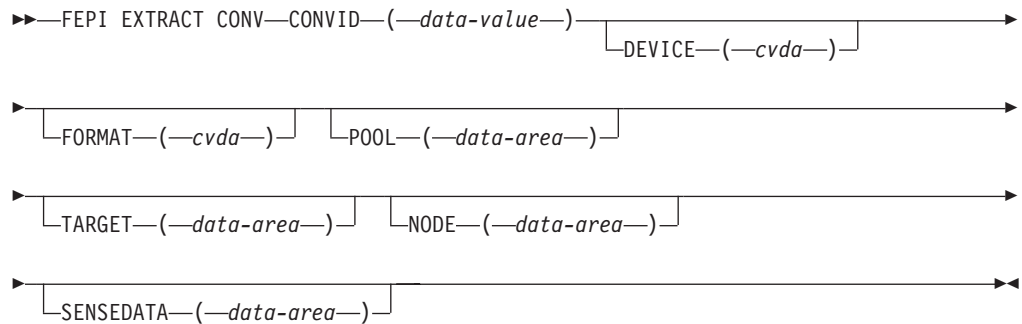
- 51 AID value not valid.
- 52 Cursor position not valid.
- 53 Character values in send data not valid.
- 54 Attribute positions or values in send data not valid.
- 55 Key stroke escape sequence in send data is not valid.
- 56 Field validation (mandatory fill, mandatory enter, trigger) failed.
- 57 Input inhibited.
- 58 VTAM SEND failed.
- 59 DBCS data rules violated.
- 60 MAXLENGTH value negative, zero, or more than the maximum allowed for the current pool.
- 71 VTAM RECEIVE failed.
- 72 RECEIVE FORMATTED processing found invalid, or unexpected data while interpreting the 3270 data stream for a WRITE, ERASE/WRITE ALTERNATE, or WRITE STRUCTURED FIELD command code.
- 210 Command not allowed for SLU P mode.
- 212 Conversation has wrong data format.
- 213 Command timed out.
- 215 Session lost.
- 216 Error occurred on previous FEPI SEND.
- 220 FEPI CONVERSE not allowed at this point in the conversation.
- 221 Data cannot be received because no AID or final attention key stroke specified.
- 224 Only FEPI ISSUE or FEPI FREE commands allowed at this point in the conversation.
- 230 SNA CLEAR command received. For an explanation of this SNA command, see the *SNA Formats* manual.
- 231 SNA CANCEL command received. For an explanation of this SNA command, see the *SNA Formats* manual.
- 232 SNA CHASE command received. For an explanation of this SNA command, see the *SNA Formats* manual.
- 233 Exception response received.
- 234 Exception request received.
- 240 Conversation ID not owned by this task.
- 241 TIMEOUT value negative or not valid.

Chapter 24. FEPI EXTRACT CONV

FEPI EXTRACT CONV gets general information about a conversation.

Syntax

FEPI EXTRACT CONV



Options

CONVID(8-character data-value)

specifies the ID of the conversation for which information is wanted. The conversation must be owned by the task issuing the command.

DEVICE(cvda)

returns a value that identifies the mode of conversation and the type of device. The relevant CVDA values are:

Value Meaning

T3278M2

SLU2 mode, 3278 Model 2

T3278M3

SLU2 mode, 3278 Model 3

T3278M4

SLU2 mode, 3278 Model 4

T3278M5

SLU2 mode, 3278 Model 5

T3279M2

SLU2 mode, 3279 Model 2B

T3279M3

SLU2 mode, 3279 Model 3B

T3279M4

SLU2 mode, 3279 Model 4B

T3279M5

SLU2 mode, 3279 Model 5B

TPS55M2

SLU2 mode, PS/55, 24 lines

TPS55M3
SLU2 mode, PS/55, 32 lines

TPS55M4
SLU2 mode, PS/55, 43 lines

LUP SLU P mode, all cases.

FORMAT(cvda)

in SLU2 mode, returns a value that identifies the data mode. The relevant CVDA values are:

- DATASTREAM
- FORMATTED

NODE(8-character data-area)

returns the node name.

POOL(8-character data-area)

returns the pool name.

SENSEDATA(fullword binary data-area)

returns the sense data associated with the last FEPI SEND, FEPI RECEIVE, or FEPI CONVERSE command for the conversation. If there is no sense data, zero is returned.

TARGET(8-character data-area)

returns the target name.

Conditions

The INVREQ condition can have the following RESP2 values:

RESP2

Meaning

215 Session lost.

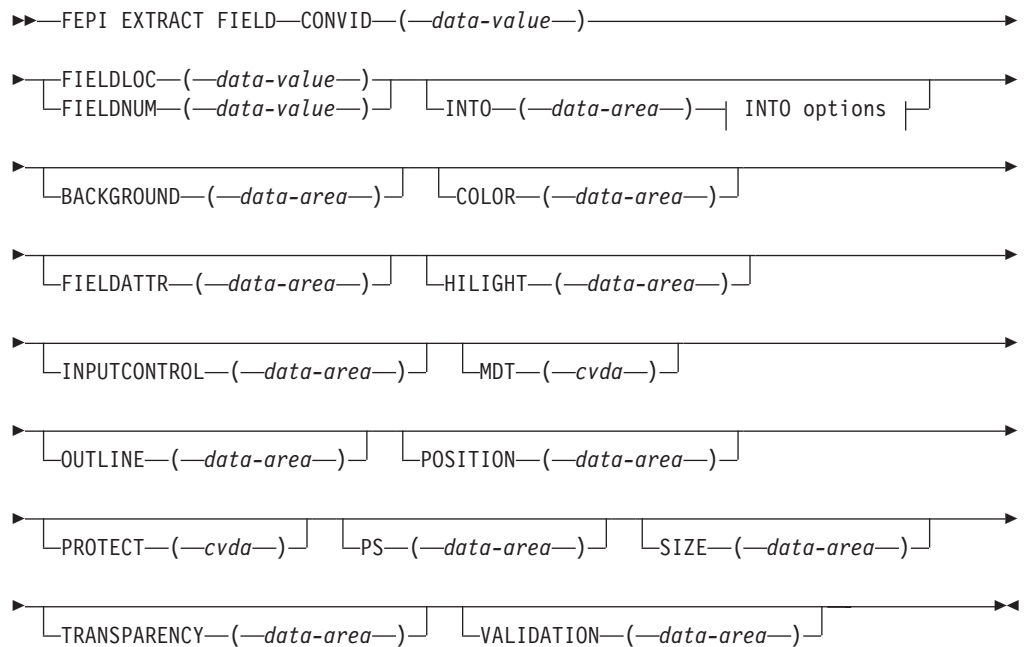
240 Conversation ID not owned by this task.

Chapter 25. FEPI EXTRACT FIELD

The command is for SLU2 mode only, and for formatted data only. FEPI EXTRACT FIELD gets information about a field in the current character buffer of the simulated terminal. It can be issued at any point in the conversation. More than one FEPI EXTRACT FIELD command can be issued for a given field. For information about field attributes and their values see *3270 Data Stream Programmer's Reference*. Symbolic names for the various attribute values are available in the DFHBMSCA copybook.

Syntax

FEPI EXTRACT FIELD



INTO options:



Options

BACKGROUND(1-character data-area)

returns the background color attribute of the field.

COLOR(1-character data-area)

returns the foreground color attribute of the field.

CONVID(8-character data-value)

specifies the ID of the conversation for which information is wanted. The conversation must be owned by the task issuing the command.

FIELDATTR(1-character data-area)

returns the 3270 field attribute of the field.

FIELDLOC(fullword binary data-value)

specifies the location of the required field expressed as an offset from the start of the screen image; offset zero is the top left-hand corner of the screen. The location can refer to any character position in the field, including its attribute byte.

FIELDNUM(fullword binary data-value)

specifies the location of the required field expressed as a field number counting from the top left-hand corner of the screen. The first field is number 1, and starts at the top-left hand corner of the screen, whether or not there is an attribute in that position. The last field ends at the bottom right-hand corner of the screen, and does not wrap back to the top.

FLENGTH(fullword binary data-area)

returns the actual length of data received in the data area identified by the INTO option.

HIGHLIGHT(1-character data-area)

returns the extended highlighting attribute of the field.

INPUTCONTROL(1-character data-area)

returns the DBCS input control attribute of the field.

INTO(data-area)

specifies the data area in which the data in the field is to be returned. The length of the area is specified by the MAXFLENGTH option, and the actual length of data written into the area is returned by the FLENGTH option.

MAXFLENGTH(fullword binary data-value)

specifies the maximum amount of data that can be returned; that is, the length of the data area identified by the INTO option. It must not be more than the maximum length allowed for the pool.

MDT(cvda)

returns a value that identifies the state of the modified data tag for the field. The relevant CVDA values are:

- NOMDT
- MDT

OUTLINE(1-character data-area)

returns the field outlining attribute of the field.

POSITION(fullword binary data-area)

returns the position of the field expressed as the offset of the first data byte from the start of the screen image; offset zero is the top left-hand corner of the screen.

PROTECT(cvda)

returns a value that indicates whether or not the field is protected. The relevant CVDA values are:

- UNPROTECTED
- PROTECTED

PS(1-character data-area)

returns the character set attribute of the field.

SIZE(fullword binary data-area)

returns the size of the field on the screen, excluding the field attribute byte, expressed as a number of bytes.

TRANSPARENCY(1-character data-area)

returns the transparency attribute of the field.

VALIDATION(1-character data-area)

returns the field validation attribute of the field.

Conditions

The INVREQ condition can have the following RESP2 values:

RESP2**Meaning**

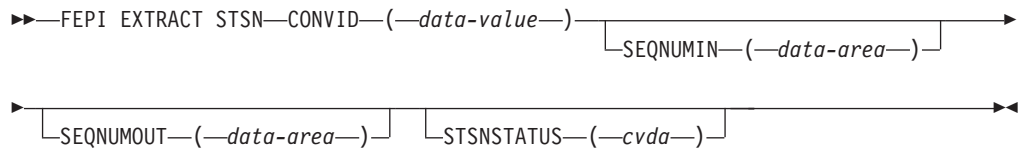
- | | |
|-----|--|
| 60 | MAXLENGTH value negative, zero, or more than the maximum allowed for the current pool. |
| 70 | FIELDLOC or FIELDNUM value negative or not valid. |
| 210 | Command not allowed for SLU P mode. |
| 212 | Conversation has wrong data format. |
| 224 | Only FEPI ISSUE or FEPI FREE commands allowed at this point in the conversation. |
| 240 | Conversation ID not owned by this task. |

Chapter 26. FEPI EXTRACT STSN

The command is for SLU P mode only. FEPI EXTRACT STSN gets sequence number status information for a conversation.

Syntax

FEPI EXTRACT STSN



Options

CONVID(8-character data-value)

specifies the ID of the conversation for which information is wanted. The conversation must be owned by the task issuing the command.

SEQNUMIN(fullword binary data-area)

returns the current sequence number for inbound data.

SEQNUMOUT(fullword binary data-area)

returns the current sequence number for outbound data.

STSNSTATUS(cvda)

returns the current sequence-number set and test status. The relevant CVDA values are:

Value Meaning

NOSTSN

No 'set' or 'test and set' issued.

STSNSET

'Set' sequence number issued.

STSNTEST

'Test and set' sequence number issued.

Conditions

The INVREQ condition can have the following RESP2 values:

RESP2

Meaning

211 Command not allowed for SLU2 mode.

240 Conversation ID not owned by this task.

Chapter 27. FEPI FREE

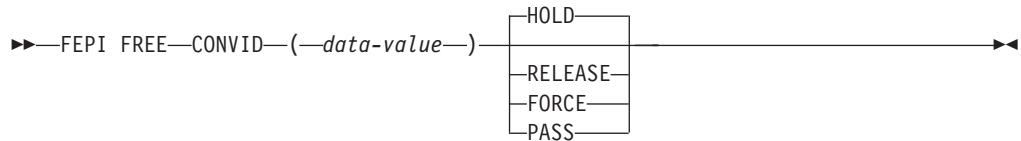
FEPI FREE ends a task's use and ownership of a conversation. The conversation may be ended completely, or may be passed to another task.

The action depends on the processing state of the conversation:

- Begin session handler
- STSN handler
- Access program
- End session handler
- Unsolicited-data handler.

Syntax

FEPI FREE



Options

CONVID(8-character data-value)

specifies the ID of the conversation to free. The conversation must be owned by the task issuing the command.

FORCE

tells FEPI what action to take. For all processing states of the conversation, FORCE instructs FEPI to end the conversation unconditionally, and to take the connection that it was using out of service immediately and, if possible, reset it.

HOLD

tells FEPI what action to take.

For the access program and the unsolicited-data handler, HOLD instructs FEPI to end the conversation and to retain the session for use by another conversation. However, this is subject to any end-session processing.

For the begin-session handler and the STSN handler, HOLD tells FEPI that begin-session or STSN processing has ended, and that the conversation is ready for the next processing state.

For the end-session handler, HOLD tells FEPI that end-session processing has ended, and instructs FEPI to end the conversation and to retain the session for use by another conversation. (If CICS shutdown is in progress, HOLD is the same as RELEASE.)

PASS

tells FEPI what action to take. For all the processing states of the conversation, PASS specifies that the task is relinquishing ownership of the conversation so that another task can acquire it. There is no change in the processing state of the conversation. (PASS is not allowed if CICS shutdown is in progress.)

RELEASE

tells FEPI what action to take.

For the access program and the unsolicited-data handler, RELEASE instructs FEPI to end the conversation, and to release and unbind the session that it was using, thereby forcing a new session to be started next time the connection is used. However, this is subject to any end-session processing.

For the begin-session handler and the STSN handler, RELEASE tells FEPI that begin-session or STSN processing has ended, and instructs FEPI to end the conversation without proceeding to the next processing state, and to release and unbind the session that it was using, thereby forcing a new session to be started next time the connection is used. However, this is subject to any end-session processing.

For the end-session handler, RELEASE tells FEPI that end-session processing has ended, and instructs FEPI to end the conversation, and to release and unbind the session that it was using, thereby forcing a new session to be started next time the connection is used.

Note that, under normal circumstances, after a FEPI FREE RELEASE command has been issued the session does not remain in RELEASED state, because FEPI automatically tries to reacquire the session. However, if a FEPI SET CONNECTION ACQSTATUS(RELEASED) command is issued before the FREE RELEASE, the session remains in RELEASED state.

Conditions

The INVREQ condition can have the following RESP2 values:

RESP2

Meaning

- | | |
|------------|---|
| 214 | CICS shutting down, conversation should be ended. |
| 240 | Conversation ID not owned by this task. |

Chapter 28. FEPI ISSUE

FEPI ISSUE sends control data, such as standard responses and sense data, to the target system. The command completes as soon as the corresponding VTAM SEND has been accepted.

Syntax

FEPI ISSUE

```
►► FEPI ISSUE—CONVID—(—data-value—)—CONTROL—(—cvda—)—————►
|
| SENSEDATA—(—data-value—) | VALUE—(—cvda—) |
|-----|-----|
```

Options

CONTROL (cvda)

specifies what type of control data to send. The relevant CVDA values depend upon the data type and the mode of the conversation:

For all modes:

Value Meaning

NORMALRESP

Send a normal response, as specified by the VALUE option.

EXCEPTRESP

Send an exception response, as specified by the VALUE option, and with the sense data specified by the SENSEDATA option.

ATTENTION

Send an attention (SNA 'signal' command X'00010000').

LUSTAT

Send an SNA 'LUSTAT' command with the sense data specified by the SENSEDATA option.

For data stream only:

Value Meaning

CANCEL

Send an SNA 'cancel' command.

For SLU P mode only:

Value Meaning

STSN Send an SNA 'set and test sequence number' command.

RTR Send an SNA 'ready to receive' command.

CONVID(8-character data-value)

specifies the ID of the conversation to use. The conversation must be owned by the task issuing the command.

SENSEDATA(fullword binary data-value)

specifies sense data to send to the target when the CONTROL is LUSTAT or EXCEPTRESP.

VALUE(cvda)

specifies the response type associated with the control data. The relevant CVDA values are determined by what is specified for the CONTROL option:

For EXCEPTRESP and NORMALRESP:**Value Meaning****DEFRESP1OR2**

Send definite response 1 or 2 as required.

DEFRESP1

Send definite response 1.

DEFRESP2

Send definite response 2.

DEFRESP3

Send definite response 1 and definite response 2.

For STSN:**Value Meaning****POSITIVE**

Send STSN positive response.

NEGATIVE

Send STSN negative response.

INVALID

Send STSN response not valid (this unbinds the session).

RESET

Send STSN reset response (this unbinds the session).

DEFRESP2

Send definite response 2.

DEFRESP3

Send definite response 1 and definite response 2.

For other controls:

None; the VALUE option is not used with the other controls.

Conditions

The INVREQ condition can have the following RESP2 values:

RESP2**Meaning**

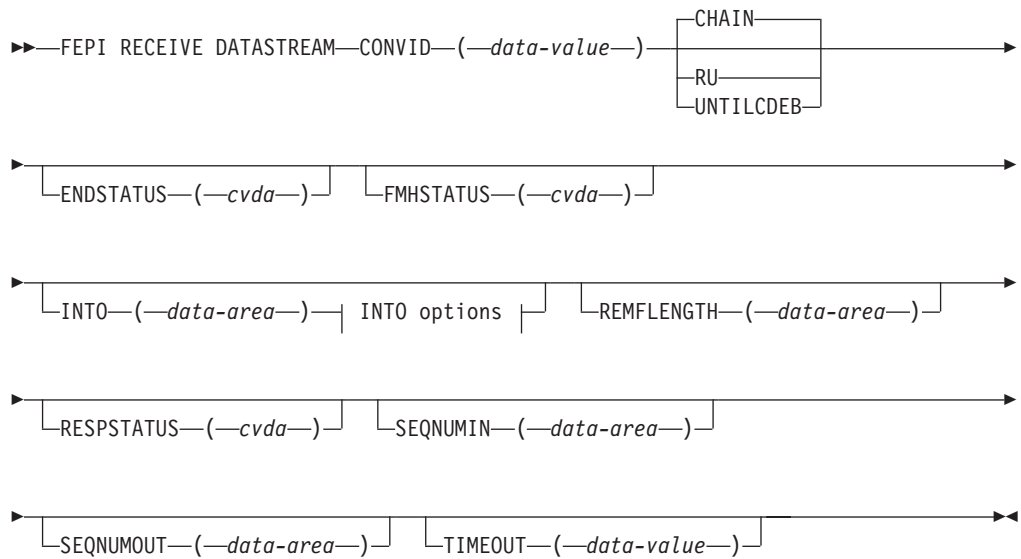
- 80 CONTROL value not valid.
- 81 VALUE value not valid: omitted when required, specified when not required, or unsuitable for the specified CONTROL value.
- 82 SENSEDATA value omitted when required or specified when not required.
- 90 Definite response type did not match what was required.
- 91 Only NORMALRESP or EXCEPTRESP are allowed at this point in the conversation.
- 92 Response to STSN SET was not positive.
- 93 Only FEPI ISSUE CONTROL(STSN) allowed at this point in the conversation.
- 94 Only FEPI ISSUE CONTROL(STSN) or FEPI ISSUE CONTROL(NORMALRESP) allowed at this point in the conversation.
- 95 CONTROL value not allowed at this point in the conversation.
- 211 Option not allowed for SLU2 mode.
- 215 Session lost.
- 216 Error occurred on previous FEPI SEND.
- 230 SNA CLEAR command received.
- 231 SNA CANCEL command received.
- 232 SNA CHASE command received.
- 233 Exception response received.
- 234 Exception request received.
- 240 Conversation ID not owned by this task.

Chapter 29. FEPI RECEIVE DATASTREAM

FEPI RECEIVE DATASTREAM receives data from a target and places the received data stream into the application's data area. Full details about the data are given in Chapter 36, "Data formats," on page 231. By default, FEPI RECEIVE DATASTREAM completes when a whole chain of data has been received. A time limit can be set for this command. For more details of ending conditions, see Chapter 37, "Ending status," on page 233.

Syntax

FEPI RECEIVE DATASTREAM



INTO options:



Options

CHAIN

specifies that the command should complete when a whole chain has been received.

CONVID(8-character data-value)

specifies the ID of the conversation to use. The conversation must be owned by the task issuing the command.

ENDSTATUS(cvda)

returns a value that indicates the ending status for the received data. The relevant CVDA values are:

Value Meaning

CD 'Change direction' received.

EB 'End bracket' received.

LIC 'Last in chain' received.

RU RU received.

MORE

The data area identified by the INTO option was too small to receive all the requested data.

For more details of ending status and how additional data is handled, see Chapter 37, "Ending status," on page 233.

FLENGTH(fullword binary data-area)

returns the actual length of data received in the data area identified by the INTO option.

FMHSTATUS(cvda)

returns a value that indicates whether the received data contains a function management header. The relevant CVDA values are:

- FMH
- NOFMH

INTO(data-area)

specifies the data area in which the received data is to be returned. The length of the area is specified by the MAXFLENGTH option, and the actual length of data written into the area is returned by the FLENGTH option.

MAXFLENGTH(fullword binary data-value)

specifies the maximum amount of data that can be returned; that is, the length of the data area identified by the INTO option. It must not be more than the maximum length allowed for the pool.

REMFLENGTH(fullword binary data-area)

returns the length, if known, of data remaining after filling the data area identified by the INTO option.

RESPSTATUS(cvda)

returns a value that indicates the type of response that is required at the back-end system. The relevant CVDA values are:

Value Meaning

DEFRESP1

Definite response 1 required.

DEFRESP2

Definite response 2 required.

DEFRESP3

Definite response 1 and definite response 2 required.

NONE

No response required.

RU specifies that the command should complete when a request unit has been received.

SEQNUMIN(fullword binary data-area)

in SLU P mode, returns the current sequence number for inbound data, as at the completion of the command. (SEQNUMIN has no significance in SLU2 mode.)

SEQNUMOUT(fullword binary data-area)

in SLU P mode, returns the current sequence number for outbound data, as at the completion of the command. (SEQNUMOUT has no significance in SLU2 mode.)

TIMEOUT(fullword binary data-value)

specifies the maximum time in seconds that the command is to wait for the requested data to begin to arrive. If TIMEOUT is not specified or the specified time is zero, the command is not timed out.

UNTILCDEB

specifies that the command should complete when 'change direction' or 'end bracket' is received.

Conditions

The INVREQ condition can have the following RESP2 values:

RESP2**Meaning**

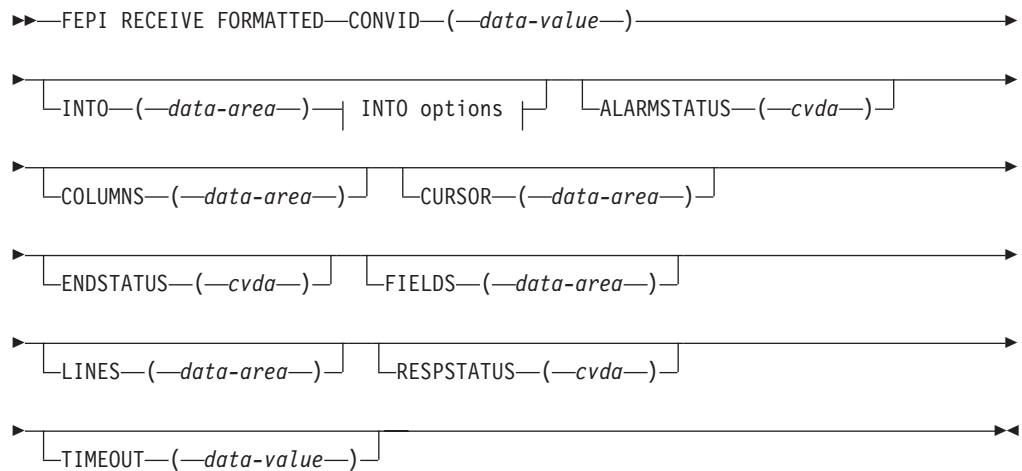
- | | |
|-----|--|
| 60 | MAXFLENGTH value negative or more than maximum allowed for the current pool. |
| 71 | VTAM RECEIVE failed. |
| 212 | Conversation has wrong data format. |
| 215 | Session lost. |
| 216 | Error occurred on previous FEPI SEND. |
| 221 | FEPI RECEIVE not allowed at this point in the conversation. |
| 224 | Only FEPI ISSUE or FEPI FREE commands allowed at this point in the conversation. |
| 230 | SNA CLEAR command received. |
| 231 | SNA CANCEL command received. |
| 232 | SNA CHASE command received. |
| 233 | Exception response received. |
| 234 | Exception request received. |
| 240 | Conversation ID not owned by this task. |
| 241 | TIMEOUT value negative or not valid. |

Chapter 30. FEPI RECEIVE FORMATTED

This command is for SLU2 mode only. FEPI RECEIVE FORMATTED receives data from a target. The data received into the application's data area is a screen image. Full details about the data are given in Chapter 36, "Data formats," on page 231. FEPI RECEIVE FORMATTED completes after receiving the inbound data with 'last in chain', 'end bracket' or 'change direction' indicated. A time limit can be set for this command. For more details of ending conditions, see Chapter 37, "Ending status," on page 233.

Syntax

FEPI RECEIVE FORMATTED



INTO options:



Options

ALARMSTATUS(cvda)

returns a value that indicates whether the received data sounded the alarm. The relevant CVDA values are:

- ALARM
- NOALARM

COLUMNS(fullword binary data-area)

returns the number of columns in the screen image.

CONVID(8-character data-value)

specifies the ID of the conversation to use. The conversation must be owned by the task issuing the command.

CURSOR(fullword binary data-area)

returns the position of the cursor in the received screen image, expressed as an offset from the start of the screen image; offset zero is the top left-hand corner of the screen.

ENDSTATUS(cvda)

returns a value that indicates the ending status for the received data. The relevant CVDA values are:

Value Meaning

CD 'Change direction' received.

EB 'End bracket' received.

LIC 'Last in chain' received.

For more details of ending status and how additional data is handled, see Chapter 37, "Ending status," on page 233.

FIELDS(fullword binary data-area)

returns the number of fields in the screen image.

FLENGTH(fullword binary data-area)

returns the actual length of data received in the data area identified by the INTO option.

INTO(data-area)

specifies the data area in which the received data is to be returned. The length of the area is specified by the MAXFLENGTH option, and the actual length of data written into the area is returned by the FLENGTH option.

LINES(fullword binary data-area)

returns the number of lines in the screen image.

MAXFLENGTH(fullword binary data-value)

specifies the maximum amount of data that can be returned; that is, the length of the data area identified by the INTO option. It must not be more than the maximum length allowed for the pool.

RESPSTATUS(cvda)

returns a value that indicates the type of response that is required at the back-end system. The relevant CVDA values are:

Value Meaning**DEFRESP1**

Definite response 1 required.

DEFRESP2

Definite response 2 required.

DEFRESP3

Definite response 1 and definite response 2 required.

NONE

No response required.

TIMEOUT(fullword binary data-value)

specifies the maximum time in seconds that the command is to wait for the requested data to begin to arrive. If TIMEOUT is not specified or the specified time is zero, the command is not timed out.

Conditions

The INVREQ condition can have the following RESP2 values:

RESP2

Meaning

- 60 MAXLENGTH value negative or more than maximum allowed for the current pool.
- 71 VTAM RECEIVE failed.
- 72 RECEIVE FORMATTED processing found invalid, or unexpected data while interpreting the 3270 data stream for a WRITE, ERASE/WRITE, ERASE/WRITE ALTERNATE, or WRITE STRUCTURED FIELD command code.
- 210 Command not allowed for SLU P mode.
- 212 Conversation has wrong data format.
- 213 Command timed out.
- 215 Session lost.
- 216 Error occurred on previous FEPI SEND.
- 221 FEPI RECEIVE not allowed at this point in the conversation.
- 224 Only FEPI ISSUE or FEPI FREE commands allowed at this point in the conversation.
- 230 SNA CLEAR command received.
- 231 SNA CANCEL command received.
- 232 SNA CHASE command received.
- 233 Exception response received.
- 234 Exception request received.
- 240 Conversation ID not owned by this task.
- 241 TIMEOUT value negative or not valid.

Chapter 31. FEPI REQUEST PASSTICKET

FEPI REQUEST PASSTICKET requests an external security manager (ESM) such as RACF to build a *PassTicket*. The PassTicket is a password substitute that your application can use to sign on to the back-end system associated with the conversation. For an explanation of how to use PassTickets to make your FEPI applications more secure, see “How to use PassTickets” on page 164.

Syntax

FEPI REQUEST PASSTICKET

```
▶▶ FEPI REQUEST PASSTICKET—(—data-area—)—CONVID—(—data-value—)—▶▶
└── ESMRESP—(—data-area—)—┘ └── ESMREASON—(—data-area—)—┘
```

Options

CONVID(8-character data-value)

specifies the ID of the conversation with the back-end system for which a PassTicket is required.

ESMREASON(fullword binary data-area)

returns the reason code from the ESM.

ESMRESP(fullword binary data-area)

returns the response code from the ESM. For an explanation of the response and reason codes returned by RACF, see *Return codes from the secured signon session key generator service* in the *z/OS V1R11.0 Security Server RACF Macros and Interfaces* manual.

PASSTICKET(8-character data-area)

returns the PassTicket generated by the ESM.

Conditions

The INVREQ condition can have the following RESP2 values:

RESP2

Meaning

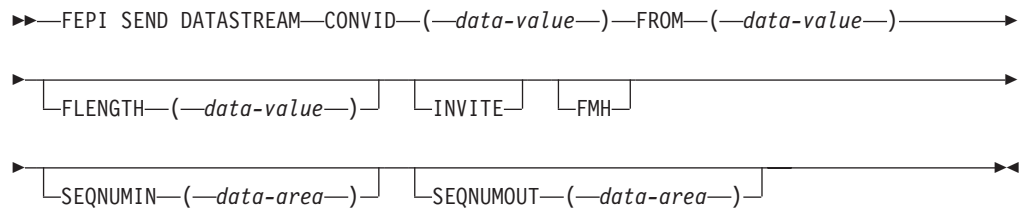
- | | |
|-----|---|
| 240 | Conversation ID not owned by this task. |
| 250 | Passticket not built successfully. |
| 251 | CICS ESM interface not initialized. |
| 252 | Unknown return code in ESMRESP from the ESM. |
| 253 | Unrecognized response from CICS security modules. |
| 254 | Function unavailable. |

Chapter 32. FEPI SEND DATASTREAM

FEPI SEND DATASTREAM sends application data to a target. The data supplied by the application must be currently valid data stream appropriate to the mode of the conversation (SLU2 or SLU P). Full details about the data are given in Chapter 36, "Data formats," on page 231. The command completes as soon as the (first) VTAM SEND has been accepted.

Syntax

FEPI SEND DATASTREAM



Options

CONVID(8-character data-value)

specifies the ID of the conversation to use. The conversation must be owned by the task issuing the command.

FMH

indicates that the data to send includes a function management header.

FLENGTH(fullword binary data-value)

specifies the length of the data to send; that is, the length of the data area identified by the FROM option. It must not be zero or more than the maximum length allowed for the pool.

FROM(data-value)

specifies the data to send to the back-end application. Its length is specified by the FLENGTH option.

INVITE

requests FEPI to send 'last in chain' and 'change direction' at the end of the data. This indicates that the data is complete, and that inbound data is expected next.

SEQNUMIN(fullword binary data-area)

in SLU P mode, returns the current sequence number for inbound data, as at the completion of the command. (SEQNUMIN has no significance in SLU2 mode.)

SEQNUMOUT(fullword binary data-area)

in SLU P mode, returns the current sequence number for outbound data, as at the completion of the command. (SEQNUMOUT has no significance in SLU2 mode.)

Conditions

The INVREQ condition can have the following RESP2 values:

RESP2

Meaning

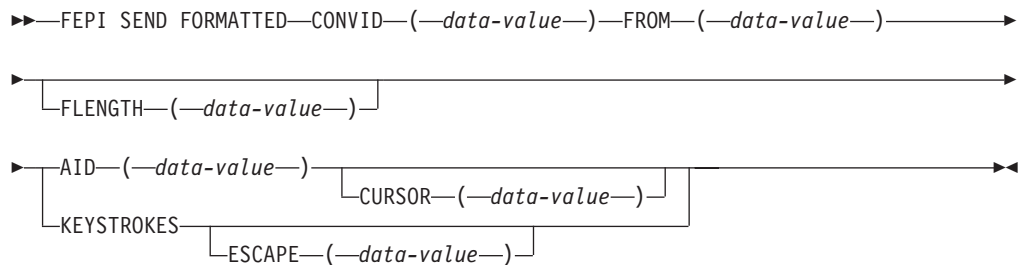
- 40 FLENGTH value negative or more than maximum allowed for the current pool.
- 50 Inbound data with 'begin bracket' to be received.
- 58 VTAM SEND failed.
- 212 Conversation has wrong data format.
- 215 Session lost.
- 216 Error occurred on previous FEPI SEND.
- 220 FEPI SEND not allowed at this point in the conversation.
- 224 Only FEPI ISSUE or FEPI FREE commands allowed at this point in the conversation.
- 230 SNA CLEAR command received.
- 231 SNA CANCEL command received.
- 232 SNA CHASE command received.
- 233 Exception response received.
- 234 Exception request received.
- 240 Conversation ID not owned by this task.

Chapter 33. FEPI SEND FORMATTED

This command is for SLU2 mode only. FEPI SEND FORMATTED sends application data to a target. The data supplied by the application must be formatted data, as key strokes or as a screen image. Full details about the data are given in Chapter 36, "Data formats," on page 231. The command completes as soon as the (first) VTAM SEND has been accepted.

Syntax

FEPI SEND FORMATTED



Options

AID(1-character data-value)

specifies the attention identifier value to send with the data. Specifying AID also indicates that the data to send is in screen-image format, as described in Chapter 36, "Data formats," on page 231. A value of null (X'00') may be specified to indicate that no attention is to be sent, and that a further FEPI SEND is to follow.

Symbolic names for the AID values are available for the supported languages in the language-specific DFHAID copybooks.

CONVID(8-character data-value)

specifies the ID of the conversation to use. The conversation must be owned by the task issuing the command.

CURSOR(fullword binary data-value)

for send data in screen-image format, specifies the position of the cursor, expressed as an offset from the start of the screen image; offset zero is the top left-hand corner of the screen. If CURSOR is not specified, the cursor remains where it was positioned by the last inbound data.

ESCAPE(1-character data-value)

for send data in key stroke format, specifies the escape character used to indicate character combinations representing special keys. You can use any value in the range X'40' through X'FE'. The default escape character is & (X'50').

FLENGTH(fullword binary data-value)

specifies the length of the data to send; that is, the length of the data area identified by the FROM option. It must not be zero or more than the maximum length allowed for the pool.

FROM(data-value)

specifies the data to send to the back-end application. Its length is specified by

the FLENGTH option. For send data in screen-image format, if the length is more than the screen image, the additional data is ignored; if it is less, the data is the first part of the screen image, and the last part of the screen image is not changed.

KEYSTROKES

specifies that the data to send is in key stroke format, a sequence of key strokes, as described in Chapter 36, "Data formats," on page 231.

Conditions

The INVREQ condition can have the following RESP2 values:

RESP2

Meaning

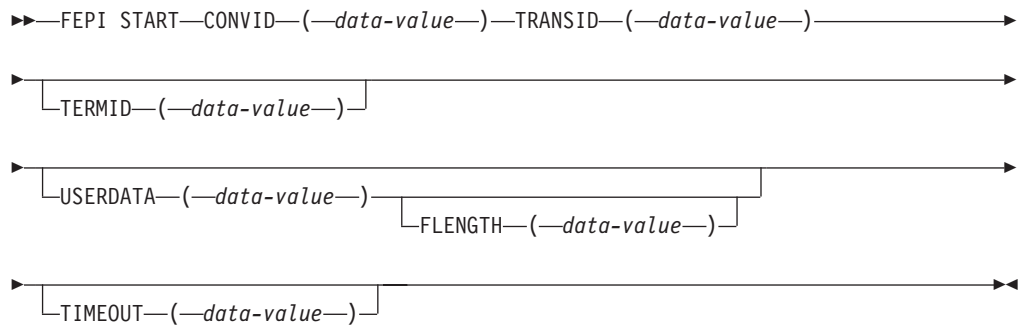
- 40 FLENGTH value negative or more than maximum allowed for the current pool.
- 41 ESCAPE value not valid.
- 50 Inbound data with 'begin bracket' to be received.
- 51 AID value not valid.
- 52 Cursor position not valid.
- 53 Character values in send data not valid.
- 54 Attribute positions or values in send data not valid.
- 55 Key stroke escape sequence in send data not valid.
- 56 Field validation (mandatory fill, mandatory error, trigger) failed.
- 57 Input inhibited.
- 58 VTAM SEND failed.
- 59 DBCS data rules violated.
- 210 Command not allowed for SLU P mode.
- 212 Conversation has wrong data format.
- 215 Session lost.
- 220 FEPI SEND not allowed at this point in the conversation.
- 224 Only FEPI ISSUE or FEPI FREE commands allowed at this point in the conversation.
- 230 SNA CLEAR command received.
- 231 SNA CANCEL command received.
- 232 SNA CHASE command received.
- 233 Exception response received.
- 234 Exception request received.
- 240 Conversation ID not owned by this task.

Chapter 34. FEPI START

FEPI START is used to relinquish control of a conversation and to specify a new transaction to be started when the next inbound data arrives. Up to 128 characters of user data can be passed to the transaction as part of the start data, as described in Chapter 35, “Start data,” on page 229 below.

Syntax

FEPI START



Options

CONVID(8-character data-value)

specifies the ID of the conversation to suspend. The conversation must be owned by the task issuing the command.

FLENGTH(fullword binary data-value)

specifies the length of the optional user data to pass to the transaction that is started; that is, the length of the data area identified by the USERDATA option. The FLENGTH value must not be greater than 128.

TERMID(4-character data-value)

specifies the name of the terminal, if any, to be associated with the transaction that is started.

TIMEOUT(fullword binary data-value)

specifies the maximum time in seconds that FEPI is to wait for inbound data to begin to arrive before starting the transaction. If TIMEOUT is not specified or the specified time is zero, the command is not timed out.

TRANSID(4-character data-value)

specifies the name of the transaction that is to be started when the next inbound data arrives.

USERDATA(data-value)

specifies optional user data to pass to the transaction that is started, in addition to control information passed by FEPI. Its length is specified by the FLENGTH option.

Conditions

The INVREQ condition can have the following RESP2 values:

RESP2**Meaning**

- 61 FLENGTH value negative or too large.
- 62 TRANSID name not valid.
- 63 TERMID name not valid.
- 214 CICS shutting down, conversation should be ended.
- 215 Session lost.
- 216 Error occurred on previous FEPI SEND.
- 223 FEPI START not allowed at this point in the conversation.
- 224 Only FEPI ISSUE or FEPI FREE commands allowed at this point in the conversation.
- 230 SNA CLEAR command received.
- 231 SNA CANCEL command received.
- 232 SNA CHASE command received.
- 233 Exception response received.
- 234 Exception request received.
- 240 Conversation ID not owned by this task.
- 241 TIMEOUT value negative or not valid.

Chapter 35. Start data

For various events, FEPI invokes a transaction, as a CICS started task, to handle the event. This might be in response to FEPI START, or to handle STSN, begin-session, end-session, or unsolicited-data.

The transactions have a start code of 'SZ', as can be determined with the **EXEC CICS ASSIGN** command. FEPI provides start data which describes the event, and the conversation which is to be used to handle it. All of this data must be retrieved by the transaction using EXEC CICS RETRIEVE. The transaction can then gain access to the conversation identified in the data by using FEPI ALLOCATE PASSCONVID.

The structure for start data is shown below; the copy books DFHSZAPA, DFHSZAPO, DFHSZAPC, and DFHSZAPP (according to your programming language) provide declarations for this structure.

DATATYPE	Fullword binary data-area
EVENTTYPE	CVDA
EVENTVALUE	CVDA
EVENTDATA	8-character data-area
spare	4-character data-area
POOL	8-character data-area
TARGET	8-character data-area
NODE	8-character data-area
CONVID	8-character data-area
DEVICE	CVDA
FORMAT	CVDA
spare	8-character data-area
FLENGTH	Fullword binary data-area
USERDATA	128-character data area.

Fields

CONVID(8-character data-area)

the ID of the conversation for which the event occurred (this is the CONVID that should be used in FEPI ALLOCATE PASSCONVID).

DATATYPE(fullword binary data-area)

Type and structure of data. Value is 1 for FEPI start data.

DEVICE(cvda)

the device type of conversation for which the event occurred, values being as for FEPI EXTRACT CONV.

EVENTDATA(8-character data-area)

always nulls.

EVENTTYPE(cvda)

Indicates why the transaction was started. Values are:

Value	Event
BEGINSESSION	Begin-session to be handled
DATA	Inbound data arrived, following a FEPI START command

Value	Event
FREE	End-session transaction started to handle end of conversation as a result of a FEPI FREE request
SESSIONLOST	Active session lost while waiting for inbound data to arrive following a FEPI START command
STSN	Set and test sequence number (STSN) to be handled
TIMEOUT	Timed out waiting for inbound data to arrive following a FEPI START command
UNSOLDATA	Inbound data arrived outside a conversation.

EVENTVALUE(cvda)

A CVDA giving further information about event types FREE and RELEASE.

Values for FREE:

FORCE	A FEPI FREE FORCE command was issued.
HOLD	A FEPI FREE HOLD command was issued.
RELEASE	A FEPI FREE RELEASE command was issued.
SHUTDOWN	CICS is shutting down.
TASK	Conversation being freed by end-of-task.

The EVENTVALUE value is zero for all other event types.

FLENGTH(fullword binary data-area)

the length of the data in USERDATA.

FORMAT(cvda)

the data format of conversation for which the event occurred, values being as for FEPI EXTRACT CONV.

NODE(8-character data-area)

the name of the node for which the event occurred.

POOL(8-character data-area)

the name of the pool for which the event occurred.

TARGET(8-character data-area)

the name of the target for which the event occurred.

USERDATA(128-character data-area)

user data as specified on the FEPI START command.

spare

nulls.

Chapter 36. Data formats

The format of outbound and inbound are described here.

Outbound data

Data stream

The data is a standard outbound data stream, exactly as would be sent from the simulated terminal to VTAM.

Screen-image format, SLU2 mode

The data replaces, byte for byte, the data in the character buffer of the simulated terminal. Any data value is allowed. Data that goes into positions within a protected field must be identical to that in the field; data for positions occupied by an attribute byte is ignored. MDTs can be set forcibly for fields by setting the value in the attribute position to X'01'. (FEPI will set MDT automatically if data has changed.)

Key stroke format, SLU2 mode

The data can contain any combination of data characters together with manipulative, special, and attention key values. Data characters are represented by their EBCDIC code values in the range X'40'–X'FE', or by their DBCS code values of pairs of bytes in the range X'41'–X'FE', plus X'4040'. Manipulative, special, and attention key values are represented by escape sequences, comprising the escape character specified by the ESCAPE option and a 2-character code. Using '&' for the escape character, the escape sequences are:

Manipulative keys

- &HO** home
- &Ln** cursor left, n times
- &Rn** cursor right, n times
- &Un** cursor up, n times
- &Dn** cursor down, n times
- &Tn** tab, n times
- &Bn** backtab, n times
- &Nn** newline, n times (where n = 1–9)

Special keys

- &IN** insert
- &DL** delete
- &RS** reset
- &EF** erase to end of field
- &EI** erase input
- &FM** field mark
- &DU** DUP
- &ES** escape character
- &MS** start secure MSR

&SO shift out

&SI shift in

Attention keys

&AT attention

&An PAn (n = 1–3)

&nn PFnn (where nn = 01–24, leading 0 must be specified)

&CL clear

&CS cursor select (light pen)

&EN enter

&ME end secure MSR

Keys not listed and data characters below X'40' are not supported. Thus, nulls (X'00') are excluded—nulls can be generated by use of the erase or delete keys. Key strokes following an attempt to enter into a protected field are ignored until 'reset' is keyed.

For magnetic stripe reader support, the sequence **&MS...data...&ME** represents passing a secure magnetic stripe card through the reader. Nonsecure cards have to be simulated by using the corresponding key strokes.

Zero, one, or more than one, attention keys may be used. If an attention key is followed by data characters, FEPI does an implicit receive operation for each one until the back-end application unlocks the keyboard and sends 'change direction' or 'end bracket' (and FEPI responds positively to any definite response requests); then the subsequent key strokes are sent.

Inbound data

Data stream

The data is a standard inbound data stream, exactly as would be sent to the simulated terminal from VTAM. Note that the received data is not complete if the command that received the data returned an ENDSTATUS of MORE.

Formatted, SLU2 mode

The data is the contents of the simulated terminal character buffer that FEPI holds. Data characters are represented by their EBCDIC or DBCS code values; positions corresponding to field attributes contain X'FF'.

Chapter 37. Ending status

This information describes in detail the conditions under which FEPI CONVERSE and FEPI RECEIVE commands complete and how the completion condition is reported to the application.

The completion conditions for each command are:

FEPI CONVERSE DATASTREAM using a temporary conversation

On the first to occur of:

- INTO data area full
- 'change direction' indicated
- 'end bracket' indicated.

It does not end at 'end of chain' alone; if a definite response request is indicated on a chain, FEPI responds positively and continues receiving data.

FEPI CONVERSE DATASTREAM using a previously allocated conversation

As for FEPI RECEIVE DATASTREAM.

FEPI CONVERSE FORMATTED using a temporary conversation

on the first to occur of:

- 'change direction' indicated
- 'end bracket' indicated.

It does not end at 'end of chain' alone; if a definite response request is indicated on a chain, FEPI responds positively and continues receiving data.

FEPI CONVERSE FORMATTED using a previously allocated conversation

As for FEPI RECEIVE FORMATTED.

FEPI RECEIVE DATASTREAM

This can be specified or defaulted to end in one of the following ways:

RU on the first to occur of:

- INTO data area full
- end of request unit.

CHAIN

on the first to occur of:

- INTO data area full
- 'end of chain'.

UNTILCDEB

on the first to occur of:

- INTO data area full
- 'end of chain' with definite response request
- 'change direction' indicated
- 'end bracket' indicated.

FEPI RECEIVE FORMATTED

At end of chain.

In all cases, ENDSTATUS is set to indicate the completion conditions and RESPSTATUS is set to indicate whether a response is required and, if so, the type of response. Where several conditions occur together, ENDSTATUS shows the most significant. The values and their meanings are shown in Table 13.

Table 13. ENDSTATUS values and associated meanings

ENDSTATUS	Commands						Conditions					Next command expected (except after CONVERSE with POOL)
	RECEIVE		CONVERSE without POOL		CONVERSE with POOL		End bracket	Change direction	End chain	End RU	INTO area full	
	DS	FM	DS	FM	DS	FM						
EB	X	X	X	X	X	X	Y	-	Y	Y	-	Any
CD	X	X	X	X	X	X	-	Y	Y	Y	-	FEPI SEND or CONVERSE
LIC	X	X	X	X	-	-	-	-	Y	Y	-	FEPI RECEIVE
RU	R	-	R	-	-	-	-	-	-	Y	-	FEPI RECEIVE
MORE	X	-	X	-	X	-	-	-	-	-	Y	FEPI RECEIVE
Note: <ul style="list-style-type: none"> • DS=Datastream • FM=Formatted • X=Possible with command • R=Possible with RU option of command • Y=Condition indicated. 												

Part 5. Appendixes

Appendix A. FEPI sample programs

The SDFHSAMP library contains a set of sample programs (in source form), including two back-end application programs, that show you how to set up and use FEPI.

Although the samples are copyrighted, you can use and copy them freely for educational purposes to help you write FEPI applications. This appendix gives an overview of these programs. It contains the following topics:

- “List of samples”
- “VS COBOL II Sample Restrictions” on page 239
- “Installing the samples” on page 239
- “Using the samples” on page 240
- “Description of the samples” on page 242.

List of samples

A subset of the sample programs is available in each of the supported programming languages.

The programs and their names are given in Table 14.

Table 14. Sample programs and their names

Description	Transaction name	COBOL	Assembler	PL/I	C
Programs:					
Setup	CZXS	DFH0VZXS	DFH0AZXS		DFH0CZXS
Monitor and unsolicited data handler	CZUX	DFH0VZUX			
Begin-session handler	CZUC	DFH0VZUC			
3270 data stream pass-through	CZTD	DFH0VZTD	DFH0AZTD		
Key stroke CONVERSE	CZTK	DFH0VZTK		DFH0PZTK	DFH0CZTK
Screen image SEND and START	CZTS	DFH0VZTS			
Screen image RECEIVE and EXTRACT	CZTR	DFH0VZTR			
End-session handler	CZUU	DFH0VZUU			
SLU P, one-out, one-in	CZPS	DFH0VZPS	DFH0AZPS		
SLU P, pseudoconversational	CZPA	DFH0VZPA	DFH0AZPA		
STSN handler	CZQS	DFH0VZQS	DFH0AZQS		
Back-end CICS	CZBC		DFH0AZBC		
Back-end IMS	CZBI		DFH0AZBI		
Copy books:					
Customization data		DFH0BZCO	DFH0BZCA	DFH0BZCP	DFH0BZCC
Messages and other text		DFH0BZMO	DFH0BZMA	DFH0BZMP	DFH0BZMC
Key stroke map		DFH0BZ1O		DFH0BZ7P	DFH0BZ6C
Send/receive map		DFH0BZ2O			
Back-end CICS map			DFH0BZ3A		
SLU P, one-out, one-in map		DFH0BZ4O	DFH0BZ8A		

Table 14. Sample programs and their names (continued)

Description	Transaction name	COBOL	Assembler	PL/I	C
SLU P, pseudoconversational map		DFH0BZ5O	DFH0BZ9A		
Maps:					
Key stroke		DFH0MZ1		DFH0MZ7	DFH0MZ6
Send/receive		DFH0MZ2			
SLU P, one-out, one-in		DFH0MZ4	DFH0MZ8		
SLU P, pseudoconversational		DFH0MZ5	DFH0MZ9		
Back-end CICS			DFH0MZ3		

There are also some sample resource definitions. Sample definitions for front-end and back-end CICS regions are in the RDO groups DFH\$0AZ, DFH\$0BZ, DFH\$0CZ, DFH\$0BZ, DFH\$0PZ, and DFH\$0VZ. A sample definition for a back-end IMS region is in DFH0IZRI. A sample definition of a CICS TD queue, DFH0IZRQ, is in the DFHDCTG RDO group.

Table 15 shows you which samples illustrate which functions.

Table 15. Functional cross-reference for sample programs

Functions	Samples (Last two letters of sample program name. See notes.)											
	TD	TK	TS	TR	PA	PS	QS	UC	UU	UX	XS	
—												
SLU2	X	X	X	X	X	X	X					
SLU P	X	X	X	X	X	X						
Data stream	X	X	X	X	X	X						
Screen-image	X	X	X	X								
Key stroke	X	X										
ALLOCATE	X	X	X	X								
ALLOCATE with PASSCONVID	X	X	X	X	X	X	X					
EXTRACT STSN	X											
EXTRACT FIELD	X	X										
SEND	X	X										
START	X	X										
RECEIVE	X	X	X	X								
CONVERSE	X	X	X									
CONVERSE with POOL	X											
ISSUE	X											
FREE	X	X	X	X	X	X	X	X				
FREE with PASS	X	X										
INSTALL	X											
ADD	X											
Start data	X	X	X	X	X							
TD queue data	X											
One-out one-in	X											
Conversational	X	X										

Table 15. Functional cross-reference for sample programs (continued)

Functions	Samples (Last two letters of sample program name. See notes.)											
	TD	TK	TS	TR	PA	PS	QS	UC	UU	UX	XS	
Pseudo- conversational	X	X	X									
Assembler language	X	X	X	X	X							
COBOL	X	X	X	X	X	X	X	X	X	X	X	X
C	X	X										
PL/I	X											
Notes:												
TD Data stream												
TK Key stroke												
TS Screen image send/start												
TR Screen image receive												
PA SLU P pseudoconversational												
PS SLU P one-out, one-in												
QS STSN												
UC Begin session												
UU End session												
UX Monitor, unsolicited data												
XS Setup												
FEPI EXTRACT CONV, SET/INQUIRE/browse, and DELETE/DISCARD commands are not illustrated in the sample programs.												

VS COBOL II Sample Restrictions

The following COBOL samples can only be compiled using the Release 3, and later, versions of the VS COBOL II compiler:

- DFH0VZUC
- DFH0VZUX
- DFH0VZPS
- DFH0VZPA

Installing the samples

To get the FEPI samples running, you need to customize them for your system.

Before you begin

About this task

Procedure

1. You need to change the following samples:
 - The customization data copy book, DFH0BZCx
 - The setup program, DFH0xZXS
 - The resource definitions, DFH0IZRx.

2. Compile or assemble and link-edit all the samples (and their maps) that you want, as you would for any CICS application program.
3. Define the samples to your front-end system, using the sample resource definitions listed in "List of samples" on page 237. The resource definitions are in the form required as input to the DFHCSDUP utility. Note that there is a separate resource group for each language because the transaction names used are the same for each programming language. You should have defined the necessary transient data (TD) queues when you installed FEPI itself. Sample definitions are provided in group, DFHDCTG.
4. Assemble, link-edit, define, and install the appropriate back-end program and maps on your back-end system. If you want to use the IMS back-end samples:
 - a. Use the sample resource definitions in DFH0IZRI.
 - b. Link-edit the back-end program with the IMS version of ASMTDLI (or the appropriate language module), and specify RMODE and AMODE as 24. If you use the CICS version of ASMTDLI, the program will abend when executed in the IMS environment.

Using the samples

The samples form an integrated set. The setup program provides the FEPI resource definitions that the other samples use. The monitor and the various handlers support and complement the access programs, to form a complete FEPI communication package, just as you need to provide.

Remember, however, that these are samples designed for illustration purposes. Although they give a great deal of help, and include suggestions about writing FEPI programs, for any particular circumstance you must consider exactly what your requirements are.

The two back-end programs, one for CICS and one for IMS, provide applications for the front-end programs to access. The back-end CICS program is for access by the front-end SLU2 mode programs, and the back-end IMS program is for access by the front-end SLU P mode programs; no SLU2 mode access to IMS is provided. Although the back-end programs are supplied in source form, it is not necessary for you to understand the internal logic - only the external operations, as is the case for a "real" existing back-end application.

The FEPI sample front-end and back-end transactions assume that the datastream sent from the back-end application is received unaltered by the front-end application. For example, FEPI samples may perform unexpectedly if the datastreams are compressed after having been sent from the back-end application.

The back-end CICS program

This program is the CICS back-end application used by the FEPI sample programs.

Module name	DFH0AZBC
Transaction name	CZBC
Abend code	USZA
Map name	DFH0MZ3

On the first invocation of the transaction, a map is sent to the terminal.

When there is input from the terminal, CICS invokes the transaction again. The customer data for the customer number from the input is found and sent to the terminal, and further input is awaited. PF3 or CLEAR ends the transaction.

Certain customer numbers cause special processing such as abends and delays, to show how a front-end application could manage such events. The valid customer numbers are:

- 0001-0005 Normal
- 0006 Delayed response
- 0007 Abend before send
- 0008 Abend after send.

```

CZBC                Customer Inquiry
Please type a customer number in the range 1 to 9999, then Enter.
Customer Number . . . . .
Name . . . . . :
Balance . . . . . :
Address . . . . . :
Last Transaction Date . . :
F3=EXIT to CICS
  
```

Figure 10. CZBC transaction: customer inquiry

Program logic

```

Main procedure:
  Set up exception condition handling:
    Map error - SEND_NEW_MAP
    CLEAR/PF3 - END_PROG
  Test COMMAREA
  If transaction not previously invoked
    Call SEND_NEW_MAP
  RECEIVE map
  If customer number not valid
    SEND message
    RETURN
  If customer type is 'ABEND before MAP'
    ABEND
  Build map with customer data
  If customer type is 'LONG DELAY'
    DELAY
  SEND map
  If customer type is 'ABEND after MAP'
    ABEND
  RETURN
SEND_NEW_MAP routine:
  SEND new map
  RETURN
END_PROG routine:
  Clear terminal
  RETURN
  
```

The back-end IMS program

This program is the IMS back-end application used by the FEPI sample programs.

Module name	DFH0AZBI
Transaction name	CZBI

This is a simple IMS back-end response mode program that is driven by input from a front-end FEPI application. It modifies the time stamp in the input message and returns the message to the front-end application.

IMS schedules this transaction when an input message is queued for it. It addresses the I/O PCB, DLI call function, and I/O area to build the parameter list for the GU call to retrieve the queued input message.

The time field of the input message is updated and the program then issues an ISRT call to place the message on the output queue. IMS then sends the output message to the front-end FEPI application.

Output messages from this program are all prefixed with a 5-byte function management header.

If any errors occur, the program ends with a nonzero return code.

Program logic

```

GETMAIN storage areas for reentrancy
Address PCB
Issue GU call to get input message
Use TIME to obtain system time
Update I/O area
Issue ISRT call to send output message
RETURN

```

Description of the samples

Each sample program performs a specific function and is associated with a CICS transaction.

Setup program

This program installs the resources—property sets, nodes, targets, and pools—that are used by the FEPI sample programs.

Module names	DFH0VZXS, DFH0AZXS, DFH0CZXS
Transaction name	CZXS

The definitions of each of these resources are organized so that they can easily be changed. They are kept separate from the processing that does the installation, and there is no hard-coding of values in the CICS commands. There are four main tables, holding details of each resource type. This enables the resources to be changed by repeating sets of definitions which are in an easy-to-understand form. If desired, the program could be changed to obtain the resource definitions from a file.

The resources defined are:

Pool	Property set	Nodes	Targets
P00L1	PROPSET1	NODE1 NODE2 NODE3 NODE4 NODE5	TARGET1
P00L2	PROPSET2	NODE6 NODE7 NODE8 NODE9 NODE10	TARGET1
P00L3	PROPSET3	NODE1 NODE2 NODE3 NODE4 NODE5	TARGET2

You must customize these definitions to match the requirements of your system. If you do, you may also need to change the definitions in the sample customization constants copy book DFH0BZCx. You do not need to change any other samples—you need recompile them.

Each table is processed in turn. Nodes and targets are organized into lists for reasons of efficiency. Details of resource installation are written to the CICS log automatically by FEPI.

On completion, a message is sent. The setup program would typically be started by a PLT program, in which case the message goes to the CICS log. It can, however, be invoked from a terminal and, in this case, the message is sent to the terminal.

For clarity, error checking is minimal. In particular, the FEPI INSTALL commands do not check errors at all, because FEPI reports any errors that occur to the FEPI transient data queue, and they are then recorded by the sample monitor program.

Program logic

```

For each property set in table
  FEPI INSTALL PROPERTYSET
For each node in table
  Add node to list
FEPI INSTALL NODELIST
For each target in table
  Add target to list
FEPI INSTALL TARGETLIST
For each pool in table
  Start new lists of nodes and targets
  For each entry within pool definition
    If node, add details to node list
    If target, add details to target list
  FEPI INSTALL POOL with NODELIST and TARGETLIST
Send completion message
RETURN

```

Monitor and unsolicited data-handler

This program monitors unexpected events and handles unsolicited data for the FEPI sample programs.

Module name	DFH0VZUX
Transaction name	CZUX
TS queue name	MONITOR

This transaction handles:

- Unexpected events that are reported by FEPI to a TD queue, which triggers this transaction
- Unsolicited data from a back-end system, for which FEPI starts this transaction.

Because the event descriptions provided by FEPI and the processing required is basically the same for both cases, this common program is used.

ASSIGN STARTCODE is used to determine how the transaction was started, and ASSIGN QNAME to determine what TD queue triggered it. Details of the event are in the start data or the TD queue record as the case may be.

For illustrative purposes, all events are handled similarly by reporting their details to a TS queue named MONITOR, which can be browsed using CEBR. In practice, for any of the events you can do whatever extra or different processing you require, or (except for unsolicited data) you can ignore the event.

For unsolicited data, the conversation started by FEPI must be accessed so that FEPI knows that the data is being handled. The data itself should be received, or else FEPI ends and restarts the session. For illustration purposes, this program discards the data; in practice, you will probably want to process the data in some way.

However, if you did want to discard such data, you should specify no unsolicited-data handling and use the UNSOLDATAACK property to tell FEPI what action to take, as is done for SLU P mode by these samples.

The general format of the TS queue records is:

```

date time CZUX description
          Event type..ACQFAIL      Pool.....POOLNAME
          Target.....TGTNAME      Node.....NODENAME
          Device.....T3278M2     Event data..X'00000000'
          Format.....0            Event value.176

```

The actual details for each event vary. Events with similar details are grouped together for processing. The groups are:

- Unknown event—an event that is not recognized
- Unsolicited data
- Session lost
- Standard events—all other events.

The groups also determine any additional processing needed. Only unsolicited data needs any processing.

If any errors occur, they are reported to the TS queue.

Program logic

```

Main procedure:
  Determine how transaction was started using ASSIGN
  If started with data by FEPI
    RETRIEVE start data
  If triggered by TD queue
    READ the queue record
  Otherwise
    Report start code
    RETURN
TD-LOOP:
  Locate event type
  Locate device type
  Build description of event: event type, device type,
    format, event value, date/time, transaction
  Call UNKNOWN-EVENT, UNSOLDATA, STANDARD-EVENT, or
    SESSION-LOST according to event group
  If triggered by TD queue
    READ the next queue record
    If one exists, loop to TD-LOOP
  RETURN
UNKNOWN-EVENT routine:
  Write event details to TS queue: description and
    event value
UNSOLDATA routine:

```

```

Write event details to TS queue:  description, event
                                type, pool, target, and node
Access conversation using FEPI ALLOCATE with PASSCONVID
FEPI RECEIVE unsolicited data
Free conversation
Handle data as required

```

Begin session

This program prepares sessions for use by the FEPI sample application programs.

Module name	DFH0VZUC
Transaction name	CZUC
TS queue name	SESSION

The CZUC transaction is started by FEPI when it begins a new session.

The conversation started by FEPI must be accessed so that FEPI knows that the event is being handled. The processing required depends on the data mode and type that the session uses (this is obtained from the start data), and whether the back-end system is IMS or CICS.

For SLU P mode (necessarily IMS), processing depends entirely on local requirements, and is typically used for handling security applications. For illustration purposes, this program gets and discards the initial data. Note that the setup for these samples does not specify a begin-session transaction for SLU P mode.

For SLU2 mode with CICS using formatted data, there is a CICS “good morning” message waiting. The message is received, and the back-end screen is cleared and left ready for a transaction ID to be entered.

For SLU2 mode with CICS using data stream, there may be a “read partition” request waiting which requires a reply—for example, if your pool has device T3279Mx or TPS55Mx specified, or if the logon mode table being used has “extended data stream” specified). Then there is a CICS “good morning” message to be received. A reply is sent to any “read partition” query request, the “good morning” message is received, and the back-end screen is cleared and left ready for a transaction ID to be entered.

For SLU2 mode with IMS, no processing is illustrated.

After the processing, the conversation is freed with the HOLD option, which leaves it ready for use by applications. A report is written to a TS queue named SESSION, which can be browsed using CEBR. The format of the TS queue records is:

```

date time CZUC Begin session completed
          RESP.....0          RESP2.....0
          Target.....TGTNAME    Node.....NODENAME
          Pool.....POOLNAME

```

If any errors occur, a report is written to the TS queue, and the conversation is freed with the RELEASE option, so that the session is ended.

Program logic

```

Main procedure:
  RETRIEVE start data
  Access conversation using FEPI ALLOCATE with PASSCONVID

```

```

Call PROCESS-LUP, PROCESS-FORMATTED, or
PROCESS-DATASTREAM according to data mode and type
Free conversation, keeping session
Write event details to TS queue
RETURN
PROCESS-LUP routine:
FEPI RECEIVE initial data
Handle data as required
PROCESS-FORMATTED routine:
FEPI RECEIVE initial data
Clear back-end screen and make ready for transaction ID
to be entered, using FEPI CONVERSE
PROCESS-DATASTREAM routine:
FEPI RECEIVE
If 'read partition' query
FEPI CONVERSE query reply and get acknowledgment
FEPI RECEIVE initial data
Clear back-end screen and make ready for transaction ID
to be entered, using FEPI CONVERSE

```

Key stroke CONVERSE

This sample program demonstrates using FEPI to obtain information from a back-end transaction using the key stroke data format.

Module names	DFH0VZTK, DFH0PZTK, DFH0CZTK
Transaction name	CZTK
Map names	DFH0MZ1, DFH0MZ6, DFH0MZ7

On the first invocation of the transaction, a map is sent to the front-end terminal.

When there is input from the front-end terminal, CICS invokes the transaction again. The customer number from the input is built into a key stroke sequence which runs a transaction at the back-end. The key strokes are sent and the results received using a FEPI ALLOCATE-CONVERSE-FREE command sequence. Information is extracted from the results and sent to the front-end terminal. Further input is then awaited.

When PF3 or CLEAR is received from the front-end terminal, the transaction ends. If there is an error, the front-end map is reset. These situations are detected using HANDLE CONDITION.

If the back-end sends a CICS message, it is sent on to the front-end terminal, and the transaction ends.

For clarity, error checking is minimal except for the FEPI commands. Note that the key stroke sequence used involves several attention keys, so that if the intermediate responses are not what is expected, the effects are unpredictable. According to your requirements, it may be advisable to send each attention sequence individually and to check each time that the results are as expected.

Screen

```
CZTK                Customer Name and Address Inquiry
Please type a customer number in the range 1 through 9999, then Enter.
Customer Number . . . . .
Name . . . . . :
Address . . . . . :
F3=EXIT to CICS
```

Figure 11. CZTK transaction: customer name and address inquiry

Program logic

```
MAIN procedure:
  Test COMMAREA
  If transaction not previously invoked
    Call SEND-NEW-MAP
  Set up exception condition handling:
    Map error - SEND-NEW-MAP
    CLEAR/PF3 - END-PROG
  RECEIVE MAP from front-end terminal
  Build key stroke sequence to:
    clear back-end screen
    type transaction ID
    ENTER
    type the customer number
    ENTER
  FEPI ALLOCATE conversation with back-end
  FEPI CONVERSE to send key strokes to back-end and get
    the resulting screen image
  FEPI FREE conversation with back-end
  If CICS message received from back-end
    SEND message to front-end terminal
    RETURN
  Get customer information from back-end screen image
  Build data for front-end terminal map
  SEND map data to front-end terminal
  RETURN TRANSID(CZTK) with COMMAREA
SEND-NEW-MAP routine:
  SEND new map to front-end terminal
  RETURN TRANSID(CZTK) with COMMAREA
END-PROG routine:
  Clear front-end terminal
  RETURN
```

Screen image SEND and START

This sample program demonstrates using FEPI to send formatted data to a back-end transaction, and requesting a transaction to be started when the reply to the data arrives.

Module name	DFH0VZTS
Transaction name	CZTS
Map name	DFH0MZ2

This program is the SEND part of a SEND-RECEIVE pair of programs, the RECEIVE part being DFH0VZTR.

On the first invocation of this send transaction, a map is sent to the front-end terminal.

When there is input from the front-end terminal, CICS invokes this send transaction again. The customer number is extracted from the input. Using FEPI ALLOCATE a conversation is started with the back-end system. Then FEPI SEND with screen image data is used to start a back-end transaction. FEPI START is issued to specify that the receive transaction is to be started when the back-end system replies.

In due course, the receive transaction is started and XCTLs to this send transaction. The customer number can now be sent to the back-end using FEPI SEND with screen image data. FEPI START is again issued.

The receive transaction gets the results from the back-end transaction and sends them on to the front-end terminal.

When there is more input from the front-end terminal, CICS invokes this transaction again. FEPI ALLOCATE with PASSCONVID is issued to gain ownership of the conversation and the customer number is sent to the back-end as before. The cycle continues until PF3 or CLEAR is received. These are passed on to the receive transaction (using the FEPI START user data) and to the back-end transaction to indicate that it is to end.

```

CZTS                Customer Name and Balance Inquiry
Please type a customer number in the range 1 through 9999, then Enter.
Customer number . . . . .
Name . . . . . :
Balance . . . . . :
F3=EXIT to CICS

```

Figure 12. CZTS transaction: customer name and balance inquiry

Program logic

```

MAIN procedure:
  Test COMMAREA
  If transaction not previously invoked
    Call SEND-MAP
  If first customer number to process
    Call CONTINUE-CONVERSATION
  Set up exception condition handling:
    Map error - SEND-MAP
    PF3/CLEAR - CONTINUE-CONVERSATION
  RECEIVE MAP from front-end terminal
  If conversation not started
    Call INITIATE-CONVERSATION
  Else
    Call CONTINUE-CONVERSATION
SEND-MAP routine:
  SEND new map to front-end terminal
  RETURN TRANSID(CZTS) with COMMAREA
INITIATE-CONVERSATION routine:
  FEPI ALLOCATE conversation with back-end
  Build screen image to invoke back-end transaction
  FEPI SEND screen image to back-end
  FEPI START the receive transaction
  RETURN
CONTINUE-CONVERSATION routine:
  Unless first customer number
    Reaccess conversation with FEPI ALLOCATE PASSCONVID
  Build screen image to send customer number
  FEPI SEND screen image to back-end
  FEPI START the receive transaction
  RETURN

```

Screen image RECEIVE and EXTRACT FIELD

This sample program demonstrates using FEPI to get formatted data from a back-end transaction.

Module name	DFH0VZTR
Transaction name	CZTR
Map name	DFH0MZ2

This program is the RECEIVE part of a SEND-RECEIVE pair of programs, the SEND part being DFH0VZTS.

This transaction is started by CICS either when data is received from the back-end transaction or if no data is received in the time set in the send transaction, as is determined from the start data obtained with RETRIEVE. The user data in the start data indicates whether the conversation is starting, continuing, or finishing.

A FEPI RECEIVE obtains the screen image from the back-end transaction and FEPI EXTRACT FIELD is used to obtain specific fields.

If the conversation is starting, control is passed to the send transaction using XCTL to allow an inquiry to be sent to the back-end transaction.

If the conversation is continuing, the results from the back-end are sent on to the front-end terminal. Access to the conversation is relinquished, and control is returned to CICS specifying that the send transaction is to be invoked when there is next user input.

If the conversation has finished, a message to that effect is sent to the front-end terminal. The conversation is freed and the transaction ends.

Program logic

```
MAIN procedure:
  RETRIEVE start data
  Reaccess conversation with FEPI ALLOCATE PASSCONVID
  If time out
    Call REPORT-PROBLEM
  FEPI RECEIVE back-end screen image
  If conversation ending (PF3 or CLEAR indicated)
    Call REPORT-PROBLEM
  If back-end problem
    (CICS message or back-end transaction message)
    Call REPORT-PROBLEM
  If conversation starting (user data has customer number)
    XCTL to program DFH0VZTS
  If conversation continuing
    Get interesting fields from back-end data using
      FEPI EXTRACT FIELD
    Build and send map to front-end terminal
    Release conversation using FEPI FREE PASS
    RETURN TRANSID(CZTS) with COMMAREA
REPORT-PROBLEM routine:
  SEND message to front-end terminal
  FEPI FREE conversation
  RETURN
```

3270 data stream passthrough

This sample program demonstrates using FEPI to passthrough 3270 data stream between a back-end application and a front-end terminal.

Module names	DFH0VZTD, DFH0AZTD
Transaction name	CZTD

On the first invocation of the transaction, a request is sent to the back-end system to start a transaction there. The response is sent on to the front-end terminal.

When there is input from the front-end terminal, CICS reinvoles the transaction. This input is sent on to the back-end system, using the FEPI CONVERSE command, and the resulting response is returned to the front-end terminal.

If there is an error, or the back-end system sends a CICS message, or PF3 is received from the front-end terminal, the transaction ends.

Program logic

```
Test COMMAREA
If transaction not previously invoked
    Build data stream request to start back-end transaction
    FEPI ALLOCATE conversation with back-end system
    FEPI CONVERSE data stream to and from back-end system
    SEND returned data stream to the front-end terminal
Else
    RECEIVE data stream from the front-end terminal
    Prepare data stream to send on to back-end system
    Reaccess conversation with FEPI ALLOCATE PASSCONVID
    FEPI CONVERSE data stream to and from back-end system
    SEND data stream to the front-end terminal
If error during processing
    SEND explanatory message
If continuing
    Release conversation using FEPI FREE PASS
    RETURN TRANSID(CZTD) with COMMAREA
Else (error, CICS message, or PF3)
    FEPI FREE conversation
    RETURN
```

End-session handler

This program cleans up sessions after use by FEPI sample application programs.

Module name	DFH0VZUU
Transaction name	CZUU
TS queue name	SESSION

This transaction is started by FEPI when an application ends a conversation or when a session is released.

The conversation passed by FEPI must be accessed so that FEPI knows that the event is being handled. The processing required depends entirely on local requirements. For illustration purposes, this program keeps the session for use by another conversation or lets it end, depending on the event type.

The CONVID picked up from the START data and passed on the FEPI ALLOCATE PASSCONVID is not the same as the CONVID for the conversation that has been freed. Nevertheless, the end-session handler can use it to access the same FEPI terminal.

For end of conversation (EVENTTYPE=FREE in start data), processing could typically involve setting the session back to a known state (such as a clear back-end screen ready to accept a new transaction name), or handling security, or overriding the type of FREE used. Such processing would depend on the data mode and type that the session uses (which is obtained from the start data), whether the back-end system is CICS or IMS, and the type of FREE used (also obtained from the start data).

For end of session (EVENTTYPE=FREE and EVENTVALUE=RELEASE in start data), processing could typically involve handling security.

For both cases, there could be an indication (in EVENTVALUE in the start data) that CICS is shutting down, which might require alternative special processing. This transaction would have to be in the XLT to allow it to be started during shutdown.

After the processing, a report is written to a TS queue named SESSION, which can be browsed using CEBR. The format of the TS queue records is:

```
date time CZUU End-session handling completed
          RESP.....0          RESP2.....0
          Target.....TGTNAME    Node.....NODENAME
          Pool.....POOLNAME
```

Program logic

```
Main procedure:
  RETRIEVE start data
  Access conversation using FEPI ALLOCATE with PASSCONVID
  Call PROCESS-RELEASE or PROCESS-FREE as appropriate
  Write event details to TS queue
  RETURN
PROCESS-RELEASE routine:
  Handle as required
  Free conversation, ending session
PROCESS-FREE routine:
  Handle as required
  Free conversation, keeping session
```

SLU P one-out one-in

This sample program demonstrates using FEPI to obtain information from a back-end IMS system, using SLU P mode and the FEPI CONVERSE command with the POOL option.

Module names	DFH0VZPS, DFH0AZPS
Transaction name	CZPS
Map names	DFH0MZ4, DFH0MZ8

On the first invocation of the program, a map is sent to the front-end terminal.

When there is input from the front-end terminal, CICS reinvokes the program. A simple inquiry is made to the back-end system—for illustration purposes, it asks the time—and the answer is displayed on the front-end terminal. Because the

inquiry requires only a one-out one-in exchange with the back-end system, a temporary conversation can be used, so the FEPI CONVERSE command with the POOL option is used.

When PF3 or CLEAR is received from the front-end terminal, the transaction ends. If there is an error, the front-end map is reset. These situations are detected using HANDLE CONDITION.

If the back-end system sends an IMS message, it is sent on to the front-end terminal and the transaction ends.

For clarity, error checking is minimal except for the FEPI commands.

```

CZPS          SLU P Sample Program.
IMS SLU P conversational sample program
This transaction will process a FEPI CONVERSE command to obtain time
and date from a back-end IMS system.
DATE   : 02/04/92
TIME   : 10:57:10
STATE  : Not started
F3=EXIT to CICS  ENTER=obtain time and date stamp from IMS
  
```

Figure 13. CZPS transaction: SLU P sample program

Program logic

```

MAIN procedure:
  Test COMMAREA
  If transaction not previously invoked
    Call SEND-NEW-MAP
  Set up exception condition handling:
    Map error - SEND-NEW-MAP
    CLEAR/PF3 - END-PROG
  RECEIVE MAP from front-end terminal
  Build SLU P data stream to request time from back-end IMS
  system
  FEPI CONVERSE to send data stream to the back-end and get
  the message containing the time
  If IMS message received from back-end system
    SEND message to front-end terminal
    RETURN
  Build data for front-end terminal map
  SEND map data to front-end terminal
  RETURN TRANSID(CZPS) with COMMAREA
SEND-NEW-MAP routine:
  SEND new map
  RETURN TRANSID(CZPS) with COMMAREA
END-PROG routine:
  Clear front-end terminal
  RETURN
  
```

SLU P pseudoconversational

This sample program demonstrates using FEPI to obtain data from an IMS back-end transaction. It is in pseudoconversational style, using the FEPI START command to schedule itself when the results arrive.

Module names	DFH0VZPA, DFH0AZPA
Transaction name	CZPA
Map names	DFH0MZ5, DFH0MZ9

On the first invocation of the program, a map is sent to the front-end terminal.

When there is input from the front-end terminal, CICS invokes the program again. After establishing a conversation, an inquiry is sent to the back-end system. FEPI START is issued to start this program again when the results arrive. Meanwhile it returns to CICS, so releasing resources.

When the results arrive, FEPI starts the program again. The results are obtained using FEPI RECEIVE, and sent on to the front-end terminal. The conversation is freed and the program returns to CICS to await more input. If the back-end system sends an IMS message, it is sent on to the front-end terminal and the transaction ends.

When PF3 or CLEAR is received from the front-end terminal, the transaction ends. If there is an error, the front-end map is reset. These situations are detected using HANDLE CONDITION.

For clarity, error checking is minimal except for the FEPI commands.

Screen

```
CZPA          SLUP Sample Program.
IMS SLUP Pseudoconversational sample program
This transaction will process SEND/START/RECEIVE requests with MFS
specified, to a back-end IMS system.
DATE   : 02/04/92
TIME   : 10:58:50
STATE  : Not Started
F3=EXIT to CICS  ENTER=obtain time and date stamp from IMS
```

Figure 14. CZPA transaction: SLU P pseudoconversational sample program

Program logic

```
MAIN procedure:
  If started from terminal
    Test COMMAREA
    If transaction not previously invoked
      Call SEND-NEW-MAP
    Set up exception condition handling:
      Map error - SEND-NEW-MAP
      CLEAR/PF3 - END-PROG
    RECEIVE map from front-end terminal
    FEPI ALLOCATE conversation with back-end system
    Build SLU P data stream to request time
    FEPI SEND data stream to back-end system
    FEPI START transaction
    RETURN
  If started by FEPI
    RETRIEVE start data
    Reaccess conversation using FEPI ALLOCATE PASSCONVID
    If EVENTTYPE = data received
      FEPI RECEIVE data stream from back-end system
      FEPI FREE conversation
      If IMS message received
        SEND message to front-end terminal
        RETURN
      Build data for front-end terminal map
      SEND map to front-end terminal
      RETURN TRANSID(CZPA) with COMMAREA
    Otherwise (timeout or session loss)
      SEND map with message to front-end terminal
      RETURN (freeing conversation implicitly)
```

```

SEND-NEW-MAP routine:
  SEND new map
  RETURN TRANSID(CZPA) with COMMAREA
END-PROG routine:
  Clear front-end terminal
  RETURN

```

STSN handler

This program handles STSN processing for the FEPI sample application programs.

Module name	DFH0AZQS
Transaction name	CZQS
TS queue name	SESSION

The CZQS transaction is started by FEPI when a request for message resynchronization ('set and test sequence number', STSN) or a 'start data traffic' indication is received from a back-end IMS system.

The conversation passed by FEPI must be accessed so that FEPI knows that the event is being handled. The processing required depends on the STSN status, which is obtained using FEPI EXTRACT STSN.

For STSNSTATUS=NOSTSN, the transaction was started because 'start data traffic' arrived. A DR1 normal response must be sent.

For STSNSTATUS=STSNSET, a positive STSN response must be sent.

For STSNSTATUS=STSNTEST, processing would typically involve comparing saved sequence numbers with those received from the back-end IMS system to determine what response to send. The *IMS Customization Guide* gives advice on the appropriate action.

After the processing, the response is sent using FEPI ISSUE. A report is written to a TS queue named SESSION, which can be browsed using CEBR. The general format of the TS queue records is:

```

date time CZQS STSN processing completed
          Target.....TGTNAME      Node.....NODENAME
          Seqnumin...nnnn          Seqnumout...nnnn
          STSN status.XXXXXXX      Response...XXXXXXX

```

Program logic

```

Main procedure:
  RETRIEVE start data
  Access conversation using FEPI ALLOCATE with PASSCONVID
  Get STSN status using FEPI EXTRACT STSN
  Call NOSTSN, STSNSET, or STSNTEST
    according to STSN status
  Send response using FEPI ISSUE CONTROL
  Write event details to TS queue
  Free conversation, keeping session
  RETURN
NOSTSN routine:
  Build DR1 normal response
STSNSET routine:
  Build STSN positive response
STSNTEST routine:
  Handle as required
  Build required response

```

Appendix B. CVDA and RESP2 values for FEPI commands

This appendix lists the CVDA and RESP2 values returned by FEPI commands.

It contains:

- “FEPI CVDA and numeric values in alphabetic sequence”
- “FEPI CVDA and numeric values in numeric sequence” on page 257
- “FEPI RESP2 values” on page 260.

FEPI CVDA and numeric values in alphabetic sequence

The following table lists the CVDA values used or returned by the FEPI commands in alphabetic sequence.

For programming information about other CVDA values, see the *CICS System Programming Reference*.

Table 16. CVDA values in alphabetic sequence

ACQFAIL	515
ACQUIRED	69
ACQUIRING	71
ADDFAIL	519
ALARM	501
APPLICATION	559
ATTENTION	524
BEGINSESSION	510
CANCEL	526
CD	491
DATA	508
DATASTREAM	543
DEFRESP1	497
DEFRESP1OR2	528
DEFRESP2	498
DEFRESP3	499
DELETEDFAIL	520
DISCARDFAIL	513
EB	490
EXCEPTRESP	523
FMH	502
FORCE	342
FORMATTED	542
FREE	85
GOINGOUT	172

Table 16. CVDA values in alphabetic sequence (continued)

HOLD	163
INBOUND	547
INOUT	532
INPUT	226
INSERVICE	73
INSTALLED	550
INSTALLFAIL	512
INVALID	359
LIC	493
LOSE	544
LUP	541
LUSTAT	525
MDT	506
MORE	492
NEGATIVE	530
NEWSESSION	485
NOALARM	500
NOCONVERT	734
NOCONV	556
NOFMH	503
NOMDT	507
NOMSGJRNL	531
NONE	496
NORMALRESP	522
NOSTSN	487
NOTINBOUND	546
NOTINSTALLED	551
OLDSESSION	486
OUTPUT	227
OUTSERVICE	74
PENDBEGIN	558
PENDDATA	560
PENDFREE	86
PENDPASS	565
PENDRELEASE	562
PENDSTART	561
PENDSTSN	557
PENDUNSOL	564
POSITIVE	529
PROTECTED	504
RELEASE	563

Table 16. CVDA values in alphabetic sequence (continued)

RELEASED	70
RELEASING	549
RESET	290
RTR	527
RU	494
SESSION	372
SESSIONFAIL	517
SESSIONLOST	516
SETFAIL	514
SHUTDOWN	288
STSN	509
STSNSET	488
STSNTEST	489
TASK	233
TIMEOUT	511
TPS55M2	552
TPS55M3	553
TPS55M4	554
T3278M2	533
T3278M3	534
T3278M4	535
T3278M5	536
T3279M2	537
T3279M3	538
T3279M4	539
T3279M5	540
UNPROTECTED	505
UNSOLDATA	521
WIN	545

Related reference

“FEPI CVDAs and numeric values in numeric sequence”

The following table lists the CVDA values used or returned by the FEPI commands in numeric sequence.

Related information

CICS-value data areas used by all commands

FEPI CVDAs and numeric values in numeric sequence

The following table lists the CVDA values used or returned by the FEPI commands in numeric sequence.

For programming information about other CVDA values, see the *CICS System Programming Reference*.

Table 17. CVDA values in numeric sequence

69	ACQUIRED
70	RELEASED
71	ACQUIRING
73	INSERVICE
74	OUTSERVICE
85	FREE
86	PENDFREE
163	HOLD
172	GOINGOUT
226	INPUT
227	OUTPUT
233	TASK
288	SHUTDOWN
290	RESET
342	FORCE
359	INVALID
372	SESSION
485	NEWSESSION
486	OLDSESSION
487	NOSTSN
488	STSNSET
489	STSNTEST
490	EB
491	CD
492	MORE
493	LIC
494	RU
496	NONE
497	DEFRESP1
498	DEFRESP2
499	DEFRESP3
500	NOALARM
501	ALARM
502	FMH
503	NOFMH
504	PROTECTED
505	UNPROTECTED
506	MDT
507	NOMDT
508	DATA
509	STSN

Table 17. CVDA values in numeric sequence (continued)

510	BEGINSESSION
511	TIMEOUT
512	INSTALLFAIL
513	DISCARDFAIL
514	SETFAIL
515	ACQFAIL
516	SESSIONLOST
517	SESSIONFAIL
519	ADDFAIL
520	DELETEFAIL
521	UNSOLDATA
522	NORMALRESP
523	EXCEPTRESP
524	ATTENTION
525	LUSTAT
526	CANCEL
527	RTR
528	DEFRESP1OR2
529	POSITIVE
530	NEGATIVE
531	NOMSGJRNL
532	INOUT
533	T3278M2
534	T3278M3
535	T3278M4
536	T3278M5
537	T3279M2
538	T3279M3
539	T3279M4
540	T3279M5
541	LUP
542	FORMATTED
543	DATASTREAM
544	LOSE
545	WIN
546	NOTINBOUND
547	INBOUND
549	RELEASING
550	INSTALLED
551	NOTINSTALLED
552	TPS55M2

Table 17. CVDA values in numeric sequence (continued)

553	TPS55M3
554	TPS55M4
556	NOCONV
557	PENDSTSN
558	PENDBEGIN
559	APPLICATION
560	PENDDATA
561	PENDSTART
562	PENDRELEASE
563	RELEASE
564	PENDUNSOL
565	PENDPASS
734	NOCONVERT

Related reference

“FEPI CVDA’s and numeric values in alphabetic sequence” on page 255
 The following table lists the CVDA values used or returned by the FEPI commands in alphabetic sequence.

Related information

CICS-value data areas used by all commands

FEPI RESP2 values

RESP2 values are used in the EVENTVALUE area of FEPI transient data queue records and returned by the RESP2 option of FEPI commands.

For details of the error conditions and related RESP2 values for each FEPI command, see the command definitions in Chapter 11, “FEPI system programming reference,” on page 89 and Part 4, “FEPI application programming reference,” on page 175.

Declarations for the RESP2 values are provided in the following copy books:

- DFHSZAPA for Assembler language
- DFHSZAPO for COBOL
- DFHSZAPP for PL/I
- DFHSZAPC for C.

Table 18. RESP2 values

1	INQUIRE START, NEXT, or END command not valid here: START Browse of this resource type already in progress NEXT INQUIRE START not issued END INQUIRE START not issued.
2	All resource definitions have been retrieved.
10	Command bypassed by user exit.
11	FEPI not installed or not active.
12	CICS shutting down, command not allowed.
13	FEPI not available.

Table 18. RESP2 values (continued)

14	FEPI busy or cannot get storage.
15	Unknown command.
16	Internal problem.
17	FEPI cannot get storage for user exit parameters.
18	Command failed because of operator or system action.
30	POOL name not known.
31	POOL name out of service.
32	TARGET name not known.
33	TARGET name out of service.
34	TARGET name required but not specified.
35	Pool name is unsuitable for temporary conversations. It has CONTENTION(LOSE) or INITIALDATA(INBOUND) but no begin-session handler.
36	No suitable session available and in service.
40	[FROM]FLENGTH value is negative, zero, or more than MAXFLENGTH value for pool.
41	ESCAPE value not valid.
50	Inbound data with 'begin bracket' to be received.
51	Attention identifier (AID) not valid.
52	Cursor position not valid.
53	Code points in formatted data not valid.
54	Attribute positions or values in send data not valid.
55	Key stroke escape sequence in send data not valid.
56	Field validation (mandatory fill, mandatory enter, trigger) failed.
57	Input is inhibited.
58	VTAM SEND failed.
59	DBCS data rules violated.
60	MAXFLENGTH value negative, or greater than MAXFLENGTH value for pool.
61	FLENGTH value negative or greater than 128.
62	TRANSID name not valid.
63	TERMID name not valid.
70	FIELDLOC or FIELDNUM value negative or not valid.
71	VTAM RECEIVE failed.
72	RECEIVE FORMATTED processing found invalid, or unexpected data while interpreting the 3270 data stream for a WRITE, ERASE/WRITE, ERASE/WRITE ALTERNATE, or WRITE STRUCTURED FIELD command code.
80	CONTROL value not valid.
81	VALUE not valid: omitted when required; included when not required; or unsuitable for specified CONTROL.
82	SENSEDATA option omitted when required, or specified when not required.
90	Definite response type did not match what was required.
91	Only NORMALRESP or EXCEPTRESP allowed at this point in conversation.
92	Response to STSN SET was not positive.
93	Only STSN allowed at this point in conversation.
94	Only STSN or NORMALRESP allowed at this point in conversation.
95	CONTROL value not allowed at this point in conversation.
100	Not authorized to issue command.
110	SERVSTATUS value not valid.
111	ACQSTATUS value not valid.
115	POOL name not known.
116	TARGET name not known.
117	NODE name not known.

Table 18. RESP2 values (continued)

118	Unknown connection (TARGET and NODE names known, but not in a common POOL).
119	Request failed for one or more items in list. Detailed errors reported to TD queue for monitor to handle.
130	TARGETNUM value negative, zero, or not valid.
131	NODENUM value negative, zero, or not valid.
132	POOLNUM value negative, zero, or not valid.
140	DEVICE value not valid.
141	CONTENTION value not valid.
142	INITIALDATA value not valid.
143	UNSOLDATAACK value not valid.
144	MSGJRNL value not valid.
150	FORMAT value not valid or unsuitable for specified device.
153	STSN name not valid or STSN unsuitable for specified device.
154	BEGINSESSION value not valid.
155	UNSOLDATA value not valid.
156	EXCEPTIONQ value not valid.
157	FJOURNALNUM value not valid.
158	MAXLENGTH value not valid.
159	ENDSESSION name not valid.
160	PROPERTYSET name not valid.
162	POOL name not valid.
163	NODE name not valid.
164	TARGET name not valid.
167	APPL name not valid.
170	PROPERTYSET name already exists.
171	PROPERTYSET name not known.
172	POOL name already exists.
173	NODE name already exists.
174	TARGET name already exists.
175	Connection already exists.
176	VTAM OPEN NODE failed.
177	VTAM APPLID already known.
178	FJOURNALNAME value not valid.
182	Session unbound, unrecoverable.
183	Session unbound, recoverable.
184	Session unbound, error.
185	Session unbound, bind coming.
186	Session unbound.
187	Lost terminal.
188	CLEANUP, abnormal.
189	CLEANUP.
190	UNBIND error.
191	SETUP error.
192	SSCP error.
193	SLU error.
194	PLU error.
195	BIND error.
196	CINIT error.
197	REQSESS error.
198	REQSESS inhibited.
199	REQSESS not available.
210	Option not valid for SLU P.
211	Option not valid for SLU2.

Table 18. RESP2 values (continued)

212	Wrong data format for conversation.
213	Command has timed out.
214	CICS shutting down, conversation should be ended.
215	Session lost.
216	Error occurred on previous SEND command.
220	SEND or CONVERSE command not allowed at this point in conversation.
221	RECEIVE command not allowed at this point in conversation.
223	START command not allowed at this point in conversation.
224	Only ISSUE or FREE allowed at this point in conversation.
230	SNA CLEAR command received.
231	SNA CANCEL command received.
232	SNA CHASE command received.
233	Exception response received.
234	Exception request received.
240	Conversation ID unknown or not owned by task.
241	TIMEOUT value negative or not valid.
250	Passticket not built successfully.
251	CICS ESM interface not initialized.
252	Unknown return code in ESMRESP from the ESM.
253	Unrecognized response from CICS security modules.
254	Function unavailable.
259	No signed-on user.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply in the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who 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 United Kingdom Laboratories, MP151, Hursley Park, Winchester, Hampshire, England, SO21 2JN.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other product and service names might be trademarks of IBM or other companies.

Bibliography

CICS books for CICS Transaction Server for z/OS

General

CICS Transaction Server for z/OS Program Directory, GI13-0536
CICS Transaction Server for z/OS What's New, GC34-6994
CICS Transaction Server for z/OS Upgrading from CICS TS Version 2.3, GC34-6996
CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.1, GC34-6997
CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.2, GC34-6998
CICS Transaction Server for z/OS Installation Guide, GC34-6995

Access to CICS

CICS Internet Guide, SC34-7021
CICS Web Services Guide, SC34-7020

Administration

CICS System Definition Guide, SC34-6999
CICS Customization Guide, SC34-7001
CICS Resource Definition Guide, SC34-7000
CICS Operations and Utilities Guide, SC34-7002
CICS RACF Security Guide, SC34-7003
CICS Supplied Transactions, SC34-7004

Programming

CICS Application Programming Guide, SC34-7022
CICS Application Programming Reference, SC34-7023
CICS System Programming Reference, SC34-7024
CICS Front End Programming Interface User's Guide, SC34-7027
CICS C++ OO Class Libraries, SC34-7026
CICS Distributed Transaction Programming Guide, SC34-7028
CICS Business Transaction Services, SC34-7029
Java Applications in CICS, SC34-7025

Diagnosis

CICS Problem Determination Guide, GC34-7034
CICS Performance Guide, SC34-7033
CICS Messages and Codes, SC34-7035
CICS Diagnosis Reference, GC34-7038
CICS Recovery and Restart Guide, SC34-7012
CICS Data Areas, GC34-7014
CICS Trace Entries, SC34-7013
CICS Supplementary Data Areas, GC34-7015
CICS Debugging Tools Interfaces Reference, GC34-7039

Communication

CICS Intercommunication Guide, SC34-7018
CICS External Interfaces Guide, SC34-7019

Databases

CICS DB2 Guide, SC34-7011
CICS IMS Database Control Guide, SC34-7016

CICSplex SM books for CICS Transaction Server for z/OS

General

CICSplex SM Concepts and Planning, SC34-7044
CICSplex SM Web User Interface Guide, SC34-7045

Administration and Management

CICSplex SM Administration, SC34-7005
CICSplex SM Operations Views Reference, SC34-7006
CICSplex SM Monitor Views Reference, SC34-7007
CICSplex SM Managing Workloads, SC34-7008
CICSplex SM Managing Resource Usage, SC34-7009
CICSplex SM Managing Business Applications, SC34-7010

Programming

CICSplex SM Application Programming Guide, SC34-7030
CICSplex SM Application Programming Reference, SC34-7031

Diagnosis

CICSplex SM Resource Tables Reference, SC34-7032
CICSplex SM Messages and Codes, GC34-7035
CICSplex SM Problem Determination, GC34-7037

Other CICS publications

The following publications contain further information about CICS, but are not provided as part of CICS Transaction Server for z/OS, Version 4 Release 1.

Designing and Programming CICS Applications, SR23-9692
CICS Application Migration Aid Guide, SC33-0768
CICS Family: API Structure, SC33-1007
CICS Family: Client/Server Programming, SC33-1435
CICS Family: Interproduct Communication, SC34-6853
CICS Family: Communicating from CICS on System/390, SC34-6854
CICS Transaction Gateway for z/OS Administration, SC34-5528
CICS Family: General Information, GC33-0155
CICS 4.1 Sample Applications Guide, SC33-1173
CICS/ESA 3.3 XRF Guide, SC33-0661

Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully.

You can perform most tasks required to set up, run, and maintain your CICS system in one of these ways:

- using a 3270 emulator logged on to CICS
- using a 3270 emulator logged on to TSO
- using a 3270 emulator as an MVS system console

IBM Personal Communications provides 3270 emulation with accessibility features for people with disabilities. You can use this product to provide the accessibility features you need in your CICS system.

Index

Special characters

, for FEPI resources 14

Numerics

16MB line, AMODE setting for FEPI 39
3270 data stream
 data formats 154
 data-stream-level commands 13
 pass-through sample program 250
3278 device type 102
3279 device type 102

A

abends 132
access program 15, 157
ACQFAIL event 47
ACQNUM option
 FEPI INQUIRE CONNECTION 97
 FEPI INQUIRE NODE 100
ACQSTATUS option
 FEPI ADD 92
 FEPI INQUIRE CONNECTION 97
 FEPI INQUIRE NODE 100
 FEPI INSTALL NODELIST 110
 FEPI INSTALL POOL 111
 FEPI SET CONNECTION 118
 FEPI SET NODE 120
ACQSTATUS, resource status 53
Acquired
 CEMT INQUIRE
 FECONNECTION 68
 CEMT INQUIRE FENODE 70
ACQUIRED option
 CEMT SET FECONNECTION 75
 CEMT SET FENODE 76
ACQUIRED resource status 53
ACQUIRING, resource status 53
ADDFAIL event 48
addressing mode 39, 139
AID (attention identifier), screen-image data 147
ALARMSTATUS option
 FEPI CONVERSE FORMATTED 194
 FEPI RECEIVE FORMATTED 217
ALL option
 CEMT SET FECONNECTION 75
 CEMT SET FENODE 76
 CEMT SET FEPOOL 77
 CEMT SET FETARGET 78
ALLOCATE command 181
 conversation 162
 PASSCONVID 181
 POOL 183
ALLOCATE POOL 183
AMODE setting
 application programs 139
 system programs 39
analysis and planning 21

AP NOOP 185
APAR (authorized program analysis report) 7
Appl
 CEMT INQUIRE FETARGET 74
APPL option
 FEPI INQUIRE TARGET 109
application programming 135
 commands 12, 175
 components of FEPI programs 15, 157
 conversational 160, 162
 CVDA values 255, 257
 data stream integrity 150
 design 157
 general sequence of commands 141
 IMS considerations 168
 IMS response mode 169
 one-out one-in conversational 159
 performance considerations 55, 171
 pseudoconversational 160
 RESP2 values 260
 writing FEPI programs 139
APPLLIST option
 FEPI INSTALL TARGETLIST 117
Assembler language
 copybook 39, 178
 sample programs 237
ATI (automatic transaction initiation)
 controlling FEPI resources 42
 unsolicited data 159
attention
 general sequence of commands 141
 keys 143, 232
 multiple attentions 145
 sending screen-image data 146
availability
 of network resources 35

B

back-end system 4
 CICS sample program 240
 hardware and software requirements 7
 IMS considerations 168
 IMS sample programs 241
 in a CICSplex 26
 initial data 158
 message sent after a bind 158
 planning 21
 sample configuration 45
 XRF takeover 60
BACKGROUND option
 FEPI EXTRACT FIELD 201
begin-session handler 15
 application design 158
 defining to FEPI 25, 42
 IMS considerations 169
 sample program 245

BEGINSESSION option
 FEPI INQUIRE POOL 102
 FEPI INQUIRE PROPERTYSET 106
 FEPI INSTALL PROPERTYSET 114
BEING ACQUIRED status 54
BEING RELEASED status 54
bind
 communication and conversations 137
 device query 158
 handling unsolicited binds with CLSDST(PASS) 47, 48, 49
 introduction to FEPI resources 15
 removing bind races 59
 selection of FEPI session parameters 36
 system message after a bind 158
 XRF takeover 59
bypass
 handling in application 167
 using in user exit 51

C

C language
 copybook 39, 178
 nulls in screen image 147
 sample programs 237
card reader, sending key stroke data 143
CDSA storage requirements 26
CECI transaction
 debugging FEPI programs 39, 140
CEMT transaction 65
 after FEPI failure 58
 DISCARD 65
 INQUIRE FECONNECTION 66
 INQUIRE FENODE 69
 INQUIRE FEPOOL 71
 INQUIRE FEPROPSET 73
 INQUIRE FETARGET 73
 SET FECONNECTION 75
 SET FENODE 76
 SET FEPOOL 77
 SET FETARGET 78
CETR transaction 131
CHAIN option
 FEPI CONVERSE
 DATASTREAM 188
 FEPI RECEIVE DATASTREAM 213
chain, receiving a 154
CICS (Customer Information Control System)
 back-end
 sample program 240
 terminal definitions 37
 CICS-supplied transactions 65
 default startup group list, DFHLIST 29, 30
 front-end, configuration of 33
 ISC and MRO considerations 4
 RDO group DFHFPEPI 29

- CICS (Customer Information Control System) *(continued)*
 - shutdown 57
 - startup procedure 31
 - updating definitions 29
 - CICSplex
 - back-end systems in 26
 - CLSDST(PASS) 48
 - COBOL language
 - copybook 39, 178
 - example of sending screen-image data 146
 - sample programs 237
 - COLLECT STATISTICS command 56
 - COLOR option
 - FEPI EXTRACT FIELD 201
 - color support
 - device attributes 25
 - getting colors 201
 - storage requirements 27
 - VTAM configuration 34
 - COLUMNS option
 - FEPI CONVERSE FORMATTED 194
 - FEPI RECEIVE FORMATTED 217
 - command-level security 17, 30
 - commands
 - application programming reference section 175
 - CEMT DISCARD 65
 - copy books for RESP2 values 39, 178
 - CVDA values 255, 257
 - data-stream-level 13, 149
 - errors and exception conditions 39, 140
 - formatted data 12
 - general sequence 141, 150
 - high-level FEPI 12
 - introduction to FEPI commands 12
 - key stroke interface 12, 141
 - list of FEPI commands 13
 - performance considerations 55
 - RESP2 values 260
 - screen-image interface 12, 141
 - SNA 175
 - specialized-level 13, 173
 - storage requirements 26
 - system programming 38, 89
 - VTAM-level 13
 - communication
 - error handling 167
 - general considerations 137
 - resources 137
 - conditions
 - error and exception 90, 178
 - configuration
 - 16MB line 39
 - AMODE setting 39
 - example of 42
 - of back-end CICS and IMS systems 37
 - of CICS 33
 - of FEPI 40
 - coding of programs 38
 - global user exits 50
 - monitoring program 46
 - sample 42
 - writing operator transactions 53
 - configuration *(continued)*
 - of VTAM
 - ISTINCLM mode table 53
 - session pacing values 37
 - session parameters 35
 - planning 21
 - programs, design of 38
 - sample programs 42, 242
 - connection 15
 - acquiring and releasing 53
 - controlling waits with event handlers 42
 - determining contents of a pool 53
 - INQUIRE CONNECTION
 - command 96
 - sample configuration 45
 - SET CONNECTION command 118
 - storage requirements 26
 - waiting in RELEASING state 54
- contention mode 145, 152
- CONTENTION option
 - FEPI INQUIRE POOL 102
 - FEPI INQUIRE PROPERTYSET 106
 - FEPI INSTALL PROPERTYSET 114
- CONTROL option
 - FEPI ISSUE 209
- conventions used by FEPI
 - node names 34
 - pool names 24
 - property set names 26
 - systems and data flow 11
- conversation ID 162
- conversation identifier 162
- conversations 137, 161
 - design of conversational applications 160
 - ownership of 162
 - passing conversations 163
 - state of 55
 - storage requirements 26
 - temporary 163
 - unknown conversation ID, error handling 167
- CONVERSE
 - data stream applications 154
 - DATASTREAM 187
 - FORMATTED 193
 - key stroke and screen image applications 148
- CONVID field
 - start data 229
 - TDQ record 124
- CONVID option
 - FEPI ALLOCATE POOL 183
 - FEPI CONVERSE
 - DATASTREAM 188
 - FEPI CONVERSE FORMATTED 194
 - FEPI EXTRACT CONV 199
 - FEPI EXTRACT FIELD 201
 - FEPI EXTRACT STSN 205
 - FEPI FREE 207
 - FEPI ISSUE 210
 - FEPI RECEIVE DATASTREAM 213
 - FEPI RECEIVE FORMATTED 217
 - FEPI REQUEST PASTTICKET 221
 - FEPI SEND DATASTREAM 223
 - FEPI SEND FORMATTED 225
- CONVID option *(continued)*
 - FEPI START 227
- CONVNUM option
 - FEPI INQUIRE CONNECTION 97
- CSZL transient data queue
 - command errors 39, 91
 - defining 29
- CSZX transient data queue
 - command errors 39, 125
 - defining 29
 - record format 124
 - reporting unexpected events 47
- CURSOR option
 - FEPI RECEIVE FORMATTED 217
 - FEPI SEND FORMATTED 225
- cursor setting 143, 147
- customization
 - journaling 85
- CVDA values 255
- CZBC transaction 240
- CZBI transaction 241
- CZPA transaction 252
- CZPS transaction 251
- CZQS transaction 254
- CZTD transaction 250
- CZTK transaction 246
- CZTR transaction 249
- CZTS transaction 247
- CZUC transaction 245
- CZUU transaction 250
- CZUX transaction 243
- CZXS transaction 242
- ## D
- data formats
 - inbound data 232
 - journaling 85
 - start data 229
 - TD queue records 123
 - data handling, using property set for 25, 40
 - data stream applications 13
 - 3270 pass-through sample program 250
 - converse 154
 - data formats 231
 - data stream integrity 150
 - FEPI commands 12, 13
 - receiving 150
 - sending 153
 - SLU P mode 155
 - SLU2 mode 154
 - writing 149
 - DATATYPE field
 - start data 229
 - TDQ record 124
 - DBCS (double-byte character set)
 - errors sending key stroke data 144
 - formatted, mode 232
 - key stroke format 231
 - debugging 39, 127
 - default CICS startup group list, DFHLIST 29, 30
 - defining transient data queues 29
 - definite responses 158, 174

- definitions
 - for sample programs 238
 - sample 45
 - updating CICS 29
- DELETE POOL command 93
- DELETEFAIL event 48
- design
 - access program 157
 - application organization 159
 - begin-session handler 158
 - end-session handler 159
 - programs 157
 - unsolicited-data handler 158
- Device
 - CEMT INQUIRE FEPOOL 72
- device attributes, using property set
 - for 25, 40
- DEVICE field
 - start data 229
 - TDQ record 124
- DEVICE option
 - FEPI EXTRACT CONV 199
 - FEPI INQUIRE POOL 102
 - FEPI INQUIRE PROPERTYSET 106
 - FEPI INSTALL PROPERTYSET 114
- device query 158
- device-type, VTAM logon mode table
 - entries 35
- DFHFEPI, RDO group 29, 30
- DFHLIST, default CICS startup group
 - list 29, 30
- DFHSZ4099E message 132
- DFHSZ4155I message 133
- DFHSZAPA, copy book 39, 178
- DFHSZAPC, copy book 39, 178
- DFHSZAPO, copy book 39, 178
- DFHSZAPP, copy book 39, 178
- DISCARD command
 - NODELIST 94
 - POOL 94
 - PROPERTYSET 95
 - TARGETLIST 95
- DISCARDFAIL event 47
- distributed program link, shipping FEPI
 - applications 33
- distribution tape 8
- DRx responses 158, 174
- dumps
 - FEPI 127

E

- ECDSA storage requirements 26
- EDF (Execution Diagnostic Facility)
 - debugging FEPI programs 39, 140
 - FEPI problem determination aids 127
- EIB (EXEC interface block) 89, 177
- end-session handler 15
 - application design 159
 - defining to FEPI 25, 42
 - IMS considerations 169
 - sample program 250
- ENDESESSION option
 - FEPI INQUIRE POOL 103
 - FEPI INQUIRE PROPERTYSET 106
 - FEPI INSTALL PROPERTYSET 115

- ENDSTATUS option
 - data stream 151
- FEPI CONVERSE
 - DATASTREAM 188
 - FEPI CONVERSE FORMATTED 194
 - FEPI RECEIVE DATASTREAM 213
 - FEPI RECEIVE FORMATTED 218
- formatted data 144
- environmental requirements 7
- error handling 166
 - application programming
 - commands 178
 - bad command sequencing 153
 - bypass by user exit 167
 - CONVERSE 148
 - general guidance 166
 - list of resources 40, 91
 - operator/system action 167
 - receiving data 145
 - receiving screen-image data 148
 - SEND failure 166
 - sending data
 - key stroke data 144
 - screen-image data 147
 - session loss 166
 - shutdown 167
 - system programming commands 91
 - time-outs 166
 - unknown conversation ID 167
- ESCAPE option
 - FEPI CONVERSE FORMATTED 195
 - FEPI SEND FORMATTED 225
- escape sequences 142, 231
- ESM (external security manager) 30
 - PassTickets 16, 164
- ESMREASON option
 - FEPI REQUEST PASSTICKET 221
- ESMRESP option
 - FEPI REQUEST PASSTICKET 221
- EVENTDATA field
 - start data 229
 - TDQ record 124
- EVENTTYPE field
 - start data 229
 - TDQ record 125
- EVENTVALUE field
 - start data 230
 - TDQ record 125
- example of FEPI configuration 42
- exception conditions
 - application programs 140, 178
 - configuration programs 39
 - general considerations 39, 140
 - system programs 39, 90
- EXCEPTIONQ option
 - FEPI INQUIRE POOL 103
 - FEPI INQUIRE PROPERTYSET 106
 - FEPI INSTALL PROPERTYSET 115
- EXEC CICS command format
 - system programming 90, 178
- EXEC interface block (EIB) 89, 177
- extended data stream
 - device attributes 25, 40
 - getting attributes 144, 147
 - storage requirements 27
 - VTAM configuration 34

- EXTRACT command
 - CONV 199
 - extracting field data 144, 148
 - FIELD 201
 - STSN 205

F

- Feno
 - CEMT INQUIRE FENODE 70
- FENODE option
 - CEMT DISCARD 65
- FEPI
 - applications 157
 - CICS-supplied transactions 65
 - commands 89, 175
 - environmental requirements 7
 - functions and services 11
 - hardware requirements 7
 - how it fits into your system 4
 - installation 29
 - introduction 3
 - operator control 65
 - planning 7, 21
 - programming interface 11
 - resources 14
 - sample programs 237
 - setup
 - sample program 242
 - software requirements 7
 - storage requirements 7
 - system integrity 7
 - translator option 89, 177
- FEPI ADD POOL command 91
- FEPI commands
 - ADD POOL 91
 - ALLOCATE 181
 - ALLOCATE POOL 183
 - AP NOOP 185
 - application programming
 - commands 175
 - CONVERSE DATASTREAM 187
 - CONVERSE FORMATTED 193
 - DELETE POOL 93
 - DISCARD NODELIST 94
 - DISCARD POOL 94
 - DISCARD PROPERTYSET 95
 - DISCARD TARGETLIST 95
 - EXTRACT CONV 199
 - EXTRACT FIELD 201
 - EXTRACT STSN 205
 - FREE 207
 - INQUIRE CONNECTION 96
 - INQUIRE NODE 99
 - INQUIRE POOL 101
 - INQUIRE PROPERTYSET 105
 - INQUIRE TARGET 108
 - INSTALL NODELIST 109
 - INSTALL POOL 111
 - INSTALL PROPERTYSET 113
 - INSTALL TARGETLIST 117
 - ISSUE 209
 - RECEIVE DATASTREAM 213
 - RECEIVE FORMATTED 217
 - REQUEST PASSTICKET 221
 - SEND DATASTREAM 223
 - SEND FORMATTED 225

- FEPI commands (*continued*)
 - SET CONNECTION 118
 - SET NODE 120
 - SET POOL 121
 - SET TARGET 122
 - SP NOOP 123
 - START 227
 - system programming commands 89
 - FEPI configuration
 - coding of programs 38
 - addressing mode 39
 - exception conditions 39
 - system programming commands 38
 - translator option 38
 - debugging programs 39
 - example configuration 42
 - global user exits 50
 - monitoring program
 - sample program 243
 - triggering of 46
 - writing of 46
 - optional functions 40
 - planning 8, 21
 - required functions 40
 - sample configuration 42
 - setup program
 - running of 41
 - sample 242
 - writing operator transactions 53
 - FEPI=YES|NO, system initialization parameter 30
 - FEPIRESOURCE, resource identifier for RACF 30
 - Fepo
 - CEMT INQUIRE FEPOOL 71
 - FEPOOL option
 - CEMT DISCARD 66
 - Fepr
 - CEMT INQUIRE FEPROPSET 73
 - FEPROPSET option
 - CEMT DISCARD 66
 - CEMT INQUIRE FEPROPSET 73
 - Feta
 - CEMT INQUIRE FETARGET 74
 - FETARGET option
 - CEMT DISCARD 66
 - FIELDATTR option
 - FEPI EXTRACT FIELD 201
 - FIELDLOC option
 - FEPI EXTRACT FIELD 202
 - FIELDNUM option
 - FEPI EXTRACT FIELD 202
 - FIELDS option
 - FEPI CONVERSE FORMATTED 195
 - FEPI RECEIVE FORMATTED 218
 - fields, getting data and attributes 144, 147
 - FJOURNALNAME option
 - FEPI INQUIRE POOL 103
 - FEPI INSTALL PROPERTYSET 115
 - FJOURNALNUM option
 - FEPI INQUIRE POOL 103
 - FEPI INQUIRE PROPERTYSET 107
 - FEPI INSTALL PROPERTYSET 115
 - FLENGTH field
 - start data 230
 - FLENGTH option
 - FEPI EXTRACT FIELD 202
 - FEPI RECEIVE DATASTREAM 214
 - FEPI RECEIVE FORMATTED 218
 - FEPI SEND DATASTREAM 223
 - FEPI SEND FORMATTED 225
 - FEPI START 227
 - FMH option
 - FEPI CONVERSE DATASTREAM 188
 - FMHSTATUS option
 - FEPI CONVERSE DATASTREAM 188
 - FEPI RECEIVE DATASTREAM 214
 - FORCE option
 - FEPI FREE 207
 - forced shutdown 58
 - FORMAT field
 - start data 230
 - TDQ record 125
 - FORMAT option
 - FEPI EXTRACT CONV 200
 - FEPI INQUIRE POOL 103
 - FEPI INQUIRE PROPERTYSET 107
 - FEPI INSTALL PROPERTYSET 115
 - formatted data 12
 - performance 171
 - programming 141
 - RECEIVE and EXTRACT field sample program 249
 - SEND and START sample program 247
 - FREE 207
 - FROM option
 - FEPI CONVERSE DATASTREAM 189
 - FEPI CONVERSE FORMATTED 195
 - FEPI SEND DATASTREAM 223
 - FEPI SEND FORMATTED 225
 - FROMCURSOR option
 - FEPI CONVERSE FORMATTED 195
 - FROMFLENGTH option
 - FEPI CONVERSE DATASTREAM 189
 - FEPI CONVERSE FORMATTED 195
 - front-end system 4
 - configuration 33
 - hardware and software requirements 7
 - XRF takeover 59
 - function identifiers, journaling 85
 - function shipping, restrictions on 33
 - functions and services provided by FEPI 11
- ## G
- generic resources, VTAM 26
 - global user exits 81
 - exit points
 - in Front End Programming Interface 81
 - introduction 50
 - XSZARQ
 - exit-specific parameters 83
 - overview 83
 - UEPSZACN parameter 83
 - global user exits (*continued*)
 - XSZBRQ
 - overview 81
 - UEPSZACT parameter 83
 - GOINGOUT status 53
 - good morning message 25, 158
- ## H
- hardware requirements 7
 - HIGHLIGHT option
 - FEPI EXTRACT FIELD 202
 - HOLD option
 - FEPI FREE 207
- ## I
- immediate shutdown 58
 - IMS (Information Management System)
 - considerations for application design 168
 - conversational sample program 251
 - end of session 169
 - message protocols 168
 - recovery 170
 - response mode 169
 - STSN handling 25, 173
 - sample program 254
 - terminal definitions 37
 - unsolicited-data handler 158
 - using MFS 168
 - inbound data
 - 3270 data stream considerations 154
 - data format 232
 - initial data 25, 158
 - journaling 85
 - terminology 11
 - INITIALDATA option
 - FEPI INQUIRE POOL 103
 - FEPI INQUIRE PROPERTYSET 107
 - FEPI INSTALL PROPERTYSET 115
 - INPUTCONTROL option
 - FEPI EXTRACT FIELD 202
 - INQUIRE command
 - CONNECTION 96
 - NODE 99
 - POOL 101
 - use in event handlers 42
 - PROPERTYSET 105
 - TARGET 108
 - INQUIRE, CEMT
 - FECONNECTION 66
 - FENODE 69
 - FEPOOL 71
 - FEPROPSET 73
 - FETARGET 73
 - Inservice
 - CEMT INQUIRE FECONNECTION 67
 - CEMT INQUIRE FENODE 70
 - CEMT INQUIRE FEPOOL 72
 - CEMT INQUIRE FETARGET 75
 - INSERVICE option
 - CEMT SET FECONNECTION 75
 - CEMT SET FENODE 76
 - CEMT SET FEPOOL 77

INSERVICE option (*continued*)
 CEMT SET FETARGET 78
 INSERVICE status 53
 INSTALL command
 NODELIST 109
 POOL 111
 PROPERTYSET 113
 TARGETLIST 117
 Installed
 CEMT INQUIRE
 FECONNECTION 67
 CEMT INQUIRE FENODE 70
 CEMT INQUIRE FEPOOL 72
 CEMT INQUIRE FETARGET 74
 INSTALLED status 55
 INSTALLFAIL event 47
 installing FEPI
 defining security profiles 30
 loading modules in the LPA 29
 overview 29
 planning considerations 8
 RDO definitions 30
 sample programs 239
 starting CICS 31
 updating CICS definitions
 PLTPI list 34
 supplied RDO group,
 DFHFEPI 29
 system initialization parameter,
 FEPI=YES|NO 30
 transient data queues 29
 INSTLSTATUS option
 FEPI INQUIRE CONNECTION 97
 FEPI INQUIRE NODE 100
 FEPI INQUIRE POOL 103
 FEPI INQUIRE TARGET 109
 INSTLSTATUS, resource status 55
 integrity of FEPI system 7
 interactive problem control system
 (IPCS) 127
 INTO option
 FEPI CONVERSE
 DATASTREAM 189
 FEPI CONVERSE FORMATTED 195
 FEPI EXTRACT FIELD 202
 FEPI RECEIVE DATASTREAM 214
 FEPI RECEIVE FORMATTED 218
 INVITE option
 command sequence 150
 FEPI SEND DATASTREAM 223
 IPCS (interactive problem control
 system) 127
 ISC (intersystem communication) 4
 hardware requirements 7
 ISSUE 209
 sending SNA commands 175
 ISTINCLM, LOGON mode table 35

J

journaling 17, 85
 use of property set for 25, 40

K

key stroke and screen-image
 applications 12
 CONVERSE 148
 data formats 231
 extracting field data 148
 general sequence of commands 141
 multiple attentions 145
 performance considerations 171
 receiving field-by-field 144
 receiving screen-image data 147
 sample programs
 key stroke converse 246
 screen image RECEIVE and
 EXTRACT 249
 screen image SEND and
 START 247
 sending key stroke data 142
 sending screen-image data 146
 writing 141
 KEYSTROKES option
 FEPI CONVERSE FORMATTED 195
 FEPI SEND FORMATTED 226

L

Lacqcode
 CEMT INQUIRE
 FECONNECTION 69
 CEMT INQUIRE FENODE 70
 LACQCODE option
 resource status 55
 LASTACQCODE option
 FEPI INQUIRE CONNECTION 97
 FEPI INQUIRE NODE 100
 LINES option
 FEPI CONVERSE FORMATTED 195
 FEPI RECEIVE FORMATTED 218
 Link Pack Area (LPA), loading FEPI
 modules into 29
 list of resources
 benefits of using 56
 errors 40, 91
 list processing 91
 LOGON mode table, VTAM 35
 LPA (Link Pack Area), loading FEPI
 modules into 29

M

magnetic stripe reader, sending key
 stroke data to 143
 managing sessions, use of property set
 for 25, 40
 manipulative keys 143, 231
 MAXLENGTH option
 FEPI CONVERSE
 DATASTREAM 189
 FEPI CONVERSE FORMATTED 195
 FEPI EXTRACT FIELD 202
 FEPI INQUIRE POOL 103
 FEPI INQUIRE PROPERTYSET 107
 FEPI INSTALL PROPERTYSET 115
 FEPI RECEIVE DATASTREAM 214
 FEPI RECEIVE FORMATTED 218

MDT (modified data tag) setting 146,
 231
 MDT option
 FEPI EXTRACT FIELD 202
 Message Format Services (MFS) 168
 message protocols (IMS) 168
 messages
 format of FEPI messages 131
 handling unexpected events 22
 IMS protocols 168
 produced during XRF takeover 59
 resynchronizing with STSN 173
 MFS (Message Format Services) 168
 mode table, VTAM 35
 modified data tag (MDT) setting 146,
 231
 module identifiers, journaling 85
 monitoring program
 handling CLSDST(PASS) 49
 sample program 243
 triggering of 46
 writing of 46
 monitoring, CICS
 performance class records
 FEPI-related fields 56
 MRO (multiregion operation)
 AOR considerations for FEPI 33
 general considerations 4
 MSGJRNL option
 FEPI INQUIRE POOL 104
 FEPI INQUIRE PROPERTYSET 107
 FEPI INSTALL PROPERTYSET 115
 multiple attentions 145
 MVS/ESA
 Integrity Programming
 Announcement 7

N

naming conventions
 nodes 22, 34
 pools 24
 property sets 26
 sample programs 12
 targets 22
 network availability 35
 node
 acquiring and releasing 53
 definition of 14
 determining contents of a pool 53
 INQUIRE NODE command 99
 name restrictions 34
 number of nodes 23
 sample node lists 44
 sample program 242
 SET NODE command 120
 storage requirements 26
 Node
 CEMT INQUIRE
 FECONNECTION 67
 CEMT INQUIRE FENODE 69, 70
 NODE field
 start data 230
 TDQ record 125
 NODE option
 CEMT SET FECONNECTION 76
 FEPI EXTRACT CONV 200

NODE option (*continued*)
 FEPI INQUIRE CONNECTION 97
 FEPI INQUIRE NODE 100
 FEPI SET CONNECTION 119
 FEPI SET NODE 120

NODELIST option
 FEPI ADD 92
 FEPI DELETE 93
 FEPI DISCARD NODELIST 94
 FEPI INSTALL NODELIST 110
 FEPI INSTALL POOL 112
 FEPI SET CONNECTION 119
 FEPI SET NODE 120

nodename option
 CEMT SET FENODE 76

NODENUM option
 FEPI ADD 92
 FEPI DELETE 93
 FEPI DISCARD NODELIST 94
 FEPI INSTALL NODELIST 110
 FEPI INSTALL POOL 112
 FEPI SET CONNECTION 119
 FEPI SET NODE 120

normal shutdown 57

Notinstalled
 CEMT INQUIRE
 FECONNECTION 67
 CEMT INQUIRE FENODE 70
 CEMT INQUIRE FEPOOL 72
 CEMT INQUIRE FETARGET 75

NOTINSTALLED status 55

O

one-out one-in conversational applications
 application design 159
 sample program 251

operator control
 commands 65
 operator/system action error 167
 transactions, user-written 53
 VTAM commands 78

operator/system action error 167

order of FEPI commands 141, 150

organizing pools 24

organizing property sets 25

outbound data
 3270 data stream considerations 154
 data formats 231
 journaling 85
 terminology 11

OUTLINE option
 FEPI EXTRACT FIELD 202

Outservice
 CEMT INQUIRE
 FECONNECTION 67
 CEMT INQUIRE FENODE 70
 CEMT INQUIRE FEPOOL 72
 CEMT INQUIRE FETARGET 75

OUTSERVICE option
 CEMT SET FECONNECTION 76
 CEMT SET FENODE 77
 CEMT SET FEPOOL 77
 CEMT SET FETARGET 78

OUTSERVICE status 53

P

padding of FEPI sessions 37

PASS option
 FEPI FREE 207

pass-through
 contention state handling 152
 problem with received data 152
 sample program 250

PASSCONVID option
 FEPI ALLOCATE PASSCONVID 181
 getting ownership of conversations 163

passing conversations 163

PASSTICKET option
 FEPI REQUEST PASSTICKET 221

PassTickets, for signon security 16, 164

PASSWORDLIST option
 FEPI INSTALL NODELIST 110

PERFORM STATISTICS RECORD
 command 57

performance
 application programs 55
 formatted data 171
 key stroke and screen-image applications 171
 of a CICSplex
 using VTAM generic resources 26

optimization through application design 171

tuning using CICS monitoring data 56

tuning using CICS statistics data 56

performance class monitoring records 56

persistent sessions, VTAM
 use of with FEPI 62

PI/I language
 sample programs 237

PL/I language
 copybook 39, 178

planning
 back-end applications 21
 configuration 8, 21
 general considerations 7, 21
 grouping of connections, for functional purposes 23
 handling special events 22
 installation 8
 journaling requirements 22
 names of nodes and targets 22
 number of nodes 23
 operator control requirements 22
 organizing pools 24
 organizing property sets 25
 pools
 using for control purposes 23
 using for functional purposes 23

signon and signoff procedures 22

storage 26

PLT (program list table) 34, 41
 post initialization (PLTPI) 34

PLTPI (program list table post initialization)
 configuring CICS for FEPI 34
 running s 41

Pool
 CEMT INQUIRE
 FECONNECTION 67

Pool (*continued*)

CEMT INQUIRE FEPOOL 71

POOL field
 start data 230
 TDQ record 125

POOL option
 FEPI ADD 92
 FEPI ALLOCATE POOL 183
 FEPI CONVERSE
 DATASTREAM 189
 FEPI CONVERSE FORMATTED 195
 FEPI DISCARD POOL 95
 FEPI EXTRACT CONV 200
 FEPI INQUIRE CONNECTION 98
 FEPI INQUIRE POOL 104
 FEPI INSTALL POOL 112
 FEPI SET POOL 121

POOLLIST option
 FEPI SET POOL 121

POOLNUM option
 FEPI SET POOL 121

pools
 connections in 53
 definition of 14
 determining contents of a pool 53
 INQUIRE POOL command 101
 INSTALL POOL command 111
 name restrictions 24
 organizing pools 24
 sample configuration 42
 sample program 242
 SET POOL command 121
 storage requirements 26
 transient data queues 30, 46
 using for control reasons 23
 using for functional reasons 23

POSITION option
 FEPI EXTRACT FIELD 202

prerequisites, hardware and software 7

problem determination
 abends 132
 debugging 127
 functions provided by FEPI 17
 handling unexpected events 47
 messages 131
 reporting problems to IBM 133
 shutdown not proceeding 58
 trace 130
 using CICS dumps 129
 using FEPI dumps 127

product tape 8

program, for FEPI resources
 sample program 242

property set
 data handling 25, 40
 definition of 14
 device attributes 25, 40
 DISCARD PROPERTYSET
 command 95
 INQUIRE PROPERTYSET
 command 105
 INSTALL PROPERTYSET
 command 113
 journaling 25, 40
 name restrictions 26
 organizing 25
 sample configuration 45

- property set (*continued*)
 - sample program 242
 - session management 25, 40
 - storage requirements 26
 - unexpected events 25, 40
- PROPERTYSET option
 - FEPI DISCARD PROPERTYSET 95
 - FEPI INQUIRE POOL 104
 - FEPI INQUIRE PROPERTYSET 107
 - FEPI INSTALL POOL 112
 - FEPI INSTALL PROPERTYSET 116
- PROTECT option
 - FEPI EXTRACT FIELD 202
- PS option
 - FEPI EXTRACT FIELD 202
- PS/55
 - FEPI device type 103
 - TYPETERM 37
- pseudoconversational applications
 - application design 160
 - sample program 252

Q

- query, device 158

R

- RACF (Resource Access Control Facility)
 - general security considerations 30
- RDO (resource definition online)
 - definitions for sample programs 238
 - updating CICS definitions for FEPI 29
 - XRF considerations 59
- RECEIVE command
 - completion 144, 151
 - DATASTREAM 213
 - error handling 145, 153
 - FORMATTED 217
- receiving data
 - data stream applications 150
 - field-by-field 144
 - screen-image 147
- reference section
 - application programming 175
 - operator commands 65
 - system programming 89
- RELEASE option
 - FEPI FREE 207
- Released
 - CEMT INQUIRE
 - FECONNECTION 68
 - CEMT INQUIRE FENODE 70
- RELEASED option
 - CEMT SET FECONNECTION 76
 - CEMT SET FENODE 77
- RELEASED resource status 53
- RELEASING, resource status 53
- REMFLENGTH option
 - FEPI CONVERSE
 - DATASTREAM 189
 - FEPI RECEIVE DATASTREAM 214
- reporting FEPI problems to IBM 133
- REQUEST PASSTICKET command 221

- request unit (RU)
 - receiving 151
- requirements, hardware and software 7
- resources
 - benefits of using list of resources 56
 - configuring 41
 - definitions 14
 - diagram showing relationship 14
 - sample configuration 42
 - status 53
- RESP2 values
 - application programming
 - commands 178
 - system programming commands 90
 - table of 260
- response mode 169
- responses
 - DRx responses 174
- RESPSTATUS option
 - FEPI CONVERSE
 - DATASTREAM 189
 - FEPI CONVERSE FORMATTED 195
 - FEPI RECEIVE DATASTREAM 214
 - FEPI RECEIVE FORMATTED 218
- restriction on use of DFH
 - in node names 34
 - in pool names 24
 - in property set names 26
- return codes
 - application programming 178
 - system programming 90
 - VTAM 55
- RU option
 - FEPI CONVERSE
 - DATASTREAM 189
 - FEPI RECEIVE DATASTREAM 214

S

- sample FEPI configuration 42
- sample programs 237, 242
 - 3270 data stream pass-through 250
 - begin session 245
 - CICS back-end 240
 - end-session handler 250
 - FEPI configuration 42
 - IMS back-end 241
 - installing of 239
 - introduction 12
 - key stroke CONVERSE 246
 - monitor and unsolicited data handler 243
 - naming conventions 12
 - one-out one-in 251
 - program descriptions and code 237
 - pseudoconversational 252
 - screen image RECEIVE and EXTRACT FIELD 249
 - screen image SEND and START 247
 - SLU P one-out one-in 251
 - SLU P pseudoconversational 252
 - STSN handler 254
 - using the samples 240
- sample resource definitions 45
- screen-image interface 13
 - data formats 231

- screen-image interface (*continued*)
 - RECEIVE and EXTRACT field sample program 249
 - SEND and START sample program 247
- secondary logical unit (SLU)
 - terminals supported by FEPI 16
- security
 - command-level 17, 30
 - handling violations with access program 157
 - restricting access to system programming commands 30
 - signoff 22
 - signon 16, 22, 157, 164
 - using PassTickets 16, 164
 - using RACF 30
- SEND command
 - DATASTREAM 223
 - error handling 166
 - errors 144, 145, 153
 - FORMATTED 225
- sending data
 - data stream applications 153
 - key stroke data 142
 - screen image 146
- sense data 167
- SENSEDATA option
 - FEPI EXTRACT CONV 200
 - FEPI ISSUE 210
- SEQNUMIN option
 - FEPI ALLOCATE POOL 183
 - FEPI CONVERSE
 - DATASTREAM 189
 - FEPI EXTRACT STSN 205
 - FEPI RECEIVE DATASTREAM 214
 - FEPI SEND DATASTREAM 223
- SEQNUMOUT option
 - FEPI ALLOCATE POOL 183
 - FEPI CONVERSE
 - DATASTREAM 189
 - FEPI EXTRACT STSN 205
 - FEPI RECEIVE DATASTREAM 214
 - FEPI SEND DATASTREAM 223
- sequence number handling 173
- sequence of FEPI commands 141, 150
- SERVSTATUS option
 - FEPI ADD 92
 - FEPI INQUIRE CONNECTION 98
 - FEPI INQUIRE NODE 100
 - FEPI INQUIRE POOL 104
 - FEPI INQUIRE TARGET 109
 - FEPI INSTALL NODELIST 110
 - FEPI INSTALL POOL 112
 - FEPI INSTALL TARGETLIST 117
 - FEPI SET CONNECTION 119
 - FEPI SET NODE 120
 - FEPI SET POOL 121
 - FEPI SET TARGET 122
- SERVSTATUS, resource status 53
- session
 - loss of 166
 - management using property sets 25, 40
 - pacing values 37
 - parameters, selection of 35
- SESSION event 47

SESSIONFAIL event 48
 SESSIONLOST event 47
 SET command
 CONNECTION 118
 NODE 120
 POOL 121
 TARGET 122
 SET, CEMT
 FECONNECTION 75
 FENODE 76
 FEPOOL 77
 FETARGET 78
 SETFAIL event 47
 shutdown
 error handling 167
 of CICS 57
 of FEPI 58
 signon security 16, 22, 157, 164
 SIT (system initialization table)
 SIT parameter, FEPI=YES|NO 30
 SIZE option
 FEPI EXTRACT FIELD 202
 SLU (secondary logical unit)
 terminals supported by FEPI 16
 SLU P
 device attributes 25
 one-out one-in sample program 251
 pseudoconversational sample
 program 252
 SLU P connections
 sample configuration 45
 SLU P mode
 data stream applications 155
 SLU2
 device attributes 25
 IMS recovery 170
 SLU2 24 x 80 connections to CICS
 sample configuration 45
 SLU2 24 x 80 connections to IMS
 sample configuration 45
 SLU2 mode
 data stream applications 154
 SMP/E (System Modification
 Program/Extended)
 installing FEPI 8
 SNA (Systems Network Architecture)
 sending commands 175
 software requirements 7
 special keys 143, 231
 specialized functions
 DRx responses 174
 SNA commands 175
 STSN 173
 START command 227
 failure during shutdown 167
 start data 160, 229
 started tasks 160
 State
 CEMT INQUIRE
 FECONNECTION 68
 STATE option
 FEPI INQUIRE CONNECTION 98
 STATE, resource status 55
 statistics, CICS
 FEPI-related
 COLLECT STATISTICS
 command 56

statistics, CICS (*continued*)
 FEPI-related (*continued*)
 PERFORM STATISTICS RECORD
 command 57
 storage planning 26
 stripe reader, sending key stroke
 data 143
 STSN (set and test sequence number) 15
 general considerations 173
 sample program 254
 STSN handler
 defining to FEPI 40
 STSN option
 FEPI INQUIRE POOL 104
 FEPI INQUIRE PROPERTYSET 107
 FEPI INSTALL PROPERTYSET 116
 syncpoints, use of in FEPI 158
 system programming commands 38, 89
 SZ, dump control keyword 127

T

target
 definition of 14
 determining contents of a pool 53
 INQUIRE TARGET command 108
 naming conventions 22
 sample program 242
 sample target lists 44
 SET TARGET command 122
 storage requirements 26

Target
 CEMT INQUIRE
 FECONNECTION 67

TARGET field
 start data 230
 TDQ record 125

TARGET option
 CEMT SET FECONNECTION 76
 FEPI ALLOCATE POOL 183
 FEPI CONVERSE
 DATASTREAM 190
 FEPI CONVERSE FORMATTED 196
 FEPI EXTRACT CONV 200
 FEPI INQUIRE CONNECTION 99
 FEPI INQUIRE TARGET 109
 FEPI SET CONNECTION 119
 FEPI SET TARGET 123

TARGETLIST option
 FEPI ADD 92
 FEPI DELETE 93
 FEPI DISCARD TARGETLIST 96
 FEPI INSTALL POOL 112
 FEPI INSTALL TARGETLIST 117
 FEPI SET CONNECTION 119
 FEPI SET TARGET 123

TARGETNUM option
 FEPI ADD 92
 FEPI DELETE 93
 FEPI DISCARD TARGETLIST 96
 FEPI INSTALL POOL 112
 FEPI INSTALL TARGETLIST 118
 FEPI SET CONNECTION 119
 FEPI SET TARGET 123

tasks, started 160
 temporary conversation 163

TERMINAL option
 FEPI START 227

terminal
 back-end definitions 37
 simulated terminal usage 160
 storage requirements 26
 VTAM logon mode table entries 35
 XRF environment 59

time-outs, error handling 166

TIMEOUT option
 FEPI ALLOCATE POOL 183
 FEPI CONVERSE
 DATASTREAM 190
 FEPI CONVERSE FORMATTED 196
 FEPI RECEIVE DATASTREAM 215
 FEPI RECEIVE FORMATTED 218
 FEPI START 227

TOCURSOR option
 FEPI CONVERSE FORMATTED 196

TOFLENGTH option
 FEPI CONVERSE
 DATASTREAM 190
 FEPI CONVERSE FORMATTED 196

trace points 130

trademarks 266

transactions
 CETR 131
 CZBC 240
 CZBI 241
 CZPA 252
 CZPS 251
 CZQS 254
 CZTD 250
 CZTK 246
 CZTR 249
 CZTS 247
 CZUC 245
 CZUU 250
 CZUX 243
 CZXS 242

TRANSID option
 FEPI START 227

transient data queues
 command errors 39
 CSZL, for FEPI messages 29
 CSZX, for unexpected events 29
 defining to CICS 29
 handling 46
 planning 23
 pool-specific 30, 46
 records 123
 sample program 243
 unexpected event reporting 46

translator options
 application programming
 commands 177
 FEPI option 38, 177
 system programming commands 89

TRANSPARENCY option
 FEPI EXTRACT FIELD 203

TYPETERMs for CICS back-end
 systems 37

U

UEPSZACN, exit-specific parameter for
 XSZARQ 83

- UEPSZACT, exit-specific parameter for XSZBRQ 83
- unexpected events 29
 - in CSZX TD queue 47
 - in pool-specific TD queue 22, 47
 - using event handlers 42
 - using property set for 25, 40
- unknown conversation ID, error handling 167
- UNSOLDATA option
 - FEPI INQUIRE POOL 104
 - FEPI INQUIRE PROPERTYSET 107
 - FEPI INSTALL PROPERTYSET 116
- UNSOLDATAACK option
 - FEPI INQUIRE POOL 104
 - FEPI INQUIRE PROPERTYSET 107
 - FEPI INSTALL PROPERTYSET 116
- unsolicited data-handler
 - sample program 243
- unsolicited-data handler 15
 - application design 158
 - defining to FEPI 25, 40
- UNTILCDEB option
 - FEPI CONVERSE
 - DATASTREAM 190
 - FEPI RECEIVE DATASTREAM 215
- USERDATA field
 - start data 230
- USERDATA option
 - FEPI INQUIRE CONNECTION 99
 - FEPI INQUIRE NODE 101
 - FEPI INQUIRE POOL 104
 - FEPI INQUIRE TARGET 109
 - FEPI SET CONNECTION 119
 - FEPI SET NODE 121
 - FEPI SET POOL 122
 - FEPI SET TARGET 123
 - FEPI START 227

V

- VALIDATION option
 - FEPI EXTRACT FIELD 203
- VALUE option
 - FEPI ISSUE 210
- VS COBOL II 239
- VTAM (Virtual Telecommunications Access method)
 - persistent sessions 62
- VTAM (Virtual Telecommunications Access Method)
 - APPL statement 34
 - BIND during XRF takeover 59
 - CLSDST(PASS) 48
 - commands 78
 - configuration of 34
 - DISPLAY command 78
 - DISPLAY SESSIONS command 79
 - FEPI commands 12, 13
 - generic resources 26
 - ISTINCLM, supplied mode table 35
 - LOGON mode table 35
 - minor nodes, sample
 - configuration 45
 - program-to-program support 4
 - releasing a connection 54
 - session pacing values 37

- VTAM (Virtual Telecommunications Access Method) (*continued*)
 - session parameters 35
 - VARY command 78
 - VARY TERM command 79
 - XRF considerations 59
- VTAM persistent sessions
 - use of with FEPI 62

W

- Waitconvnum
 - CEMT INQUIRE
 - FECONNECTION 69
 - CEMT INQUIRE FEPOOL 72
- WAITCONVNUM option
 - event handlers 42
 - FEPI INQUIRE CONNECTION 99
 - FEPI INQUIRE POOL 104
- WAITCONVNUM resource status 55
- WCC (write control character)
 - handling 154
- workload balancing
 - in a CICSplex
 - using VTAM generic resources 26
- writing application programs 139

X

- XLT (transaction list table)
 - application programming 167
 - operations 57
- XRF (extended recovery facility) 58
 - back-end system configuration 37
 - FEPI resource definition 59
 - takeover of back-end CICS
 - effect on applications 61
 - effect on FEPI connections 61
 - takeover of FEPI CICS
 - effect on alternate CICS 60
 - effect on back-end terminals 60
 - effect on back-end transactions 59
 - varying setup resources 41
 - VTAM considerations 59
- XSZARQ, global user exit 81
 - overview 83
 - UEPSZACN parameter 83
- XSZBRQ, global user exit 81
 - overview 81
 - UEPSZACT parameter 83

Readers' Comments — We'd Like to Hear from You

CICS Transaction Server for z/OS
Version 4 Release 1
Front End Programming Interface User's Guide

Publication No. SC34-7027-02

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:

- Send your comments to the address on the reverse side of this form.
- Send a fax to the following number: +44 1962 816151
- Send your comments via email to: idrctf@uk.ibm.com

If you would like a response from IBM, please fill in the following information:

Name

Address

Company or Organization

Phone No.

Email address



Fold and Tape

Please do not staple

Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM United Kingdom Limited
User Technologies Department (MP095)
Hursley Park
Winchester
Hampshire
United Kingdom
SO21 2JN

Fold and Tape

Please do not staple

Fold and Tape



SC34-7027-02

