Connect:Enterprise®
for z/OS

Application Agents and User Exits Guide

Version 1.4

Sterling Commerce
An IBM Company

Connect:Enterprise for z/OS Application Agents and User Exits Guide

Version 1.4

First Edition

Sterling Commerce, Inc.

4600 Lakehurst Court Dublin, OH 43016-2000 *
614/793-7000

Contents

Chapter 1

Overview of Connect:Enterprise Application Agents

This chapter provides an overview of the Connect:Enterprise for z/OS application agents and summarizes how Connect:Enterprise uses application agents. The next two chapters describe creating the rule sets that control application agents and implementing application agents, respectively, so the three chapters should be used together.

An application agent is an interface that customizes the execution and automation of your Connect:Enterprise environment. Application agents are event driven and controlled by a set of user-specified rules. Application agents are dispatched just prior to calling the corresponding user exit, if one is implemented. Application agents and user exits are independent of one another. For example, you can use the End of Batch application agent with or without implementing the End of Batch user exit. Similarly, you can use the End of Batch user exit with or without implementing the End of Batch application agent.

Application agents are controlled by rule sets. You define a rule set for each type of application agent to specify the actions to perform based on the characteristics of the event that causes Connect:Enterprise to initiate an application agent request. A rule set consists of statements and instructions. The RULE statement defines the name of the rule that has associated instructions to execute. The SELECT statement specifies the selection criteria and points to up to eight rules to process if a match is made for an application agent request. Instructions specify the operations to perform when the selection criteria is met.

Connect:Enterprise provides the following application agents:

- Console (CON). This agent is invoked whenever Connect:Enterprise issues a specific message.
- End of Batch (EOB). This agent is invoked whenever an online batch is collected successfully.
- Logging (LOG). This agent is invoked whenever a log record is written.
- Scheduler (SCH). This agent is invoked whenever a specific time of day is reached.
- Wake Up Terminate (WKT). This agent is invoked whenever a wake up acknowledgment is sent by CICS.

Console Application Agent

A Console Application Agent is triggered in the following instances:

- When a $$INVOKE console command specifies RULES=CON,TEXT='msgid …'.
- When an application agent rule contains a MESSAGE instruction that specifies the parameter CONEVENT=YES.
- When a MSG01 specified by a SELECT statement in the RULESCON member matches the message ID portion of a WTO issued by the Connect:Enterprise Main Task. However, multiple classes of these messages cannot trigger Application Agents, including:
  - Messages that are not issued by STCC03 and error messages that are generated by STCC03 while another message is being processed. If these messages are specified in a SELECT MSG01 parameter, they are detected during RULES scanning and rejected with message CMR115I in SYSPRINT.
    - CMB000I
    - CMB001I
    - CMB011E
    - CMB220E
    - CMB333I
    - CMB400E
    - CMB595I
    - CMB596I
    - CMB597I
    - CMB598I
    - CMB599I
    - CMB703E
    - CMB996I
    - CMB997I
    - CEDEBUG

    Note: See the *Connect:Enterprise for z/OS Messages and Codes Guide* for more information on all messages.

  - Messages that are only issued once at startup but before the Process Router initialization is complete. If these messages are specified in a SELECT MSG01 parameter, they are detected during RULES scanning and rejected with message CMR115I in SYSPRINT.
    - CMB277I
    - CMB278I
    - CMB279I
    - CMB388E

- Messages that are issued before Process Router initialization is complete and that can be reissued later. For these messages, STCC03 issues message CMB997I (ENVIRONMENT NOT ACTIVE) to the Main Task's JESMSGLG when the too-early condition is encountered. Otherwise, these messages are eligible for CONSOLE agent processing.
  - CMB002I
  - CMB006I
  - CMB096I
  - CMB126E
  - CMB170I
  - CMB171I
  - CMB219I
  - CMB349I
  - CMB353I
  - CMB2101I

When an application agent is triggered by a $$INVOKE console command or by a MESSAGE instruction that specifies CONEVENT=YES, the Console application agent checks the message against all Console application agent SELECT statements for a match. If no match is found, no further processing is done for that message. Unless the Console application agent rules trace is turned on, there is no indication of the failure to find a match.

By defining the Console application agent rules, you can specify different actions to take based on certain messages. Each rule defined can instruct the application agent to process one or more of the following instructions:

- COMMAND—issue a console command
- EXECUTE—execute a user-specified program
- MESSAGE—issue a console message
- NOP—issue a no operation instruction to selectively exclude log records from processing
- SNMPTRAP—issue an SNMP trap (a message that reports a problem or a significant event, formatted and encoded as defined in RFC 1442, and sent as a UDP datagram) to an IP address
- SUBMIT—submit a job to the internal reader

Console Rules

The following RULE statements and instructions are valid for the Console application agent. Statements and instructions are highlighted in bold text. Parameters are indented under the related instruction, and parameter default values are underlined.

```
RULE NAME=xxxxxxxx
  COMMAND TEXT='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    ERROR=CONTINUE|QUIT
  EXECUTE PROGRAM=xxxxxxxx,
    ERROR=CONTINUE|QUIT
  MESSAGE TEXT='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    ROUTCODE=(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16),
    DESCCODE=(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16),
    CONEVENT=YES|NO,
    ERROR=CONTINUE|QUIT
  NOP
  SNMPTRAP TEXT='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    IPADDR=host-name|nnn.nnn.nnn.nnn,
    PORT=nnnnn|162,
    GROUP1=ALARM|STATUS,
    GROUP2=nnnn|9999,
    ERROR=CONTINUE|QUIT
  SUBMIT MEMBER=xxxxxxxx,
    ERROR=CONTINUE|QUIT
SELECT RULE=(xxxxxxxx,xxxxxxxx…),
  MSG01=(xxxxxxxxx,xxxxxxxx,….),
  MSG02=(xxxxxxxxx,xxxxxxxx,….),
  …
  MSG96=(xxxxxxxxx,xxxxxxxx,….)
```

Console Rules Example

The following is an example of Console rules along with an explanation of what each line means.

```
01 RULE NAME=RULE1
02   EXECUTE PROGRAM=PGM1,ERROR=CONTINUE
03   COMMAND TEXT='F ENTPRS,$$CONNECT LISTNAME=&MSG02'
04 RULE NAME=RULE2
05   MESSAGE TEXT='CON Echo: &MSG',ROUTCODE=(15)
06   SUBMIT MEMBER=JOB25
07 RULE NAME=RULE3
08   NOP
09 RULE NAME=RULE4
10   SNMPTRAP  TEXT=' C:E ALARM 1234: &MSG',IPADDR=33.44.55.66,GROUP1=ALARM,GROUP2=1234
11 SELECT RULE=(RULE1,RULE4),MSG01=(CMB331E,CMB332E)
12 SELECT RULE=RULE2,MSG01=MYMSG,MSG02=(TEST3,TEST9)
13 SELECT RULE=(RULE4,RULE3,RULE1),MSG01=(GREEK),MSG02=(LETTER),MSG03=(ALPHA,GAMMA,OMEGA)
```

Note: The line numbers listed in the preceding example are not displayed in your rules. They are for illustrative purposes only.

The following table outlines detailed information for each line in the preceding example:

Line Number	Definition
01	Defines a rule named RULE1.
02	Specifies an EXECUTE instruction to call user program PGM1. If the program returns with a non-zero return code, execute the next instruction.
03	Specifies a COMMAND instruction to issue the $$CONNECT console command indicated in the quotes.
04	Defines a rule named RULE2.
05	Specifies the MESSAGE instruction is to write the message contained inside the single quotes. The triggering message is substituted into &MSG, and the resulting text is truncated if necessary to 125 bytes. The ROUTCODE operand specifies a user-specified destination (route code 15).
06	Specifies a SUBMIT instruction. The member name JOB25 is submitted to the Connect:Enterprise internal reader. JOB25 is retrieved from the data set or data sets referenced by the //RULESJCL DD statement in the Connect:Enterprise JCL.
07	Defines a rule named RULE3.
08	Specifies the NOP instruction. No operation is performed.
09	Defines a rule named RULE4.
10	Specifies the SNMPTRAP instruction. A trap containing the text is sent to IP address 33.44.55.66 port 162 (the default port), using GROUP1=ALARM (1) and GROUP2=1234.
11-13	Indicates the selection criteria to use in determining which rules to process for specific Console agent events.

End of Batch Application Agent

The End of Batch application agent is invoked whenever an online batch is collected successfully. You can also invoke the application agent manually by using the $$INVOKE operator command and specifying a single batch number or a range of batch numbers to process.

By defining the End of Batch application agent rules, you can specify different actions to occur based on certain characteristics of the batch. Each rule defined can instruct the application agent to process one or more of the following instructions:

- COMMAND—issue a console command
- EXECUTE—execute a user-specified program
- MESSAGE—issue a console message
- NOP—issue a no operation instruction to selectively exclude batches from processing

- ROUTE—sign on to Connect:Direct, submit a Process to Connect:Direct; usually to extract the batch from Connect:Enterprise, and sign off from the Connect:Direct node
- SNMPTRAP—issue an SNMP trap (a message that reports a problem or a significant event, formatted and encoded as defined in RFC 1442, and sent as a UDP datagram) to an IP address
- STATFLG—set batch status flags on or off
- SUBMIT—submit a job to the internal reader
- WAKEUP—issue a wakeup notification to inform CICS of the batch just collected

End of Batch Rules

The following RULE statements and instructions are valid for the End of Batch application agent. Statements and instructions are highlighted in bold text. Parameters are indented under the related instruction, and parameter default values are underlined.

```
RULE NAME=xxxxxxxx
COMMAND TEXT='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    ERROR=CONTINUE|QUIT
  EXECUTE PROGRAM=xxxxxxxx,
    ERROR=CONTINUE|QUIT
  MESSAGE TEXT='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    ROUTCODE=(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16),
    DESCCODE=(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16),
    CONEVENT=YES|NO,
    ERROR=CONTINUE|QUIT
  NOP
  ROUTE PROC=membername
PROCDSN='filename[(membername)]',
PNODE='primary_nodename',
PNODEID=(userid,password),
SNODE='secondary_nodename',
SNODEID=(userid,password),
TODSN='filename[(membername)]',
FTYPE=filetype,
```

Continued

```
PROFDSN='filename[(membername)]',
NETMAP='filename[(membername)]',
NEWNAME='alias_process_name',
PRTY=nn,
CLASS=nn,
CASE=YES|NO,
NOTIFY=userid|%USER,
SIGNONUID=(userid,password),
LOCAPPL=ISPF_APPLID_prefix,
LOGMODE=ISPF_logmode,
MBAPPL=APPC_APPL_ID,
BUFSIZE=nnnnn,
MAILBOXUID=(userid,password),
TRANSPORT=SNA|TCP,nnnnn,nnn.nnn.nnn.nnn
ERROR=CONTINUE|QUIT
  SNMPTRAP TEXT='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
IPADDR=host-name|nnn.nnn.nnn.nnn,
PORT=nnnnn|162,
GROUP1=ALARM|STATUS,
GROUP2=nnnn|9999,
ERROR=CONTINUE|QUIT
STATFLG ONFLAGS=(REQUESTABLE,DELETED,TRANSMITTED,EXTRACTED,MULTXMIT),
   OFFFLAGS=(REQUESTABLE,DELETED,TRANSMITTED,EXTRACTED,MULTXMIT),
   ERROR=CONTINUE|QUIT
  SUBMIT MEMBER=xxxxxxxx,
    ERROR=CONTINUE|QUIT
  WAKEUP CICSDEFN=TRANSACTION|PROGRAM,
    CICSSYSID=xxxx,
    CICSPGMNM=xxxxxxxx,
    CICSTRANID=xxxx,
    CICSTERMID=xxxx,
    CICSUSER=xxxxxxxx,
    ERROR=CONTINUE|QUIT
SELECT RULE=(xxxxxxxx,xxxxxxxx…),
    BATCHID='xxxx....xxxx'|"yyyy....yyyy",
    CASE_SENSITIVE=YES|NO
    ID=xxxxxxxx,
    LINE=xxxxxxxx,
    REMOTE=xxxxxxxx,
    STATOR=(ADDED,BSC,COLLECTED,DELETED,EBCDIC,EXTRACTED,
    WILD_CARD=BID
    WILD_CARD_MULTI_CHAR=*|x[xxxxxxx]
    WILD_CARD_SINGLE_CHAR=%|x[xxxxxxx]
  FILE_STUCTURE,FTP,INCOMPLETE,MULTXMIT,
  NONTRANSMITTABLE,REQUESTABLE,SNA,SSL,
  TRANSPARENT,TRANSMITTED,UNEXTRACTABLE)
  STATUS=(ADDED,BSC,COLLECTED,DELETED,EBCDIC,EXTRACTED,
  FILE_STUCTURE,FTP,INCOMPLETE,MULTXMIT,
  NONTRANSMITTABLE,REQUESTABLE,SNA,SSL,
  TRANSPARENT,TRANSMITTED,UNEXTRACTABLE)
```

End of Batch Rules Example

The following is an example of End of Batch rules along with an explanation of what each line means. The line numbers listed in this example are not displayed in your rules. They are for illustrative purposes only.

Note: To ensure accurate processing results, SELECT statements containing more detailed selection criteria *must precede* SELECT statements containing general criteria, as shown in the following example.

```
                                                                        72
01 RULE NAME=RULE1
02   EXECUTE PROGRAM=PGM1,ERROR=QUIT
03   EXECUTE PROGRAM=PGM2,ERROR=CONTINUE
04   COMMAND TEXT='F &STCNAME,$$CONNECT LISTNAME=ACCTING ID=&IDFIELD-
05                BID=&BATCH#'
06 RULE NAME=RULE2
07   SUBMIT MEMBER=JOB25
08   COMMAND TEXT='F ENTPRS,$$CONNECT LISTNAME=SALES'
09 RULE NAME=RULE3
10   SUBMIT MEMBER=JOB03,ERROR=QUIT
11   STATFLG ONFLAGS=(REQUESTABLE,MULTXMIT)
12 WAKEUP CICSDEFN=PROGRAM,CICSYSID=SYS1,CICSPGMNM=PGM085
13 RULE NAME=RULE4
14 WAKEUP CICSDEFN=TRANSACTION,CICSSYSID=SYS1,
15        CICSUSER=USER1,CICSTRANID=TRN1,CICSTERMID=TRM1

16 RULE NAME=RULE5
17   NOP
18 RULE NAME=RULE6
19  SNMPTRAP  TEXT='BATCH#=&BATCH# ID=&IDFIELD BID=&BID24',IPADDR=11.22.33.44
20 SELECT RULE=RULE1,ID=SANFRAN,BATCHID='PAYROLL',REMOTE=BRANCH1
21 SELECT RULE=RULE1,ID=SANFRAN,BATCHID='RECEIVABLES',REMOTE=BRANCH1
22 SELECT RULE=RULE1,ID=SANFRAN,BATCHID='PAYABLES',REMOTE=BRANCH1
23 SELECT RULE=RULE3,ID=NEWYORK,BATCHID="INVENTORY"
24 SELECT RULE=RULE5,ID=CANADA,REMOTE=CANADA01
25 SELECT RULE=RULE4,ID=CANADA,REMOTE=CAN*
26 SELECT RULE=RULE2,ID=SANFRAN
27 SELECT RULE=RULE2,ID=DALLAS
28 SELECT RULE=(RULE6,RULE5),ID=RIPON
29 SELECT RULE=RULE5,ID=*
```

The following table outlines detailed information for each line in the preceding example:

Line Number	Definition
01	Defines a rule named RULE1.
02	Specifies an EXECUTE instruction to call user program PGM1. If the program returns with a nonzero return code, do NOT execute any of the subsequent instructions.
03	Specifies an EXECUTE instruction to call user program PGM2. If the program returns with a nonzero return code, continue with execution of the next instruction.

Line Number	Definition
04	Specifies a COMMAND instruction to issue the $$CONNECT console command to transmit the newly collected batch. The command text is concatenated and continued on the next line. Column 72 indicates continuation of the TEXT parameter.
05	This line is a continuation of the command text from line 04. The Connect:Enterprise started task name, ID, and BID symbolics are resolved when the instruction is executed.
06	Defines a rule named RULE2.
07	Specifies a SUBMIT instruction. The member name JOB25 is submitted to the Connect:Enterprise internal reader (//JESRDR DD). JOB25 is retrieved from the data set or data sets referenced by the //RULESJCL DD statement in the Connect:Enterprise JCL.
08	Specifies a COMMAND instruction to issue the $$CONNECT console command indicated in the quotes.
09	Defines a rule named RULE3.
10	Specifies a SUBMIT instruction. The member name JOB03 is submitted to the Connect:Enterprise internal reader. JOB03 is retrieved from the data set or data sets referenced by the //RULESJCL DD statement in the Connect:Enterprise JCL. If SUBMIT cannot successfully process, do NOT execute any of the subsequent instructions.
11	Specifies a STATFLG instruction to turn on the R (REQUESTABLE) and M (MULTXMIT) batch status flags.
12	Specifies the WAKEUP instruction. CICS Wakeup Initiate processing is started for this batch.
13	Defines a rule named RULE4.
14	Specifies the WAKEUP instruction. CICS Wakeup Initiate processing is started for this batch. The ending comma implies continuation of the WAKEUP instruction.
15	This line is a continuation of the WAKEUP instruction coded on line 13.
16	Defines a rule named RULE5.
17	Specifies the NOP instruction. No operation is performed.
18	Defines a rule named RULE6.
19	Specifies the SNMPTRAP instruction. A trap containing the text is sent to IP address 11.22.33.44. (The default port is 162.)
20-29	Indicates the selection criteria that determines which set of rules to process for specific batches. **Note:** In the SELECT statements on lines 23 and 24 in the preceding example, for a specific match of REMOTE=CANADA01, a different rule is processed than for the generic REMOTE=CAN* at line 24. If lines 23 and 24 are coded in reverse order, RULE5 for REMOTE=CANADA01 is not chosen. Instead, a match is made for the generic value REMOTE=CAN*. The order of the SELECT statement is very important.

Logging Application Agent

The Logging application agent is invoked whenever Connect:Enterprise writes a log record for one of the following log record types:

- Auto Connect summary
- Auto Connect detail
- Remote Connect summary
- Remote Connect detail
- Auto Connect queue

By defining the Logging application agent rules, you can specify different actions to take based on certain characteristics of the log record. Each rule defined can instruct the application agent to process one or more of the following instructions:

- COMMAND—issue a console command
- EXECUTE—execute a user-specified program
- MESSAGE—issue a console message
- NOP—issue a no operation instruction to selectively exclude log records from processing
- SNMPTRAP—issue an SNMP trap (a message that reports a problem or a significant event, formatted and encoded as defined in RFC 1442, and sent as a UDP datagram) to an IP address
- SUBMIT—submit a job to the internal reader

When you invoke the logging application agent, the number of requests that are queued to the rules processors is increased very quickly, resulting in multiple occurrences of the message, CMB321W–C:E/PR MAXRP CONGESTION OCCURRED. Although this congestion does not cause problems processing the requests, processing is delayed until a rules processor is available. If you get these messages often, increase the value of the ODF parameter MAXRP= to allow more concurrent transactions to be processed.

Logging Rules

The following RULE statements and instructions are valid for the Logging application agent. Statements and instructions are highlighted in bold text. Parameters are indented under the related instruction, and parameter default values are underlined.

```
                                                                      72
RULE NAME=xxxxxxxx
COMMAND TEXT='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
     ERROR=CONTINUE|QUIT
  EXECUTE PROGRAM=xxxxxxxx,
     ERROR=CONTINUE|QUIT
  MESSAGE TEXT='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
     ROUTCODE=(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16),
     DESCCODE=(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16),
     CONEVENT=YES|NO,
     ERROR=CONTINUE|QUIT
  NOP
  SNMPTRAP TEXT='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
     IPADDR=host-name|nnn.nnn.nnn.nnn,
     PORT=nnnnn|162,
     GROUP1=ALARM|STATUS,
     GROUP2=nnnn|9999,
     ERROR=CONTINUE|QUIT
  SUBMIT MEMBER=xxxxxxxx,
     ERROR=CONTINUE|QUIT
 SELECT RULE=(xxxxxxxx,xxxxxxxx…),
    ACFUNC=(CONN,DISC,LOGON,RECV,SEND,SESSEN,SESSST,ULOG),
     ACQREASON=(LINE,ACTIVE,SESSION,THREAD),
     BATCHID='xxxx....xxxx'|"yyyy....yyyy",
     FAILCODE=nnn|nnn-nnn|(nnn,...,nnn),
     ID=xxxxxxxx,
     LINE=xxxxxxxx,
     LISTNAME=xxxxxxxx,
     LOGFUNC=(NEW,UPDATE),
     RCFUNC=(ADD,CONN,DEL,DIR,DISC,NOADD,REQ,SIGNON),
     RECTYPE=ACSUMMARY|ACDETAIL|RCSUMMARY|RCDETAIL|ACQUEUE,
     REMOTE=xxxxxxxx
     ULTEXT01-96=xxxxxxxx|(xxxxxxxx,xxxxxxxx,xxxxxxxx)
```

Logging Rules Example

The following is an example of Logging rules. An explanation of what each line means follows the example. The line numbers listed in this example are not displayed in your rules. They are for illustrative purposes only.

```
                                                                      72
 01 RULE NAME=RULE1
 02   EXECUTE PROGRAM=PGM1,ERROR=CONTINUE
 03   COMMAND TEXT='F ENTPRS,$$CONNECT LISTNAME=GROUP1'
 04 RULE NAME=RULE2
 05   MESSAGE TEXT='C:E ALERT! AC FAILED, LISTNAME=&LISTNAM LINE=            -
 06                &LINNAME',ROUTCODE=(1,15)
 07   SUBMIT  MEMBER=JOB25
 08 RULE NAME=RULE3
 09   SUBMIT MEMBER=JOB03
 10   MESSAGE TEXT='C:E ALERT! RC FAILED, REMOTE=&RMTNAME'
 11 RULE NAME=RULE4
 12   SUBMIT  MEMBER=JOB30
 13 RULE RULE5
 14   MESSAGE TEXT='C:E ALERT - AC FAILED, LISTNAME=&LISTNAM'
 15 RULE NAME=RULENOP
 16   NOP
 17 RULE NAME=RULE6
 18   SNMPTRAP TEXT='C:E ALERT! AC
FAILED,LISTNAME=&LISTNAM',IPADDR=11.22.33.44,GROUP2=9123
 19 SELECT RULE=RULE1,RECTYPE=ACDETAIL,LISTNAME=GROUP1,FAILCODE=046
 20 SELECT RULE=RULE2,RECTYPE=ACSUMMARY,LISTNAME=BSCTEST,
 21 LOGFUNC=UPDATE,FAILCODE=2-999
 22 SELECT RULE=RULE3,RECTYPE=RCDETAIL,REMOTE=RMT1*,FAILCODE=1-999
 23 SELECT RULE=RULE4,RECTYPE=RCSUMMARY,LOGFUNC=UPDATE
 24 SELECT RULE=RULE4,RECTYPE=RCDETAIL,RCFUNC=(ADD,NOADD)
 25 SELECT RULE=RULENOP,RECTYPE=ACDETAIL,FAILCODE=004
 26 SELECT RULE=RULE5,RECTYPE=ACDETAIL,FAILCODE=050-100
 27 SELECT RULE=(RULE2,RULE6),RECTYPE=ACSUMMARY,LOGFUNC-UPDATE,FAILCODE=2-999
 28 SELECT RULE=RULE7,RECTYPE=ACDETAIL,ACFUNC=ULOG,LISTNAME=LFTP1,REMOTE=FTPRMT1,
 29 FAILCODE=240,ULTEXT02=('emergency','failure','error'),ULTEXT08='call',
 30 ULTEXT10='Helpdesk'
```

The following table outlines detailed information for each line in the preceding example:

Line Number	Definition
01	Defines a rule named RULE1.
02	Specifies an EXECUTE instruction to call user program PGM1. If the program returns with a nonzero return code, execute the next instruction.
03	Specifies a COMMAND instruction to issue the $$CONNECT console command indicated in the quotes.
04	Defines a rule named RULE2.

Line Number	Definition
05	Specifies the MESSAGE instruction is to write the message contained inside the single quotes. The nonblank character in column 72 indicates the parameter is continued on the next line.
06	This line is a continuation of the MESSAGE instruction TEXT parameter on line 05. The ROUTCODE operand specifies a system console and a user-specified destination (route codes 1 and 15, respectively).
07	Specifies a SUBMIT instruction. The member name JOB25 is submitted to the Connect:Enterprise internal reader. JOB25 is retrieved from the data set or data sets referenced by the //RULESJCL DD statement in the Connect:Enterprise JCL.
08	Defines a rule named RULE3.
09	Specifies a SUBMIT instruction. The member name JOB03 is submitted to the Connect:Enterprise internal reader. JOB03 is retrieved from the data set or data sets referenced by the //RULESJCL DD statement in the Connect:Enterprise JCL.
10	Specifies the MESSAGE instruction to write the message contained inside the single quotes. The remote name symbolic variable is resolved before the message is displayed. The default route and descriptor codes are used (1 and 7, respectively).
11	Defines a rule named RULE4.
12	Specifies a SUBMIT instruction. The member name JOB30 is submitted to the Connect:Enterprise internal reader. JOB30 is retrieved from the data set or data sets referenced by the //RULESJCL DD statement in the Connect:Enterprise JCL.
13	Defines a rule named RULE5.
14	Specifies the MESSAGE instruction to write the message contained inside the quotes. The default route and descriptor codes are used (1 and 7, respectively).
15	Defines a rule named RULENOP.
16	Specifies the NOP instruction. No operation is performed.
17	Defines a rule named RULE6.
18	Specifies the SNMPTRAP instruction. A trap containing the text is sent to IP address 11.22.33.144, port 162, using GROUP 2=9123.
19—30	Indicates the selection criteria to use in determining which rules to process for specific log records.

Scheduler Application Agent

The Scheduler application agent can be invoked by:

- Setting parameters in the Scheduler SELECT statement

The Scheduler SELECT statement enables you to specify a maximum of eight rules to be processed and the dates, days, and times to activate processing. The CALENDAR= parameter enables you to specify dates or days of the week to activate processing or exclude from processing. The TIME= parameter enables you to specify the time of day to activate processing. To initiate processing, all parameters set in the SELECT statement must match the application agent event.

- Using the $$INVOKE RULES=SCH command to specify the names of rules to be processed or a SELECT statement to use for processing

  The $$INVOKE command provides two ways of invoking rule processing. You can use the RULENAME= parameter to specify a maximum of eight rules to be processed. When the RULENAME= parameter is used with the $$INVOKE command, rules are processed in the order in which they are listed, and multiple rules must be enclosed in parentheses. You can also request rules processing using the SELECT=*nnnnnnnn* parameter, where *nnnnnnnn* is the sequence number assigned to a SELECT statement during startup and refresh. The SELECT statement that you specify using the sequence number determines the rule statements that are processed.

By defining the Scheduler application rules, you can specify different actions to take based on the time of day. Each rule can instruct the application agent to process one or more of the following instructions:

- COMMAND—issues a console command
- EXECUTE—executes a user-specified program
- MESSAGE—issues a console message
- NOP—issues a no operation instruction to selectively exclude log records from processing
- SNMPTRAP—issues an SNMP trap (a message that reports a problem or a significant event, formatted and encoded as defined in RFC 1442, and sent as a UDP datagram) to an IP address
- SUBMIT—submits a job to the internal reader

Scheduler Rules

The following RULE statements and instructions are valid for the Scheduler application agent. Statements and instructions are highlighted in bold text. Parameters are indented under the related instruction, and parameter default values are underlined.

```
RULE NAME=xxxxxxxx
   COMMAND TEXT='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    ERROR=CONTINUE|QUIT
   EXECUTE PROGRAM=xxxxxxxx,
    ERROR=CONTINUE|QUIT
   MESSAGE TEXT='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    ROUTCODE=(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16),
    DESCCODE=(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16),
    CONEVENT=YES|NO,
    ERROR=CONTINUE|QUIT
   NOP
   SNMPTRAP TEXT='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    IPADDR=host-name|nnn.nnn.nnn.nnn,
    PORT=nnnnn|162,
    GROUP1=ALARM|STATUS,
    GROUP2=nnnn|9999,
    ERROR=CONTINUE|QUIT
   SUBMIT MEMBER=xxxxxxxx,
    ERROR=CONTINUE|QUIT
SELECT RULE=(xxxxxxxx,xxxxxxxx…),
   DESCRIPTION='xxx.xxx'
   TIME=(hh:mm,hh:mm,….),
   CALENDAR=xxxxxxxx
```

Scheduler Rules Example

The following is an example of Scheduler rules. An explanation of what each line means follows the example. The line numbers listed in this example are not displayed in your rules. They are for illustrative purposes only.

```
01 RULE NAME=RULE1
02   EXECUTE PROGRAM=PGM1,ERROR=CONTINUE
03   COMMAND TEXT='F ENTPRS,$$CONNECT LISTNAME=LN&TIME6'
04 RULE NAME=RULE2
05   MESSAGE TEXT='SCH Rule2 TIME=&TIME6',ROUTCODE=(15)
06   SUBMIT MEMBER=JOB25
07 RULE NAME=RULE3
08   NOP
09 RULE NAME=RULE4
10   SNMPTRAP  TEXT=' C:E HEARTBEAT: &HHMMSSTH',IPADDR=33.44.55.66
11 SELECT RULE=(RULE1,RULE4),TIME=(11:33,23:33)
12 SELECT RULE=RULE2,TIME=(00:00,00:01,00:02,00:03,00:04,00:05,00:06,00:07,00:08,00:09,00:10,00:11,00:12)
13 SELECT RULE=(RULE4,RULE3,RULE1), TIME=08:15,CALENDAR=WEEKDAYS
```

The following table outlines detailed information for each line in the preceding example:

Line	Definition
01	Defines a rule named RULE1.
02	Specifies an EXECUTE instruction to call user program PGM1. If the program returns with a nonzero return code, execute the next instruction.
03	Specifies a COMMAND instruction to issue the $$CONNECT console command indicated in the quotes.
04	Defines a rule named RULE2.

Line	Definition
05	Specifies the MESSAGE instruction is to write the message contained inside the single quotes. The ROUTCODE operand specifies a user-specified destination (route code 15).
06	Specifies a SUBMIT instruction. The member name JOB25 is submitted to the Connect:Enterprise internal reader. JOB25 is retrieved from the data set or data sets referenced by the //RULESJCL DD statement in the Connect:Enterprise JCL.
07	Defines a rule named RULE3.
08	Specifies the NOP instruction. No operation is performed.
09	Defines a rule named RULE4.
10	Specifies the SNMPTRAP instruction. A trap containing the text is sent to 33.44.55.66 port 162, using defaults, GROUP1= STATUS (2) and GROUP2 =9999.
11-13	Indicates the selection criteria to use in determining which rules to process for specific Scheduler agent events.

Wake Up Terminate Application Agent

The Wake Up Terminate application agent is invoked whenever CICS sends a Wake Up Terminate message to Connect:Enterprise. A Wake Up Terminate message is an acknowledgement to a prior Wake Up Initiate sent from Connect:Enterprise to CICS.

By defining the Wake Up Terminate application agent rules, you can specify different actions to take based on certain characteristics of the batch. Each rule defined can instruct the application agent to process one or more of the following instructions:

- COMMAND—issue a console command
- EXECUTE—execute a user-specified program
- MESSAGE—issue a console message
- NOP—issue a no operation instruction to selectively exclude batches from processing
- SNMPTRAP—issues an SNMP trap (a message that reports a problem or a significant event, formatted and encoded as defined in RFC 1442, and sent as a UDP datagram) to an IP address
- STATFLG—set batch status flags on or off
- SUBMIT—submit a job to the internal reader

Wake Up Terminate Rules

The following RULE statements and instructions are valid for the Wake Up Terminate application agent. Statements and instructions are highlighted in bold text. Parameters are indented under the related instruction, and parameter default values are underlined.

```
                                                                      72
RULE NAME=xxxxxxxx
  COMMAND TEXT='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    ERROR=CONTINUE|QUIT
  EXECUTE PROGRAM=xxxxxxxx,
    ERROR=CONTINUE|QUIT
  MESSAGE TEXT='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    ROUTCODE=(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16),
    DESCCODE=(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16),
    CONEVENT=YES|NO,
    ERROR=CONTINUE|QUIT
   NOP
  SNMPTRAP TEXT='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  IPADDR=host-name|nnn.nnn.nnn.nnn,
  PORT=nnnnn|162,
  GROUP1=ALARM|STATUS,
  GROUP2=nnnn|9999,
     ERROR=CONTINUE|QUIT
  STATFLG ONFLAGS=(REQUESTABLE,DELETED,TRANSMITTED,EXTRACTED,MULTXMIT),
   OFFFLAGS=(REQUESTABLE,DELETED,TRANSMITTED,EXTRACTED,MULTXMIT),
   ERROR=CONTINUE|QUIT
  SUBMIT  MEMBER=xxxxxxxx,
    ERROR=CONTINUE|QUIT
SELECT RULE=(xxxxxxxx,xxxxxxxx…),
    BATCHID='xxxx....xxxx'|"yyyy....yyyy",
    CASE_SENSITIVE=YES|NO
    CICSPGMNM=xxxxxxxx,
    CICSTRANID=xxxxxxxx,
    ID=xxxxxxxx,
    LINE=xxxxxxxx,
    ORIGIN=ALL|EOBRULES|EOBEXIT,
    REMOTE=xxxxxxxx,
    RTNCODE=nnnn|nnnn-nnnn|(nnnn,....,nnnn),
    WILD_CARD=BID
    WILD_CARD_MULTI_CHAR=*|x[xxxxxxx]
    WILD_CARD_SINGLE_CHAR=%|x[xxxxxxx]
  STATOR=(ADDED,BSC,COLLECTED,DELETED,EBCDIC,EXTRACTED,
  FILE_STUCTURE,FTP,INCOMPLETE,MULTXMIT,
  NONTRANSMITTABLE,REQUESTABLE,SNA,SSL,
  TRANSPARENT,TRANSMITTED,UNEXTRACTABLE)
  STATUS=(ADDED,BSC,COLLECTED,DELETED,EBCDIC,EXTRACTED,
  FILE_STUCTURE,FTP,INCOMPLETE,MULTXMIT,
  NONTRANSMITTABLE,REQUESTABLE,SNA,SSL,
  TRANSPARENT,TRANSMITTED,UNEXTRACTABLE)
```

Wake Up Terminate Rules Example

The following is an example of Wake Up Terminate rules and an explanation of what each line means. The line numbers listed in this example are not displayed in your rules. They are for illustrative purposes only.

```
                                                                      72
01 RULE NAME=RULE1
02   MESSAGE TEXT='WARNING, CICS WAKEUP FAILED FOR BATCHNO=&BATCH#
03                 ID=&IDFIELD  BID=&BID24',ROUTCODE=(1,13,15)
04   COMMAND TEXT='F &STCNAME,$$DIR ID=&IDFIELD'
05 RULE NAME=RULE2
06   SUBMIT MEMBER=JOB25
07 RULE NAME=RULE3
08   STATFLG ONFLAGS=(EXTRACTED,DELETED)
09 EXECUTE PROGRAM=PGM33
10 RULE NAME=RULENOP
11 NOP
12 RULE NAME=RULE6
13   SNMPTRAP  TEXT='BATCH#=&BATCH# ID=&IDFIELD BID=&BID24',IPADDR=11.22.33.44
14 SELECT RULE=RULENOP,RTNCODE=0025
15 SELECT RULE=RULE1,RTNCODE=F9F9
16 SELECT RULE=RULE2,RTNCODE=0001-0999,ORIGIN=EOBRULES
17 SELECT RULE=RULE3,ID=SANFRAN,BATCHID='RECEIVABLES',
18 REMOTE=BRANCH*,RTNCODE=0000,STATOR=(SNA,BSC)
19 SELECT RULE=(RULE1,RULE6),RTNCODE=F8F8
```

The following table outlines detailed information for each line in the preceding example:

Line Number	Definition
01	Defines a rule named RULE1.
02	Specifies a user-defined console message. The symbolic variables for batch number, Mailbox ID, and user batch ID is replaced with the actual values prior to issuing the message. The continuation character, nonblank in column 72, indicates the parameter is continued on the next line.
03	This line is a continuation of the MESSAGE instruction on line 02.
04	Specifies a COMMAND instruction to issue the $$DIRECTORY console command indicated in the quotes. The symbolic variables for the task name and Mailbox ID is replaced with the actual values prior to issuing the command.
05	Defines a rule named RULE2.
06	Specifies a SUBMIT instruction. The member name JOB25 is submitted to the Connect:Enterprise internal reader. JOB25 is retrieved from the data set referenced by the //RULESJCL DD statement allocated in the Connect:Enterprise job.
07	Defines a rule named RULE3.
08	Specifies a STATFLG instruction to turn on the E (EXTRACTED) and D (DELETED) batch status flags.
09	Specifies an EXECUTE instruction to call user program PGM33.

Line Number	Definition
10	Defines a rule named RULENOP.
11	Specifies the NOP instruction. No operation is performed.
12	Defines a rule named RULE6.
13	Specifies the SNMPTRAP instruction. A trap containing the text is sent to IP address 11.22.33.44. (The default port is 162.)
14 - 19	Indicates the selection criteria that determines which set of rules to process for specific criteria. Note the SELECT statement on line 16 is continued on the next line.

How Connect:Enterprise Uses Application Agents

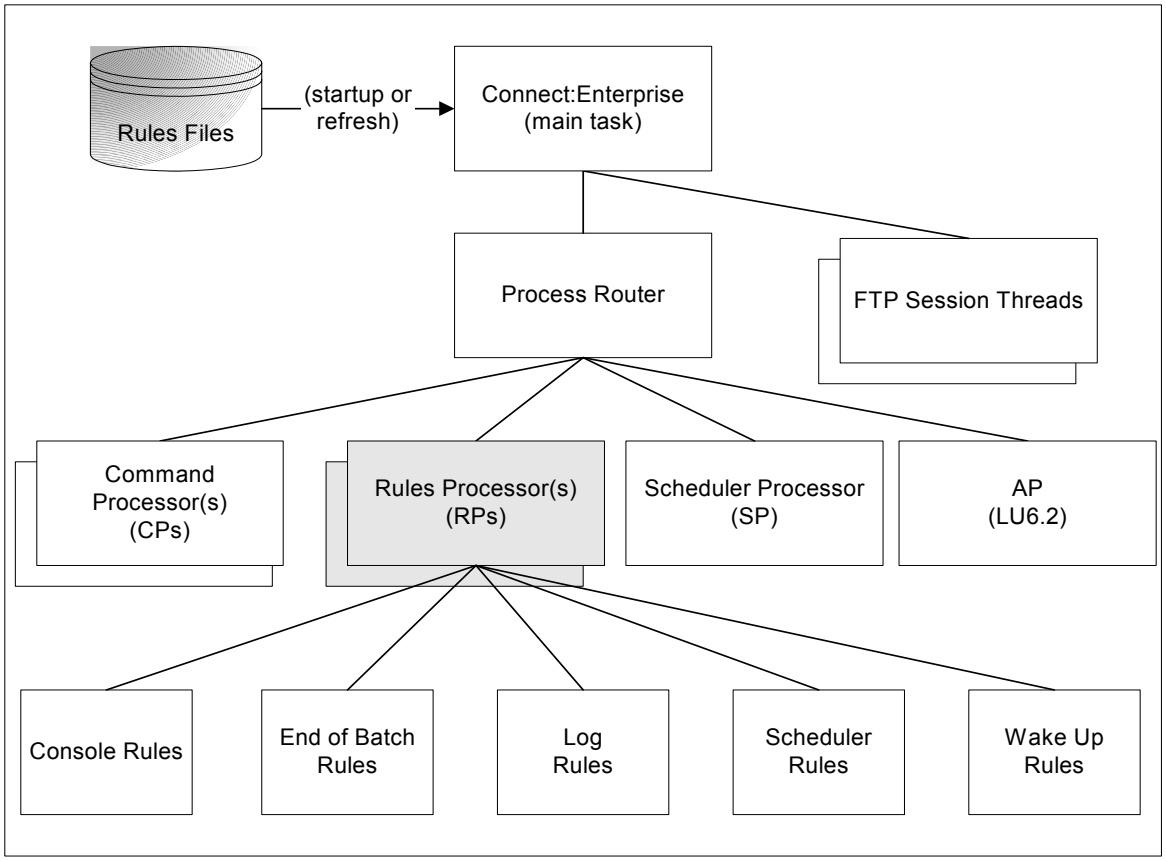
Connect:Enterprise uses application agents as follows:

- Connect:Enterprise reads each rule set during startup or rules refresh and loads it into memory. If Connect:Enterprise detects an error during startup, it terminates. If it detects an error during a rules refresh, no refresh occurs and the application agent continues using the existing rules tables loaded in memory. To avoid this, use the Rule Verification utility to verify the syntax each time you make a change to an existing rule set.
- An event causes Connect:Enterprise to send an application agent request to the Process Router task. Events are defined as follows for each type of application agent:
  - CON—a message is issued by Connect:Enterprise
  - EOB—a batch collection completes successfully
  - LOG—a log record is written
  - SCH—a specified time of day is reached
  - WKT—a wake up acknowledgement is sent from CICS
- The Process Router routes the request to the Rules Processor.

> **Note:** Each application agent request is processed asynchronously and independently from all other Connect:Enterprise activity.

- The Rules Processor (RP) scans through the rules SELECT statements for a match on the selection criteria.
- When a match is found, each instruction in the selected rules is executed.
- Processing is complete and the Rules Processor task is ready for another application agent request.

The following diagram illustrates how the Rules Processor task processes the application agent requests.



See Chapter 2, *Creating and Verifying Application Agent Rules*, for the guidelines, structure, and syntax for creating rule sets; the format and parameters of instructions; and how to verify rule sets. See Chapter 3, *Implementing Application Agent Rules*, for information on implementing rules, changing rules during processing, using the trace facility to test and debug rule sets, and for sample implementations.

Chapter 2

Creating and Verifying Application Agent Rules

This chapter describes the components and structure of a rule set for an application agent, guidelines and syntax for rules, and the instructions and parameters that are valid for the application agents.

Rule Set Components and Structure

A rule set controls how Connect:Enterprise uses application agents. It consists of at least one RULE statement, one instruction, and one SELECT statement. The RULE statement specifies the start of a rule and the name of the rule. Each rule name must be unique within an application agent PDS member. The format is as follows:

```
RULE NAME=xxxxxxxx
```

Instructions define the operations associated with a rule. The RULE statement must always be followed by one or more instructions to execute for the rule. The SELECT statement specifies selection criteria used to evaluate an event for processing and the rule, or rules, to process when an event matches the selection criteria. The SELECT statement can point to up to eight rules. The basic structure of each application agent rule set is as follows:

```
RULE NAME=rulename
 instruction
 instruction
 instruction
RULE NAME=rulename
 instruction
 instruction
 instruction
SELECT RULE=rulename,match criteria
SELECT RULE=rulename,match criteria
SELECT RULE=(rulename,rulename),match criteria
SELECT RULE=rulename,match criteria
```

The following example illustrates a simple rule set used to submit an extract job when Connect:Enterprise collects a batch online with batch ID of BRANCH01. You implement the End of Batch application agent to process the request, specify the rule name, and use the SUBMIT instruction to define the extract job. The SELECT parameter specifies to apply RULE001 when a batch with ID BRANCH01 is collected:

```
RULE   NAME=RULE001
       SUBMIT MEMBER=EXTRACT
SELECT RULE=RULE001,ID=BRANCH01
```

The End of Batch application agent processes the request when Connect:Enterprise collects a batch online, then each SELECT statement is processed until a match is made on the selection criteria. As the preceding rule set specifies, RULE001 is processed each time Connect:Enterprise collects a batch with an ID of BRANCH01. RULE001 specifies a SUBMIT instruction with a JCL member name of EXTRACT that is submitted to the internal reader. The member is read from the RULESJCL DD file.

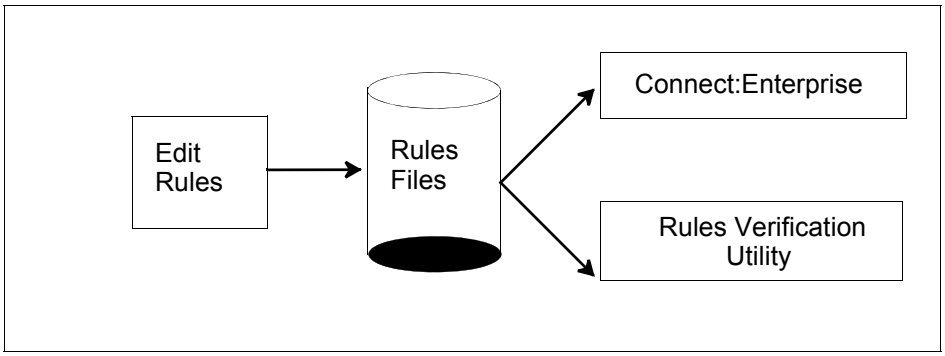
Guidelines for Defining a Rule Set

Adhere to the following guidelines when writing a rule set:

- Define a unique name for each rule; duplicate names are not allowed.
- Specify one or more instructions for each rule. There is no limit to the number of instructions that you can specify in each rule.
- Specify instructions in each rule in the order that you want them processed. Each instruction must have at least one parameter specified on the same line (record) as the instruction itself.
- Specify each SELECT statement to point to up to eight rules and include at least one additional selection parameter. You can also specify multiple SELECT statements to reference a single rule.
- To ensure accurate processing, list the SELECT statements that contain detailed selection criteria before the SELECT statements that contain general selection criteria.

  If NO match occurs on any of the SELECT statements, no rule is processed for the request.
- Specify RULE statements and SELECT statements in any order in each rule set.
- If you change a rules file while Connect:Enterprise is running, use the Verification utility to verify the syntax before refreshing the rule set.

The following diagram illustrates how the rules are used as input to Connect:Enterprise and the Rules Verification utility:



Rule Structure and Syntax

This section explains the general syntax for writing rules. In each record, columns 1–71 are for writing rules, column 72 is only for continuation, and columns 73–80 are ignored. The following table details the function of some syntax elements.

Element	Function	Example
Statement Identifier	Specifies a RULE or SELECT statement or an instruction to execute within a rule.	`RULE NAME=Rule01` (statement) SUBMIT MEMBER=EXTRACT (instruction) SELECT RULE=RULE01,ID=BRANCH01 (statement)
Keyword parameter	Specifies further instructions for statements and is followed by an equals sign. It can have a set of subparameters. Multiple parameters must be separated by commas. Rules parameters are keyword parameters.	`MESSAGE TEXT='My test message',` `ROUTCODE=(1,7,11),ERROR=QUIT`
Subparameter	Specifies variable information in a keyword parameter including a list of subparameters. Such a list can comprise both text strings and symbolic variables. Enclose the subparameter list in parentheses and separate the items by commas.	`ROUTCODE=(1,7,11)`

Element	Function	Example
Comma	Separates items within a subparameter list.	`ROUTCODE=(1,7,11)`
	Separates keyword parameters.	`MESSAGE TEXT='My test message', ROUTCODE=(1,7,11),ERROR=QUIT`
Parentheses	Enclose lists and associate groups of values.	`MESSAGE TEXT='This is a test message',ROUTCODE=(1,5,7,13)` The subparameters specified in the ROUTCODE keyword are enclosed in parentheses.

Continuation Marks in a Statement

A comma followed by a blank space indicates that the statement continues on the next line. The following example illustrates a statement specified on one line:

```
SELECT RULE=RULE022,BATCHID='Test Batch ID',ID=MBOX99
```

You can also specify the previous example over multiple lines, as follows:

```
SELECT RULE=RULE022,
       BATCHID='Test Batch ID',
       ID=MBOX99
```

Continuation Marks in a Parameter

You can use a nonblank character in column 72 as a continuation mark to indicate a parameter value continues on multiple lines. In the following examples, a hyphen (-) is the continuation character in column 72:

```
                                                                72
MESSAGE TEXT='This is a test msg to show how to conti-
            nue to line 2'
```

The previous illustration is displayed as:

```
This is a test msg to show how to continue to line 2
```

Another example of continuation marks follows:

```
                                                                72
STATOR=(SNA,BSC,TRANSMITTED,REQUESTABLE,TRANSPARENT,E-
       BCDIC,MULTXMIT)
```

The previous illustration is displayed as:

```
STATOR=(SNA,BSC,TRANSMITTED,REQUESTABLE,TRANSPARENT,EBCDIC,MULTXMIT)
```

The parameter value is concatenated with the first nonblank character on the next line.

Comments

Use comments to include additional information in a set of Connect:Enterprise application agent rules. Comments are indicated by one of the following methods:

- Placing an asterisk in column 1 indicates to ignore the entire record.
- Placing comments on any rule record to the right of all statements and parameters. Precede the comment with at least one space. An example follows:

```
************************************************
* This rule executes the weekly payroll program *
************************************************
RULE NAME=RULE001         this is a comment
  SUBMIT MEMBER=PAYROLL1  this is also a comment
SELECT RULE=RULE001,      this is also a comment
  ID=MBOX001              this is also a comment
```

Special Characters

The symbols defined as special characters for Connect:Enterprise application agent rules are the ampersand (&), the at sign (@), and the following delimiters:

Symbol	Description
	blank
=	equal sign
*	asterisk
,	comma
( )	parentheses
’	single quotation mark
“	double quotation mark

Special-Purpose Bracketing

To maintain special characters as part of a string, enclose the string in either single quotation marks (’ ’) or double quotation marks (“ ”).

- Use single quotation marks to embed special characters or blanks in a parameter or subparameter value.

The following is an example of special purpose bracketing before resolution:

```
MESSAGE TEXT='This is a test message with blanks'
```

The previous illustration is displayed as:

```
This is a test message with blanks
```

The following is an example of using single quotation marks:

```
                                                    72
MESSAGE TEXT='This is a test msg with '''quotes''' -
 and blanks on two lines'
```

The previous illustration is displayed as:

```
This is a test msg with 'quotes' and blanks on two lines
```

To obtain embedded single quotes, specify three consecutive single quotes to resolve to one.

- Use double quotation marks to specify a generic user batch ID in the BATCHID parameter in a SELECT statement and enable you to use embedded blanks and special characters in the user batch ID. An example follows:

```
SELECT RULE01,BATCHID="Test Batch 123"
```

Symbolic Substitution

Symbolic substitution enables you to substitute information in JCL members, console commands, console messages, and SNMP trap messages. When rules processing encounters either an ampersand (&) or an at sign (@), plus 1–7 alphanumeric characters defined as a symbolic variable, the variable is replaced by the corresponding data string. Trailing blanks are treated differently in substitutions using the ampersand (&) variable and the at sign (@) variable:

- When the & variable is used, trailing blanks are removed when the variable value is resolved.
- When the @ variable is used, trailing blanks are *not* removed when the variable value is resolved (with the exception of the @APKEY, @MSG, @MSG01–@MSG96, and @ULTEXT01–@ULTEXT96 variables).

You can use the symbolic variables only in the following places:

- Any of the SUBMIT instruction's JCL member (columns 1–71 only)
- Console COMMAND instruction text
- Console MESSAGE instruction text
- SNMPTRAP instruction text

Connect:Enterprise replaces symbolic variables with the actual values prior to submission. No abbreviations are permitted. Specify each variable in upper case text.

All symbolic variables specified in a text string, for example, in MESSAGE TEXT or SNMPTRAP TEXT, must be valid variables for that particular agent and RECTYPE under the SELECT instruction. If a symbolic variable not valid for the agent and record type is included, the error, CMB365E - C:E RULES DATA NOT AVAILABLE TO REPLACE SYMBOLICS IN COMMAND xxxxxxx, results. For example, if you used the LOG agent with an ACS record type under the SELECT instruction and included an SNMP instruction containing the ACFUNC symbolic variable, an error would result because ACFUNC is not valid for the ACS RECTYPE. (For a quick reference, see the matrix in *Symbolic Variables Valid for Application Agent Rules* on page 43.) In this case, you would see the message, CMB365E - C:E RULES DATA NOT AVAILABLE TO REPLACE SYMBOLICS IN SNMPTRAP.

Along with the CMB365E console message, Connect:Enterprise issues M$DUMP to trace the data in error, which will be written to the SNAPOUT DD entitled 'STRPPS01 - DATA NOT AVAILABLE TO REPLACE SYMBOLIC.'

Ampersand (&) and At Sign (@) Variables

The following table describes the value used for each ampersand (&) and at sign (@) variable. Precede the variable with either the ampersand symbol (&) or at sign (@) symbol depending on whether you want trailing blanks or not.

Variable	Value Used
ACCDATE	The Auto Connect completion date in 7-digit Julian date yyyyddd format.
ACCTIME	The Auto Connect completion time in 6-digit hhmmss format.
ACFAILC	The 4-digit Auto Connect summary total failed collection count.
ACFAILX	The 4-digit Auto Connect summary total failed transmission count.
ACFUNC	A 1-byte function code which corresponds to an Auto Connect function: 1 = RECV (batch collected) 3 = SEND (batch transmittedl) 8 = LOGON (FTP user logon) C = CONN (FTP connect) D = DISC (FTP disconnect) E = SESSEN (FTP session end) S = SESSST (FTP session start) U = ULOG (FTP USERLOG record)
ACSDATE	The Auto Connect start date in 7-digit Julian date yyyyddd format.
ACSTIME	The Auto Connect start time in 6-digit hhmmss format.
ACSUCCC	The 4-digit Auto Connect summary total successful collection count.
ACSUCCX	The 4-digit Auto Connect summary total successful transmission count.

Variable	Value Used
APKEY	A string containing the APKEY records, W and Z records not included. The APKEY can contain up to 1024 bytes. The records are stripped of trailing blanks and x'0D' is added to the end of each record as a delimiter. x'0D' is also added when the end of file is reached. If APDSN is not specified in the ODF, the APKEY value is null.
AQQDATE	The queuing date of the Auto Connect in 5-digit Julian date yyddd format.
AQQTIME	The queuing time of the Auto Connect in 4-digit hhmm format.
AQRDATE	The queued Auto Connect restart date in 5-digit Julian date yyddd format.
AQRTIME	The queued Auto Connect restart time in 6-digit hhmmss format.
BATCH#	The 7-digit batch number, including the leading zeros.
BCDATE	The batch creation date in 7-digit Julian date yyyyddd format.
BCTIME	The batch creation time in 6-digit hhmmss format.
BID(PP,LL)	A user-specified starting position and length of the user batch ID, where: PP = a two-digit value representing the starting position. Valid values are 01–64. LL = a two-digit value representing the length to be resolved. Valid values are 01–64. **Note:** If start position PP + length value LL exceeds the end of the 64-character user batch ID, the LL value is automatically adjusted such that PP + LL equals the end of the batch ID and is resolved accordingly. Example 1: If BID(14,06) is specified, positions 14–19 of the user batch ID are resolved. Example 2: If BID(01,64) is specified, the entire 64 character user batch ID is resolved. This example would effectively be the same as specifying &BID64. Example 3: If BID(64,01) is specified, only the last byte of the user batch ID is resolved. Example 4:If BID(60,10) is specified, the length value is adjusted to BID(60,05) at time of symbolic resolution and bytes 60-64 of the batch ID are resolved.
BID1	The first 8 bytes of the 24-byte (or 64-byte) user batch ID.
BID2	The second 8 bytes of the 24-byte (or 64-byte) user batch ID.
BID3	The third 8 bytes of the 24-byte (or 64-byte) user batch ID.
BID4	The fourth 8 bytes of the 64-byte user batch ID.
BID5	The fifth 8 bytes of the 64-byte user batch ID.
BID6	The sixth 8 bytes of the 64-byte user batch ID.
BID7	The seventh 8 bytes of the 64-byte user batch ID.
BID8	The eighth 8 bytes of the 64-byte user batch ID.
BID24	The 24-byte user batch ID.
BID64	The 64-byte user batch ID.
BLKCNT	The number of physical VSAM records in the batch.

Variable	Value Used
BYTECNT	The number of bytes in the batch.
CCVBQDSN	The 44-byte data set name of the current collection file.
CCVBQID	The 5-byte current collection file VBQnn value (VBQ01 - VBQ20).
CCVBQNUM	The 2-byte number of the current collection VBQ (01 - 20).
CCVLFDSN	The 44-byte data set name of the current log file.
CCVLFNUM	The 1-byte number of the current collection VLF (1 - 8).
CCVLFID	The 4-byte current collection file VLFn value (VLF1 - VLF8).
DATE	The current Julian date (YYYYDDD).
DAY	The day of the week in mixed case (Monday, Tuesday, etc.).
DAYUC	The day of the week in uppercase (MONDAY, TUESDAY, etc.).
DD	The current date in 2-digit dd format.
DDMMYYYY	The current date in 8-digit ddmmyyyy format.
FAILCODE	A 3-digit character value of nnn (where nnn is the auto connect -or- remote connect failcode).
FCDATE	The remote connection function completion date in 7-digit Julian date yyyyddd format. See *SELECT Statement Parameters* on page 76, for a description of the RCFUNC parameter.
FCTIME	The remote connection function completion time in 6-digit hhmmss format. See *SELECT Statement Parameters* on page 76, for a description of the RCFUNC parameter.
FSDATE	The remote connection function start date in 7-digit Julian date yyyyddd format. See *SELECT Statement Parameters* on page 76, for a description of the RCFUNC parameter.
FSTIME	The remote connection function start time in 6-digit hhmmss format. See *SELECT Statement Parameters* on page 76, for a description of the RCFUNC parameter.
HHMM	The current time in 4-digit HHMM format.
HHMMSS	The current time in 6-digit HHMMSS format.
HHMMSSTH	The current time in 8-digit HHMMSSTH format.
HOUR	The current time in 2-digit hh (hours) format.
IDFIELD	The 8-byte Mailbox ID, if available.
KEY	The 36 byte unpacked key of the log record being written which triggered the logging application agent event. Log records are often updated. The &KEY variable allows you to associate a "New" logging event with a subsequent "Update" logging event.
LINNAME	The 8-byte BSC line ID.

Variable	Value Used
LISTNAM	The Auto Connect list name (Auto Connect only).
LOGFUNC	A 1-byte character indicating that the log record being written is new (N) or an update (U) to an existing log record.
MBXNAME	The 8-byte Mailbox name specified in the ODF *OPTIONS (MBXNAME=).
MIN	The current time in 2-digit mm (minutes) format.
MM	The current month in 2-digit mm format.
MMDDYYYY	The current date in 8-digit MMDDYYY format.
MONTH	The month in mixed case (January, February, etc.).
MONTHUC	The month in uppercase (JANUARY, FEBRUARY, etc.).
MSG	The message text, which can contain up to 700 bytes.
MSG01	The first blank-delimited word of the &MSG variable uppercased.
MSG02–MSG96	The words (delimited by blank, comma, period, or colon) of the &MSG variable after &MSG01, or are null.
NXVBQDSN	The 44-byte data set name of the next (CC+1, or next available) VBQ. If VBQROTAT is less than 2, the data set name of the current collection file is used.
NXVBQID	The 5-byte ID of the next (CC+1, or next available) VBQ. If VBQROTAT is less than 2, the data set name of the current collection file is used.
NXVBQNUM	The 2-byte number of the next (CC+1, or next available) VBQ (01 - 20). If VBQROTAT is less than 2, the number of the current collection file is used.
NXVLFDSN	The 44-byte data set name of the next (CC+1, or next available) VLF. If VLFOTAT is less than 2, the data set name of the current collection file is used.
NXVLFID	The 5-byte ID of the next (CC+1, or next available) VLF (1 - 8). If VBQROTAT is less than 2, the data set name of the current collection file is used.
NXVLFNUM	The 1-byte number of the next (CC+1, or next available) VLF (1 - 8). If VBQROTAT is less than 2, the number of the current collection file is used.
PRVBQDSN	The 44-byte data set name of the previous (CC-1) VBQ. If VBQROTAT is less than 2, the data set name of the current collection file is used.
OSNAME	The 4 to 6 byte operating system name, such as z/OS)
OSVER	The operating system version in a 6-digit vvrrmm format
PRVBQNUM	The 2-byte number of the previous (CC-1) VBQ (01 - 20).
PRVBQID	The 5-byte ID of the previous (CC-1) VBQ. If VBQROTAT is less than 2, the data set name of the current collection file is used.
PRVLFDSN	The 44-byte data set name of the previous (CC-1) VLF. If VLFROTAT is less than 2, the data set name of the current collection file is used.

Variable	Value Used
PRVLFID	The 5-byte ID of the previous (CC-1) VLF. If VLFROTAT is less than 2, the data set name of the current collection file is used.
PRVLFNUM	The 1-byte number of the previous (CC-1) VLF (1 - 8).
RCFUNC	A 1-byte function code which corresponds to an Remote Connect function: 1 = ADD (batch containing a $$ADD record) 2 = NOADD (batch without a $$ADD record) 3 = REQ (batch request from the remote) 4 = DIR (directory request from the remote) 5 = DEL (delete request from the remote) 8 = SIGNON (signon/logon) C = CONN (connect) D = DISC (disconnect)
RCSCDAT	The remote connection summary completion date in 7-digit Julian date yyyyddd format. See *SELECT Statement Parameters* on page 76, for a description of the RCFUNC parameter.
RCSCTIM	The remote connection summary completion time in 6-digit hhmmss format. See *SELECT Statement Parameters* on page 76, for a description of the RCFUNC parameter.
RCSSDAT	The remote connection summary start date in 7-digit Julian date yyyyddd format. See *SELECT Statement Parameters* on page 76, for a description of the RCFUNC parameter.
RCSSTIM	The remote connection summary start time in 6-digit hhmmss format. See *SELECT Statement Parameters* on page 76, for a description of the RCFUNC parameter.
RECCNT	The number of logical data records in the batch.
RMTNAME	The 8-byte remote name.
RMTTYPE	The 1-byte representation of the remote type (1=BSC, 2=SNA, 4=FTP).
SEC	The current time in 2-digit (seconds) ss format.
SRVRID	The 4-character VSAM file server name.
STCNAME	The Connect:Enterprise started task name.
TH	The current time in 2-digit (tenths/hundredths) th format.
TIME	The hour and minute (HHMM) of the current time.
TIME6	The current time in hour/minute/second hhmmss format.
ULTEXT01–ULTEXT96	The words, delimited by blanks, commas, periods, or colons, of the text written to the FTP Auto Connect Detail USERLOG record.
VBQO1DSN	The VBQ01 data set name.
VBQO2DSN	The VBQ02 data set name.

Variable	Value Used
VBQO3DSN	The VBQ03 data set name.
VBQO4DSN	The VBQ04 data set name.
VBQO5DSN	The VBQ05 data set name.
VBQO6DSN	The VBQ06 data set name.
VBQO7DSN	The VBQ07 data set name.
VBQO8DSN	The VBQ08 data set name.
VBQO9DSN	The VBQ09 data set name.
VBQ10DSN	The VBQ10 data set name.
VBQ11DSN	The VBQ11 data set name.
VBQ12DSN	The VBQ12 data set name.
VBQ13DSN	The VBQ13 data set name.
VBQ14DSN	The VBQ14 data set name.
VBQ15DSN	The VBQ15 data set name.
VBQ16DSN	The VBQ16 data set name.
VBQ17DSN	The VBQ17 data set name.
VBQ18DSN	The VBQ18 data set name.
VBQ19DSN	The VBQ19 data set name.
VBQ20DSN	The VBQ20 data set name.
VCFDSN	The VCF data set name.
VLF1DSN	The VLF1 data set name.
VLF2DSN	The VLF2 data set name.
VLF3DSN	The VLF3 data set name.
VLF4DSN	The VLF4 data set name.
VLF5DSN	The VLF5 data set name.
VLF6DSN	The VLF6 data set name.
VLF7DSN	The VLF7 data set name.
VLF8DSN	The VLF8 data set name.
VPFDSN	The VPF data set name.
YY	The current date in 2-digit yy format, e.g., 03.
YYDDD	The current Julian date in 5-digit YYDDD format.

Variable	Value Used
YYYY	The current date in 4-digit yyyy format, e.g., 2003.
YYYYDDD	The current Julian date in 7-digit YYYYDDD format.
YYYYMMDD	The current Julian date in 8-digit YYYYDDD format.

The symbolic variable values listed are used in the following examples:

Symbolic Variable	Description	Data Length	Data Value
IDFIELD	Mailbox ID	8	'MBOX1 '
BID24	User batch ID	24	'Test Batch '
RMTNAME	Remote Name	8	'RMT001 '

The following is an example of using the ampersand (&) variable with trailing blanks removed:

```
MESSAGE TEXT='BATCH COLLECTED ID=&IDFIELD BID=&BID24
        REMOTE=&RMTNAME'
```

The previous illustration is displayed as:

```
BATCH COLLECTED ID=MBOX1 BID=Test Batch REMOTE=RMT001
```

The following is an example of using the at sign (@) variable without the trailing blanks removed:

```
MESSAGE TEXT='BATCH COLLECTED ID=@IDFIELD BID=@BID24 REMOTE=@RMTNAME'
```

The previous example is displayed as:

```
BATCH COLLECTED ID=MBOX1   BID=Test Batch             REMOTE=RMT001
```

Symbolic Variables Valid for Application Agent Rules

The following table identifies which symbolic variables are valid for each application agent rule type and applies to both & and @ symbolic variables. Logging rules are further classified by log record type (RECTYPE).

Symbolic Variable	Application Agent Rule Type								
	CON	EOB	SCH	WKT	LOG (ACS)	LOG (ACD)	LOG (RCS)	LOG (RCD)	LOG (ACQ)
ACCDATE					X	X			
ACCTIME					X	X			
ACFAILC					X				
ACFAILX					X				
ACFUNC						X			
ACSDATE					X	X			
ACSTIME					X	X			
ACSUCCC					X				
ACSUCCX					X				
APKEY	X	X	X	X	X	X	X	X	X
AQQDATE									X
AQQTIME									X
AQRDATE									X
AQRTIME									X
BATCH#		X		X		X		X	
BCDATE		X		X					
BCTIME		X		X					
BID(PP,LL)		X		X		X		X	
BID1		X		X		X		X	
BID2		X		X		X		X	
BID3		X		X		X		X	
BID4		X		X		X		X	
BID5		X		X		X		X	

1 CON=Console
EOB=End of Batch
SCH=Scheduler
WKT=Wake Up Terminate

2 LOG=Logging
ACS=Auto Connect Summary Log Record
ACD=Auto Connect Detail Log Record
RCS=Remote Connect Summary Log Record
RCD=Remote Connect Detail Log Record
ACQ=Auto Connect Queue Log Record

Symbolic Variable	Application Agent Rule Type								
	CON	EOB	SCH	WKT	LOG (ACS)	LOG (ACD)	LOG (RCS)	LOG (RCD)	LOG (ACQ)
BID6		X		X		X		X	
BID7		X		X		X		X	
BID8		X		X		X		X	
BID24		X		X		X		X	
BID64		X		X		X		X	
BLKCNT		X		X				X	
BYTECNT		X						X	
CCVBQDSN	X	X	X	X	X	X	X	X	X
CCVBQID	X	X	X	X	X	X	X	X	X
CCVBQNUM	X	X	X	X	X	X	X	X	X
CCVLFDSN	X	X	X	X	X	X	X	X	X
CCVLFID	X	X	X	X	X	X	X	X	X
CCVLFNUM	X	X	X	X	X	X	X	X	X
DATE	X	X	X	X	X	X	X	X	X
DAY	X	X	X	X	X	X	X	X	X
DAYUC	X	X	X	X	X	X	X	X	X
DD	X	X	X	X	X	X	X	X	X
DDMMYYYY	X	X	X	X	X	X	X	X	X
FAILCODE					X	X	X	X	
FCDATE								X	
FCTIME								X	
FSDATE								X	
FSTIME								X	

1 CON=Console
EOB=End of Batch
SCH=Scheduler
WKT=Wake Up Terminate

2 LOG=Logging
ACS=Auto Connect Summary Log Record
ACD=Auto Connect Detail Log Record
RCS=Remote Connect Summary Log Record
RCD=Remote Connect Detail Log Record
ACQ=Auto Connect Queue Log Record

Symbolic Variable	Application Agent Rule Type								
	CON	EOB	SCH	WKT	LOG (ACS)	LOG (ACD)	LOG (RCS)	LOG (RCD)	LOG (ACQ)
HHMM	X	X	X	X	X	X	X	X	X
HHMMSS	X	X	X	X	X	X	X	X	X
HHMMSSTH	X	X	X	X	X	X	X	X	X
HOUR	X	X	X	X	X	X	X	X	X
IDFIELD		X		X		X		X	
KEY					X	X	X	X	
LINNAME		X		X		X		X	
LISTNAM					X	X			X
LOGFUNC						X		X	
MBXNAME	X	X	X	X	X	X	X	X	X
MIN	X	X	X	X	X	X	X	X	X
MM	X	X	X	X	X	X	X	X	X
MMDDYYYY	X	X	X	X	X	X	X	X	X
MONTH	X	X	X	X	X	X	X	X	X
MONTHUC	X	X	X	X	X	X	X	X	X
MSG	X								
MSG01	X								
MSG02 – MSG96	X								
NXVBQDSN	X	X	X	X	X	X	X	X	X
NXVBQID	X	X	X	X	X	X	X	X	X
NXVBQNUM	X	X	X	X	X	X	X	X	X
NXVLFDSN	X	X	X	X	X	X	X	X	X

1 CON=Console
EOB=End of Batch
SCH=Scheduler
WKT=Wake Up Terminate

2 LOG=Logging
ACS=Auto Connect Summary Log Record
ACD=Auto Connect Detail Log Record
RCS=Remote Connect Summary Log Record
RCD=Remote Connect Detail Log Record
ACQ=Auto Connect Queue Log Record

Symbolic Variable	Application Agent Rule Type								
	CON	EOB	SCH	WKT	LOG (ACS)	LOG (ACD)	LOG (RCS)	LOG (RCD)	LOG (ACQ)
NXVLFID	X	X	X	X	X	X	X	X	X
NXVLFNUM	X	X	X	X	X	X	X	X	X
PRVBQDSN	X	X	X	X	X	X	X	X	X
PRVBQID	X	X	X	X	X	X	X	X	X
PRVBQNUM	X	X	X	X	X	X	X	X	X
PRVLFDSN	X	X	X	X	X	X	X	X	X
PRVLFID	X	X	X	X	X	X	X	X	X
PRVLFNUM	X	X	X	X	X	X	X	X	X
RCFUNC								X	
RCSCDAT							X		
RCSCTIM							X		
RCSSDAT							X		
RCSSTIM							X		
RECCNT		X							
RMTNAME		X		X		X	X	X	
RMTTYPE		X		X		X	X	X	
SEC	X	X	X	X	X	X	X	X	X
SRVRID	X	X	X	X	X	X	X	X	X
STCNAME	X	X	X	X	X	X	X	X	X
TH	X	X	X	X	X	X	X	X	X
TIME	X	X	X	X	X	X	X	X	X
TIME6	X	X	X	X	X	X	X	X	X

1 CON=Console
EOB=End of Batch
SCH=Scheduler
WKT=Wake Up Terminate

2 LOG=Logging
ACS=Auto Connect Summary Log Record
ACD=Auto Connect Detail Log Record
RCS=Remote Connect Summary Log Record
RCD=Remote Connect Detail Log Record
ACQ=Auto Connect Queue Log Record

Symbolic Variable	Application Agent Rule Type								
	CON	EOB	SCH	WKT	LOG (ACS)	LOG (ACD)	LOG (RCS)	LOG (RCD)	LOG (ACQ)
ULTEXT01–ULTEXT96						X			
VBQ01DSN	X	X	X	X	X	X	X	X	X
VBQ02DSN	X	X	X	X	X	X	X	X	X
VBQ03DSN	X	X	X	X	X	X	X	X	X
VBQ04DSN	X	X	X	X	X	X	X	X	X
VBQ05DSN	X	X	X	X	X	X	X	X	X
VBQ06DSN	X	X	X	X	X	X	X	X	X
VBQ07DSN	X	X	X	X	X	X	X	X	X
VBQ08DSN	X	X	X	X	X	X	X	X	X
VBQ09DSN	X	X	X	X	X	X	X	X	X
VBQ10DSN	X	X	X	X	X	X	X	X	X
VBQ11DSN	X	X	X	X	X	X	X	X	X
VBQ12DSN	X	X	X	X	X	X	X	X	X
VBQ13DSN	X	X	X	X	X	X	X	X	X
VBQ14DSN	X	X	X	X	X	X	X	X	X
VBQ15DSN	X	X	X	X	X	X	X	X	X
VBQ16DSN	X	X	X	X	X	X	X	X	X
VBQ17DSN	X	X	X	X	X	X	X	X	X
VBQ18DSN	X	X	X	X	X	X	X	X	X
VBQ19DSN	X	X	X	X	X	X	X	X	X
VBQ20DSN	X	X	X	X	X	X	X	X	X
VCFDSN	X	X	X	X	X	X	X	X	X

1 CON=Console
EOB=End of Batch
SCH=Scheduler
WKT=Wake Up Terminate

2 LOG=Logging
ACS=Auto Connect Summary Log Record
ACD=Auto Connect Detail Log Record
RCS=Remote Connect Summary Log Record
RCD=Remote Connect Detail Log Record
ACQ=Auto Connect Queue Log Record

Symbolic Variable	Application Agent Rule Type								
	CON	EOB	SCH	WKT	LOG (ACS)	LOG (ACD)	LOG (RCS)	LOG (RCD)	LOG (ACQ)
VLF1DSN	X	X	X	X	X	X	X	X	X
VLF2DSN	X	X	X	X	X	X	X	X	X
VLF3DSN	X	X	X	X	X	X	X	X	X
VLF4DSN	X	X	X	X	X	X	X	X	X
VLF5DSN	X	X	X	X	X	X	X	X	X
VLF6DSN	X	X	X	X	X	X	X	X	X
VLF7DSN	X	X	X	X	X	X	X	X	X
VLF8DSN	X	X	X	X	X	X	X	X	X
VPFDSN	X	X	X	X	X	X	X	X	X
YY	X	X	X	X	X	X	X	X	X
YYDDD	X	X	X	X	X	X	X	X	X
YYYY	X	X	X	X	X	X	X	X	X
YYYYDDD	X	X	X	X	X	X	X	X	X
YYYYMMDD	X	X	X	X	X	X	X	X	X

1 CON=Console
EOB=End of Batch
SCH=Scheduler
WKT=Wake Up Terminate

2 LOG=Logging
ACS=Auto Connect Summary Log Record
ACD=Auto Connect Detail Log Record
RCS=Remote Connect Summary Log Record
RCD=Remote Connect Detail Log Record
ACQ=Auto Connect Queue Log Record

Instructions

Instructions specify the operations to perform based on the characteristics of the event that causes Connect:Enterprise to initiate an application agent request. The following table summarizes the instructions that are valid for each type of application agent rule. The remainder of this section describes the format and parameters for each instruction.

Instruction	Application Agent Rule Type				
	Console	End of Batch	Scheduler	Wake up Terminate	Logging
COMMAND	X	X	X	X	X
EXECUTE	X	X	X	X	X
MESSAGE	X	X	X	X	X
NOP	X	X	X	X	X
ROUTE		X			
SNMPTRAP	X	X	X	X	X
STATFLG		X		X	
SUBMIT	X	X	X	X	X
WAKEUP		X			

COMMAND Instruction

You can use the COMMAND instruction to issue an operator console command. The COMMAND instruction is valid for all rule types.

To execute the COMMAND instruction, all load libraries in the Connect:Enterprise JOBLIB or STEPLIB must be APF authorized.

This instruction uses the z/OS Extended Multiple Console Support facility. Each time a COMMAND instruction is processed, a temporary logical console ID is allocated. The console ID is the value specified in the MBXNAME=xxxxxxxx parameter found in the *OPTIONS section of the ODF (Options Definition File). If the MBXNAME parameter is not specified, a default value of MAILBOX is used.

Each installation has the option of securing extended multiple console support through a vendor security package such as RACF, CA-TOPSECRET, and CA-ACF2. Check with your security administrator to determine if your site is using such a security package to restrict the use of extended multiple console support. If so, the Connect:Enterprise name must be authorized to issue any console commands specified in your application agent rules.

COMMAND Instruction Format

The following example illustrates the COMMAND instruction format. Default values for parameters and subparameters are underlined.

```
COMMAND TEXT='xxxx...xxxx',
  ERROR=CONTINUE|QUIT
```

COMMAND Instruction Parameters

The following table describes the COMMAND instruction parameters:

Parameter	Definition
TEXT='xxxx....xxxx'	Required. Specifies a console command, up to 112 bytes in length, to issue. You must enclose the value in single quotes. Any symbolic parameters embedded in the command are replaced by the proper values. This parameter is case sensitive. If the console command that is issued is a Connect:Enterprise command, Connect:Enterprise must run as a started task with MODIFY=YES specified in the ODF. To issue a command to Connect:Enterprise, the COMMAND text can use symbolic substitution to supply the Connect:Enterprise started task name. For example: TEXT='F &STCNAME,$$CONNECT L=ACCTING ID=&IDFIELD'
ERROR=CONTINUE | QUIT	Optional. Indicates the action for Connect:Enterprise to take if the command is not successfully issued. CONTINUE **=** specifies that the next instruction executes. QUIT = specifies that no subsequent instructions execute.

EXECUTE Instruction

Use the EXECUTE instruction to execute a user-written program. The EXECUTE instruction is valid for all rule types.

Connect:Enterprise loads the user program into memory for each execution. A branch instruction passes control to the program. User-written programs do not have to be reentrant. If you need for a user-written program to maintain information from one invocation to another, use the Initialization Exit Word to obtain and point to common storage. Refer to the documentation on the Initialization Exit in Chapter 3, *Using Connect:Enterprise Online Exits*. Because multiple copies of a user-written program execute simultaneously, use caution when updating information in common storage.

EXECUTE Instruction Format

The following describes the EXECUTE instruction format. Default values for parameters and subparameters are underlined.

```
EXECUTE PROGRAM=xxxxxx,
  ERROR=CONTINUE|QUIT
```

EXECUTE Instruction Parameters

The following table describes the EXECUTE instruction parameters:

Parameter	Definition
PROGRAM=xxxxxxxx	Required. Specifies the name of a user-written program that is called when this rule is executed. Link each program as a separate load module. This parameter is case sensitive.
ERROR=CONTINUE | QUIT	Optional. Indicates the action for Connect:Enterprise to take if the program returns a non-zero return code. CONTINUE = specifies that the next instruction executes. QUIT = specifies that no subsequent instructions execute.

Application Agent Parameters Passed to User-Specified Programs

Parameters passed to the user-specified program are addressed by register 1. The parameters and contents of the data field pointed to by the parameters for each type of application agent are discussed in the following sections.

Console Application Agent Parameters

The following table lists the parameters passed from the Console application agent:

R1 Points to Parameter List	Contains Address of	Data Field Contents
+0(4)	Fullword return code	A fullword return code field which the user program can set. Any non-zero value is treated as an error condition.
+4(4)	C$CON control block	Message text parsed into words
+8(4)	Initialization Exit Word	A fullword containing the information or address returned from the Initialization Exit. This value is X'00' if no Initialization Exit is used.

End of Batch Application Agent Parameters

The following table lists the parameters passed from the End of Batch application agent:

R1 Points to Parameter List	Contains Address of	Data Field Contents
+0(4)	Fullword return code	A fullword return code field which the user program can set. Any non-zero value is treated as an error condition. **Note:** The user program can update only the return code.

R1 Points to Parameter List	Contains Address of	Data Field Contents
+4(4)	Batch Control Record	4-byte address of a copy of the Batch Control Record (M$BCREC Macro)
+8(4)	Initialization Exit Word	A fullword containing the information or address returned from the Initialization Exit. This value is X'00' if no Initialization Exit is used.

Logging Application Agent Parameters

The following table lists the parameters passed from the Logging application agent:

R1 Points to Parameter List	Contains Address of	Data Field Contents
+0(4)	Fullword return code	A fullword return code field which the user program can set. Any non-zero value is treated as an error condition. **Note:** The user program can update only the return code.
+4(4)	Log Function Type	1-byte Log Function Type C'1' = Put New Log Record C'2' = Update Log Record
+8(4)	Address of Log Record Key	18-byte Log Record Key 1st byte C'A' = Auto Connect 1st byte C'B' = Remote Connect 1st byte C'D' = Auto Connect Queue
+12(4)	Address of Log Record Data	4-byte address of a copy of the Log Record. Refer to the M$ACREC macro for the Auto Connect DSECT. Refer to the M$LOGB macro for the Remote Connect DSECT. Refer to the M$DCREC macro for the Auto Connect Queue DSECT.
+16(4)	Address of Log Record Data Length	Halfword length of data portion of the log record.
+20(4)	Initialization Exit Word	A fullword containing the information or address returned from the Connect:Enterprise Initialization Exit. This value is X'00' if no Initialization Exit is used.

Scheduler Application Agent Parameters

The following table lists the parameters passed from the Scheduler application agent:

R1 Points to Parameter List	Contains Address of	Data Field Contents
+0(4)	Fullword return code	A fullword return code field which the user program can set. Any non-zero value is treated as an error condition. **Note:** The user program can update only the return code.
+4(4)	Initialization Exit Word	A fullword containing the information or address returned from the Initialization Exit. This value is X'00' if no Initialization Exit is used.

Wake Up Terminate Application Agent Parameters

The following table lists the parameters passed from the Wake Up Terminate application agent:

R1 Points to Parameter List	Contains Address of	Data Field Contents
+0(4)	Fullword return code	A fullword return code field which the user program can set. Any non-zero value is treated as an error condition. **Note:** The user program can update only the return code.
+4(4)	Input Parameter Structure GDS Header	4-byte address of a copy of the Application Agent Request Header (A$GDS Macro)
+8(4)	Input Parameter Structure Header	4-byte address of a copy of the Application Agent Request Header (C$H00 Macro)
+12(4)	Input Parameter Structure Trailer	4-byte address of a copy of the Application Agent Request Trailer (C$WKT Macro)
+16(4)	Initialization Exit Word	A fullword containing the information or address returned from the Connect:Enterprise Initialization Exit. This value is X'00' if no Initialization Exit is used.

MESSAGE Instruction

Use the MESSAGE instruction to issue a console message. The MESSAGE instruction is valid for all rule types.

MESSAGE Instruction Format

The following example illustrates the MESSAGE instruction format. Default values for parameters and subparameters are underlined.

```
MESSAGE TEXT='xxxx...xxxx',
  ROUTCODE=(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16),
  DESCCODE=(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16),
  CONEVENT=YES|NO,
  ERROR=CONTINUE|QUIT
```

MESSAGE Instruction Parameters

The following table describes the MESSAGE instruction parameters:

Parameter	Definition
TEXT='xxxx....xxxx'	Required. Specifies a user-defined console message of a maximum of 125 bytes. You must enclose the message in single quotes. Any symbolic parameters embedded in the message are replaced by the proper values. The message is routed to the operator console by default. The message is not prefixed with a message number, so you may want to assign your own in the first 8 bytes of the text. This parameter is case sensitive.
ROUTCODE=(1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16)	Optional. Specifies one or more routing codes to use when issuing the message. Delimit the codes with commas and enclose them in parentheses. The values of the codes follow: 1 = master console action 2 = master console information 3 = tape pool 4 = direct access pool 5 = tape library 6 = disk library 7 = unit record pool 8 = teleprocessing control 9 = system security 10 = system error/maintenance 11 = programmer information 12 = emulators 13 = reserved for user's choice 14 = reserved for user's choice 15 = reserved for user's choice 16 = reserved for user's choice

Parameter	Definition
DESCCODE=(1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16)	Optional. Specifies one or more descriptor codes to use when issuing the message. Delimit the codes with commas and enclose them in parentheses. The value of the codes follow: 1 = system failure 2 = immediate action required 3 = eventual action required 4 = system status 5 = immediate command response 6 = job status 7 = application/programmer response 8 = out-of-line message 9 = operator request 10 = dynamic status displays 11 = critical eventual action required 12 = reserved for future use 13 = reserved for future use 14 = reserved for future use 15 = reserved for future use 16 = reserved for future use
CONEVENT=YES|NO	Indicates whether the message will cause the Console application agent to be invoked. The default value, NO, specifies that the message will not cause the Console application agent to be invoked. YES specifies that the message will cause the Console application agent to be invoked. In addition, a Console agent rule must be coded with a SELECT statement specifying the message ID in the MSG01 parameter; otherwise, the invoked Console application agent will not process any instructions for this message. See *Detecting Application Agent Loops* on page 97 for more information.
ERROR=CONTINUE | QUIT	Optional. Indicates the action for Connect:Enterprise to take if the command is not successfully issued. CONTINUE = specifies that the next instruction executes. QUIT = specifies that no subsequent instructions execute.

NOP Instruction

The NOP instruction is a null instruction and performs no activity (no operation). The NOP instruction is valid for all rule types.

Use the NOP instruction to selectively make exclusions from generic selection criteria. To implement this use, define a rule that contains only the NOP instruction. Place the specific SELECT statements to exclude from the generic selection *before* the generic selection statement.

You can also use the NOP instruction with the application agent trace facility. The trace report displays application agent requests, but no activity is performed.

The following example illustrates the NOP instruction format. It has no parameters.

```
NOP
```

ROUTE Instruction

The ROUTE instruction tells Connect:Enterprise to sign on to the Connect:Direct node, submit a Process to Connect:Direct, and sign off from the Connect:Direct node. The ROUTE instruction is valid for the End-of-Batch rule type.

ROUTE Instruction Format

The following example illustrates the ROUTE instruction format. Default values for parameters and subparameters are underlined.

```
ROUTE PROC=membername
  PROCDSN=‘filename[(membername)]’,
  PNODE=‘primary_nodename’,
  PNODEID=(userid,password),
  SNODE=‘secondary_nodename’,
  SNODEID=(userid,password),
  TODSN=‘filename[(membername)]’,
  FTYPE=filetype,
  PROFDSN=‘filename[(membername)]’,
  NETMAP=‘filename[(membername)]’,
  NEWNAME=‘alias_process_name’,
  PRTY=nn,
  CNFMFIL=’filename[(membername)]’
  RPTSFIL='filename[(membername)]',
  CLASS=nnn,
  CASE=YES|NO,
  ESF=NO|YES,
  NOTIFY=userid|%USER,
  SIGNONUID=(userid,password),
  LOCAPPL=APPLID_prefix,
  LOGMODE=logmode,
  MBAPPL=APPC_APPL_ID,
  BUFSIZE=nnnnn,
  MAILBOXUID=(userid,password),
  TRANSPORT=SNA|TCP,nnnnn,nnn.nnn.nnn.nnn
  ODISP='disp',
  ODCB='dcb-parameter-string',
  OBLKSIZE='blksize',
  ODSORG='dsorg',
  OLRECL='lrecl',
  ORECFM='recfm',
  OSPACE='space',
  OUNIT='unit',
  ERROR=CONTINUE|QUIT
```

If a value needs to maintain lowercase, enclose the characters in lowercase within single quotes and set the keyword, CASE, to YES. The following format example illustrates how to specify lowercase values.

```
ROUTE PROC=membername
...
    PNODEID=('userid','password')
    SNODEID=('userid','password')
    SIGNONUID=('userid','password')
    CASE=YES
...
```

ROUTE Instruction Parameters

The following table describes the ROUTE instruction parameters:

Parameter	Definition
PROC=membername	Specifies the member name that contains the Connect:Direct Process to submit for the EXTRACT (the MB#EXT Process). The Process member resides in the //DMPUBLIB DD data set allocated by Connect:Enterprise. **Note:** You must specify either this parameter or the PROCDSN parameter, but not both.
PROCDSN=filename[(membername)]	Specifies the name of a sequential data set or a member of a partitioned data set containing the Connect:Direct Process to submit for the EXTRACT. This data set is dynamically allocated when the ROUTE instruction is executed. **Note:** You must specify either this parameter or the PROC parameter, but not both.
PNODE=primary_nodename	Optional. The 1–16 character Connect:Direct node name where the SUBMIT Process is submitted (the primary node). If you omit this parameter, the default is the Connect:Direct node defined as the local node in the DMNETMAP DD data set allocated in the Connect:Enterprise JCL. Either the DMNETMAP DD statement must point to a valid NETMAP data set or you must specify the NETMAP parameter. **Note:** Connect:Enterprise must have access to the Connect:Direct NETMAP data set.
PNODEID=(userid,password)	Optional. Specifies the security information used to sign onto the primary node, if required. It consists of the user ID, followed by the password. Enclosing the ID and password in single quotation marks enforces lowercase values, for example, PNODEID=('userid','password').
SNODE=secondary_nodename	Optional. Specifies the 1–16 character secondary node to use in the SUBMIT Process. This is usually the Connect:Direct node where the batch is sent.

Parameter	Definition
SNODEID=(userid,password)	Optional. The security information to sign onto the secondary node, if required. It consists of the user ID, followed by the password. Enclosing the ID and password in single quotation marks enforces lowercase values, for example, SNODEID=('userid','password').
TODSN=filename[(membername)]	Optional. Specifies the name to assign to the file created on the Connect:Direct node. The name must follow Connect:Direct file naming conventions. You can use symbolic values in the file name. Connect:Enterprise replaces the strings in the file name with the actual values. For a list of valid symbolic values, see the *Symbolic Substitution* on page 36.
FTYPE=file type	Optional. Specifies the file type parameter for the file created on the Connect:Direct node. See the *Connect:Direct Administration Guide* for valid file types.
PROFDSN=filename[(membername)]	Optional. Specifies the data set and member name of the EXTRACT profile that Connect:Direct uses to extract the batch from Connect:Enterprise. Only Connect:Direct accesses this data set. You do not have to allocate it to the Connect:Enterprise address space.
NETMAP=filename[(membername)]	The name of the Connect:Direct NETMAP to use. This parameter enables you to use different NETMAPS for different ROUTE instructions. **Note:** If a DMNETMAP DD statement is not specified in the Connect:Enterprise JCL, this parameter is required; otherwise, it is optional.
NEWNAME=alias_process_name	Optional. A Process alias used on all displays and reports of the submitted Process.
PRTY=nn	Optional. Specifies the 1–2 digit priority of the SUBMIT command. Specify a number from 1–15. If you omit this keyword, no priority is specified. This parameter specifies the Process priority in the Connect:Direct Transmission Control Queue. The higher the number the higher the priority. Connect:Direct uses this priority only for Process selection. It does not affect the priority during transmission.
CNFMFIL='filename[(membername)]'	Optional. Provides the fully qualified name of the file which receives the Confirmation SYSPRINT report. You can use symbolic values in the file name. Connect:Enterprise replaces the strings in the file name with the actual values. For a list of valid symbolic values, see *Symbolic Substitution* on page 36.

Parameter	Definition
RPTSFIL='filename[(membername)]'	Optional. Provides the fully qualified name of the file which receives the REPORTS file (Connect:Enterprise offline REPORTS DD file). This report is formatted the same as the offline utility reports and includes all of the same detail information provided in the STOUTL REPORTS DD. You can use symbolic values in the file name. Connect:Enterprise replaces the strings in the file name with the actual values. For a list of valid symbolic values, see *Symbolic Substitution* on page 36.
CLASS=nnn	Optional. Specifies the 1–3 digit CLASS that Connect:Direct uses to select a node-to-node session. Specify a number from 1–255.
CASE=YES | NO	Optional. Passed to the Connect:Direct Process. Specifies whether parameters associated with accounting data, user ID, password and data set names in the Connect:Direct SUBMIT command and in the PROCESS are case sensitive. The valid values are: YES = Parameters are case sensitive. NO = Parameters are not case sensitive. This is the default.
ESF=YES|NO	Optional. Specifies whether the Extended Submit Facility (ESF) is available for the current signon. The default value is NO. ESF enables you to submit Processes even if the DTF is not active. When signing on to a Connect:Direct DTF that is active, but with no VTAM APPLID available, ESF enables signon to the DTF. When you submit the Process, it is enabled and placed in the TCQ.
NOTIFY=userid | %USER	Optional. Specifies the user ID to notify of the request status. Valid values are: userid = Notifies the specified user ID to notify . %USER = Notifies the user who submitted the Process.
SIGNONUID=(userid,password)	Optional. Specifies the user ID and password to sign onto Connect:Direct. Each value has a maximum length of 64 characters. Enclosing the ID and password in single quotation marks enforces lowercase values, for example, SIGNONUID=('userid','password').
LOCAPPL=APPLID_prefix	Optional. Specifies the APPLID prefix that the API uses to sign onto Connect:Enterprise.
LOGMODE=logmode	Optional. Specifies the logmode.
MBAPPL=APPC_APPL_ID	Optional. Specifies the APPC APPLID that the API signs on to.
BUFSIZE=nnnnn	Optional. Specifies the VTAM buffer size.
MAILBOXUID=(userid,password)	Optional. Specifies the user ID and password to use to sign onto Connect:Enterprise.

Parameter	Definition
TRANSPORT=SNA | TCP,nnnnn,nnn.nnn.nnn.nnn	Optional. Specifies the protocol to use to sign onto Connect:Direct. The valid values are: SNA = Use SNA protocol. TCP = Use TCP protocol. If you use TCP, you must also specify the following subparameters: ◆ nnnnn = The port that the Connect:Direct TCP.API is listening on. ◆ nnn.nnn.nnn.nnn = The IP address of the open API port.

Parameter	Definition
ODISP=(NEW|MOD|RPL|SHR]|[,KEEP|CATLG]|[,KEEP|CATLG|DELETE])	Optional. Passed to the Connect:Direct Process. Specifies the status and disposition of the EXTRACT OUTFILE file on the receiving node Subparameters are as follows: First Subparameter specifies the status of the file before the Process executes. Only the OLD and RPL dispositions apply to VSAM files. Valid options for this subparameter are as follows: ◆ NEW specifies that the Process step will create the destination file. NEW is the default. ◆ OLD specifies that the destination file already exists. The Process will have exclusive control of the file. ◆ MOD specifies that the Process step will modify the file by appending data at the end of the file. ◆ RPL specifies that the destination file will replace any existing file or if none exists, will allocate a new file. ◆ SHR specifies that the destination file already exists. The file can be used simultaneously by another job or Process. Second Subparameter specifies the normal termination disposition. Valid destination file dispositions are as follows: ◆ KEEP specifies that the system keeps the file after the Process step completes. If DISP=(NEW,KEEP), a volume serial number must also be specified. ◆ CATLG specifies that the system keeps the file after the Process step completes and an entry is to be placed in the catalog. CATLG is the default. Third Subparameter specifies the disposition of the file after an abnormal Process step termination resulting in a non-zero completion code. This subparameter applies only to non-VSAM files. Valid destination file dispositions are as follows: ◆ KEEP specifies that the system keeps the file after the Process step terminates abnormally or with a non-zero return code. ◆ CATLG specifies that the system keeps the file after the Process step terminates abnormally and that an entry is to be placed in the catalog. ◆ DELETE specifies that the system deletes the file if the Process step terminates abnormally.

Parameter	Definition
ODCB=([BLKZIZE=no.-bytes][,DSORG=PS|PO][,LRECL=no.-bytes][,RECFM=record-fmt])	Optional. Passed to the Connect:Direct Process. Specifies the attributes to be used in allocating the EXTRACT OUTFILE. BLKSIZE = length in bytes of the block. The maximum block size is 32,760 bytes. DSORG = file organization. Supported file organizations are PO and PS. LRECL = record length in bytes. RECFM = format of the records in the EXTRACT output file. Any valid record format, such as F (Fixed), FA (Fixed ASA printer control), FB (Fixed Block), FBA (Fixed Block ANSI carriage control), FM (Fixed Machine code control character), U (Undefined), V (Variable), VB (Variable Block), VBA (Variable Block ASA printer control), VBM (Variable Block Machine code control character), VS (Variable Spanned) and VBS (Variable Block Spanned), can be specified. **Note:** The ODCB= parameter allows the entire DCB parameter string to be passed to the Connect:Direct Process. Alternatively, any of the 4 DCB subparameters (BLKSIZE, DSORG, LRECL, and RECFM) may be individually passed to the Connect:Direct Process by using the 4 corresponding ROUTE instruction parameters:OBLKSIZE, ODSORG, OLRECL, and ORECFM respectively.
OBLKSIZE=no.-bytes	Optional. Specifies the length in bytes of the block. The maximum length is 32,760 bytes.
ODSORG=PO|PS	Optional. Specifies the file organization. Supported file organizations are PO and PS.
OLRECL=no.-bytes	Optional. Specifies the record length in bytes.
ORECFM=record-format	Optional. Specifies the format of the records in the EXTRACT OUTFILE.

Parameter	Definition
OSPACE=(CYL|TRK|blk|av-rec-len, (prim,[sec],dir),,[RLSE],[CONTIG], [ROUND])	Optional. Specifies the amount of storage to be allocated for new files on the destination node. If SPACE is specified, the DISP of the destination file must be NEW. If SPACE is not specified in the Process or the TYPE file, and the DISP is NEW, the output file is allocated as follows: ◆ If no secondary space allocation exists on the input file, then the primary amount of space allocated (rather than used) is used to allocate the NEW output file. ◆ If secondary space exists on the input file, then space used (rather than allocated) is used to allocate the output file and it is allocated with secondary extents. ◆ If the AVGREC parameter is also specified in the TO clause of the COPY statement, the allocation of the data set is done on a record size basis instead of TRK, CYL, or blk. The TRK, CYL and blk subparameters are not valid when the AVGREC parameter is specified in the COPY TO statement. Valid choices for this parameter are as follows: CYL = space will be allocated by cylinder TRK = space will be allocated by track blk = space will be allocated by the average block length of the data. The system computes the number of tracks to be allocated. If the subparameter ROUND is also specified, the system allocates the space in cylinders. ROUND is preferred because allocation is performed on cylinders in a device-independent manner. If no space information is specified, allocation is in blocks, due to device dependencies. av-rec-length = average record length, in bytes, of the data. The system computes the BLKSIZE and the number of tracks to allocate. The record length must be a decimal value from 1-65535. prim = primary allocation of storage (number of units) sec = secondary allocation of storage (number of units) dir = number of PDS directory blocks to be created in the file RLSE =release of the unused storage allocated to the output file CONTIG = storage for the primary allocation must be contiguous ROUND= storage allocated by average block length is rounded to an integral number of cylinders
OUNIT=([unit-address|device-type| group-name]	Optional. Specifies the unit address, device type, or user-assigned group name where the file resides or will reside.
ERROR=CONTINUE | QUIT	Optional. Indicates the action for Connect:Enterprise to take if the ROUTE instruction is not successfully issued. Valid values are: CONTINUE = specifies that the next instruction executes. QUIT = specifies that no subsequent instructions execute.

Resolving Route Instruction Parameters When Communicating with Connect:Direct

When communicating with Connect:Direct, you control the value of certain variables sent from Connect:Enterprise using the ROUTE instruction parameters listed in the following table. Column one contains the ROUTE instruction parameter and column two lists the resulting text string that is sent to Connect:Direct as part of the SIGNON and SUBMIT commands.

Parameter	Keyword=*value* string used by Connect:Direct
NETMAP='filename[(member-name)]'	NETMAP=filename[(member-name)]
SIGNONUID=(userid,password)	USERID=(userid,password)
TRANSPORT=SNA | TCP,nnnnn,nnn,nnn,nnn,nnn	TRANSPORT=SNA | TCP COMADDR=nnnnn,nnn,nnn,nnn,nnn
PROC=member-name	PROC=member-name
PROCDSN='filename[(member-name)]'	DSN=filename[(member-name)]
PNODE=primary-nodename	PNODE=primary-nodename
PNODEID=(userid,password)	PNODEID=(userid,password)
SNODE=secondary-nodename	SNODE=secondary-nodename
SNODEID=(userid,password)	SNODEID=(userid,password)
NEWNAME='alias-process-name'	NEWNAME=alias-process-name
PRIORITY=nn	PRTY=nn
CLASS=nnn	CLASS=nnn
NOTIFY=userid | %USER	NOTIFY=userid | %USER
TODSN='filename[(member-name)]'	&&TODSN=filename[(member-name)]
PROFDSN='filename[(member-name)]'	&&PROFDSN=filename &&PROFMEM=member
FTYPE=filetype	&&FILETYP=filetype
LOCAPPL=applid-prefix	&&LOCAPPL=applid-prefix
MBAPPL=appc-appl-id	&&MBAPPL=appc-appl-id
LOGMODE-logmode	&&LOGMODE=logmode
BUFFSIZE=nnnnn	&&BUFFSIXE=nnnnn
MAILBOXUID=(userid,password)	&&USERID=userid &&PASSWORD=password

Parameter	Keyword=*value* string used by Connect:Direct
Note: The remaining variables used by Connect:Direct have no corresponding Connect:Enterprise ROUTE instruction parameter. The values in these variables are generated from the JCL or from the VCF or ODF definition for each SUBMIT command sent to Connect:Direct.	&&RMTID=mailbox-id (source: VCF – BC$KYID)
	&&BATCHNO=batch number (source: VCF – BC$KYBNO)
	&&BATCHID=batchid (source: VCF – BC$BCHID)
	&&MBNAME=mailbox-name (source: ODF – MBXNAME=)
	&&VSERVER=server-name (source: JCL – SC$SUBSN)
	&&VPF=vpf-filename (source: ODF – VPF=)

SNMPTRAP Instruction

Use the SNMPTRAP instruction to issue a SNMP V2 trap (a message that reports a problem or a significant event, formatted and encoded as defined in RFC 1442, and sent as a UDP datagram) to any IP address and port. The SNMPTRAP instruction is valid for all rule types.

Note: UDP is a protocol which does not guarantee delivery of IP packets. Therefore, keep in mind that there is a very low but non-zero probability that an SNMP trap can get lost in the Internet. Unfortunately, Connect:Enterprise cannot affect this probability in any way.

SNMPTRAP Instruction Format

The following example illustrates the SNMPTRAP instruction format. Default values for parameters and subparameters are underlined.

```
SNMPTRAP TEXT='xxxx...xxxx',
IPADDR=hostname | nnn.nnn.nnn.nnn,
PORT=162 | nnnnn,
GROUP1=ALARM | STATUS,
GROUP2=9999 | nnnn,
ERROR=CONTINUE | QUIT
```

SNMPTRAP Instruction Parameters

The following table describes the SNMPTRAP instruction parameters:

Parameter	Definition
TEXT=‘string’	Required. Specifies a user-defined string of a maximum 254 bytes. You must enclose the string in single quotes. Any symbolic parameters embedded in the string are replaced by the proper values. The resolved value can be up 1024 bytes. The text will be encoded into a trap varbind pair, using OID 1.3.6.1.4.1.1733.4.5.11 which is defined in the Management Information Base as ceSNMPTRAPtext. See the CE.MIB file on the distribution tape.

Parameter	Definition
IPADDR=hostname | nnn.nnn.nnn.nnn	Required. Specifies the IPADDR to send the trap to. It can be either in dotted decimal format or a more user-friendly hostname format. The maximum length is 64 bytes. If rule tracing is active, both formats are shown. If the dotted decimal format is specified, the first value in a reverse-hostname lookup is shown.
PORT=162 | nnnnn	Optional. Specifies the port number to send the trap to. It must be a number from 1 to 65535.
GROUP1=ALARM | STATUS	Optional. Specifies the first of two parameters which provides partial user specification of the snmpTrapOID data object (which is an OID). This parameter sets a middle level of the snmpTrapOID OID to either 1 or 2. ALARM is resolved to the digit 1 and STATUS to 2, thus providing a way to categorize a trap using these two SNMP standard types. If desired, user-defined labels for the resulting snmpTrapOID OID can be added to a MIB imported by the trap receiver. Sterling Commerce does not distribute labels for the nearly 100,000 possible application agent snmpTrapOID OIDs that can be generated by combinations of the agent type, GROUP1 and GROUP2 parameters.
GROUP2=9999 | nnnn	Optional. Specifies the second of two parameters which provides partial user specification of the snmpTrapOID data object (which is also an OID). This parameter sets the final level of the snmpTrapOID OID to a number between 1 and 9999. If desired, user-defined labels for the resulting snmpTrapOID OID can be added to a MIB imported by the trap receiver. Sterling Commerce does not distribute labels for the nearly 100,000 possible application agent snmpTrapOID OIDs that can be generated by combinations of the agent type, GROUP1 and GROUP2 parameters.
ERROR=CONTINUE | QUIT	Optional. Indicates the action for Connect:Enterprise to take if the SNMPTRAP instruction is not successfully issued. CONTINUE specifies that the next instruction executes. QUIT specifies that no subsequent instructions execute.

SNMPTRAP Layout and Contents

The layout of a SNMP V2 trap is defined by RFC (Request for Comments, that is, Internet protocol standards) 1442. Generally, the layout consists of a header followed by multiple varbind pairs, each of which consists of an object identifier (OID) – a multilevel dotted decimal string – and a data object. The prefix for all Sterling Commerce OIDs is 1.3.6.1.4.1.1733. The next level determines the Sterling Commerce product. For Connect:Enterprise, it is 4. OIDs are given labels in a Management Information Base (MIB). A copy of the entire Sterling Commerce MIB is in the CE.MIB file on the distribution tape, but the portions not relating to Connect:Enterprise may be out of date. Contact Sterling Commerce Customer Support if you suspect you may not have the most recent MIB. The relevant parts of the Connect:Enterprise MIB are summarized in the following table:

Label	Object Identifier (OID)
centerpriseOS390	1.3.6.1.4.1.1733.4
ceAlarmTraps	1.3.6.1.4.1.1733.4.1
ceApplAgentsAlarm	1.3.6.1.4.1.1733.4.1.1
ceAAeobAlarm	1.3.6.1.4.1.1733.4.1.1.1
ceAAlogAlarm	1.3.6.1.4.1.1733.4.1.1.2
ceAAwktAlarm	1.3.6.1.4.1.1733.4.1.1.3
ceAAschAlarm	1.3.6.1.4.1.1733.4.1.1.4
ceAAconAlarm	1.3.6.1.4.1.1733.4.1.1.5
ceStatusTraps	1.3.6.1.4.1.1733.4.2
ceApplAgentsStatus	1.3.6.1.4.1.1733.4.2.1
ceAAeobStatus	1.3.6.1.4.1.1733.4.2.1.1
ceAAlogStatus	1.3.6.1.4.1.1733.4.2.1.2
ceAAwktStatus	1.3.6.1.4.1.1733.4.2.1.3
ceAAschStatus	1.3.6.1.4.1.1733.4.2.1.4
ceAAconStatus	1.3.6.1.4.1.1733.4.2.1.5
ceTrapMembers	1.3.6.1.4.1.1733.4.5
ceDateTimeTrapSent	1.3.6.1.4.1.1733.4.5.3
ceMVSsystemName	1.3.6.1.4.1.1733.4.5.4
ceJobName	1.3.6.1.4.1.1733.4.5.5
ceJobId	1.3.6.1.4.1.1733.4.5.6
ceApplAgentType	1.3.6.1.4.1.1733.4.5.7
ceRuleMemberName	1.3.6.1.4.1.1733.4.5.8
ceRelativeSelectStmt	1.3.6.1.4.1.1733.4.5.9
ceRuleName	1.3.6.1.4.1.1733.4.5.10
ceSNMPTRAPtext	1.3.6.1.4.1.1733.4.5.11
ceVersion	1.3.6.1.4.1.1733.4.5.12

The SNMP trap contains the following varbind pairs (in order of appearance):

Label	Object Identifier (OID)	Data Object
sysUpTime	1.3.6.1.2.1.1.3.0	The number of seconds Connect:Enterprise has been up

Label	Object Identifier (OID)	Data Object
snmpTrapOID	1.3.6.1.6.3.1.1.4.1.0	The trap OID generated from the agent type, GROUP1 and GROUP2
ceDateTimeTrapSent	1.3.6.1.4.1.1733.4.5.3	The local system date, time and UTC offset from CVTTZ when the trap was sent
ceSNMPTRAPtext	1.3.6.1.4.1.1733.4.5.11	The resolved value of the SNMPTRAP TEXT parameter
ceMVSsystemName	1.3.6.1.4.1.1733.4.5.4	The MVS system name from CVTSNAME
ceJobName	1.3.6.1.4.1.1733.4.5.5	The jobname of the Connect:Enterprise main task
ceJobId	1.3.6.1.4.1.1733.4.5.6	The JES2 jobid of the Connect:Enterprise main task
ceApplAgentType	1.3.6.1.4.1.1733.4.5.7	The application agent type (CON, EOB, LOG, SCH, WKT)
ceRuleMemberName	1.3.6.1.4.1.1733.4.5.8	The current rule set member name
ceRelativeSelectStmt	1.3.6.1.4.1.1733.4.5.9	The matching SELECT statement number in the rule set
ceRuleName	1.3.6.1.4.1.1733.4.5.10	The name of the RULE containing the SNMPTRAP instruction
ceVersion	1.3.6.1.4.1.1733.4.5.12	A string containing the Connect:Enterprise version

STATFLG Instruction

Use the STATFLG instruction to change the status flags for the selected batches on the VSAM batch files. The batch status is changed immediately, prior to executing the next instruction.

The STATFLG instruction is valid for the End of Batch and Wake Up Terminate rule types.

STATFLG Instruction Format

The following example illustrates the STATFLG instruction format. Default values for the parameters and subparameters are underlined.

```
STATFLG ONFLAGS=(REQUESTABLE,DELETED,TRANSMITTED,EXTRACTED,MULTXMIT),
OFFFLAGS=(REQUESTABLE,DELETED,TRANSMITTED,EXTRACTED,MULTXMIT),
ERROR=CONTINUE|QUIT

or (short form)

STATFLG ONFLAGS=(R,D,T,E,M),
OFFFLAGS=(R,D,T,E,M),
ERROR=CONTINUE|QUIT

or (short form)

STATFLG ON=(R,D,T,E,M),
OFF=(R,D,T,E,M),
ERROR=CONTINUE|QUIT
```

STATFLG Instruction Parameters

The following table describes the STATFLG instruction parameters. You must specify either ONFLAGS or OFFFLAGS. Both parameters can be used in a single STATFLG instruction. If both are used, the flags are processed in the order specified in the instruction.

Note: If you turn on the 'M' (MULTXMIT) flag, the 'R' (REQUESTABLE) flag is automatically turned on. If you turn off the 'R' (REQUESTABLE) flag, the 'M' (MULTXMIT) flag is automatically turned off.

Parameter	Definition
ONFLAGS	Optional. Specifies which batch status flags are turned on for the selected batch. Separate multiple values by commas. REQUESTABLE = A remote site can request the batch or a host-initiated Auto Connect can transmit the batch. DELETED = The batch is flagged for deletion. TRANSMITTED = The batch was transmitted to a remote site. EXTRACTED = The batch was extracted from the VSAM batch files. MULTXMIT = The batch is available for multiple transmission and transmission to any remote site. **Note:** You must specify ONFLAGS or OFFFLAGS. Both can be used in a single instruction.

Parameter	Definition
OFFFLAGS	Optional. Specifies which batch status flags are turned off for the selected batch. Separate multiple values by commas. REQUESTABLE = A remote site can request the batch or a host-initiated Auto Connect can transmit the batch. DELETED = The batch is flagged for deletion. TRANSMITTED = The batch was transmitted to a remote site. EXTRACTED = The batch was extracted from the VSAM batch files. MULTXMIT = The batch is available for multiple transmission and transmission to any remote site. **Note:** You must specify ONFLAGS or OFFFLAGS. Both can be used in a single instruction.
ERROR=CONTINUE | QUIT	Optional. Indicates the action for Connect:Enterprise to take if the STATFLG instruction is not successfully issued. CONTINUE = specifies that the next instruction executes. QUIT = specifies that no subsequent instructions execute.

SUBMIT Instruction

Use the SUBMIT instruction to submit a job to the internal reader. The SUBMIT instruction is valid for End of Batch, Wake Up Terminate, and Logging rule types.

SUBMIT Instruction Format

The following example illustrates the SUBMIT instruction format. Default values for parameters and subparameters are underlined.

```
SUBMIT MEMBER=xxxxxxxx,
  ERROR=CONTINUE|QUIT
```

SUBMIT Instruction Parameters

The following table describes the SUBMIT instruction parameters:

Parameter	Definition
MEMBER=xxxxxxxx	Required. Specifies the member in the //RULESJCL DD data set that is read and written to the internal reader. Any symbolic parameters embedded in columns 1–71 of the JCL are replaced by the proper values. This parameter is case sensitive.
ERROR=CONTINUE | QUIT	Optional. Indicates the action for Connect:Enterprise to take if the JCL cannot be submitted to the internal reader. CONTINUE = specifies that the next instruction executes. QUIT = specifies that no subsequent instructions execute.

WAKEUP Instruction

Use the WAKEUP instruction to indicate that Wake Up Initiate processing is started for a specific batch. A Wake Up notification is sent to CICS for the batch just added to Connect:Enterprise. The parameters are passed to CICS in the Wake Up request message (C$W00 IPS).

The WAKEUP instruction is valid for the End of Batch rule type.

WAKEUP Instruction Format

The following example illustrates the WAKEUP instruction format. Default values for parameters and subparameters are underlined.

```
WAKEUP CICSDEFN=TRANSACTION|PROGRAM,
    CICSSYSID=xxxx,
    CICSPGMNM=xxxxxxxx,
    CICSTRANID=xxxx,
    CICSTERMID=xxxx,
    CICSUSER=xxxxxxxx,
    ERROR=CONTINUE|QUIT
```

WAKEUP Instruction Parameters

The following table describes the WAKEUP instruction parameters:

Parameters	Definition
CICSDEFN=TRANSACTION/PROGRAM	Required. Informs CICS to execute either a transaction or a program with this notification.
CICSSYSID=xxxx	Required. Specifies the 4-byte ID of the CICS system that receives this notification. This value is defined in the CICS System Initialization Table uniquely identifying the CICS region.
CICSPGMNM=xxxxxxxx	Optional. Specifies the 8-byte program name to invoke as a result of this notification. **Note:** This parameter is required if CICSDEFN=PROGRAM.
CICSTRANID=xxxx	Optional. Specifies the 4-byte transaction ID to invoke as a result of this notification. **Note:** This parameter is required if CICSDEFN=TRANSACTION.
CICSTERMID=xxxx	Optional. Specifies the 4-byte terminal ID that is passed with this notification. This parameter is only valid if CICSDEFN=TRANSACTION.
CICSUSER=xxxxxxxx	Specifies the 8-byte user ID that is passed with this notification. This parameter is only valid if CICSDEFN=TRANSACTION and CICSTERMID is specified.

Parameters	Definition
ERROR=CONTINUE | QUIT	Optional. Indicates the action for Connect:Enterprise to take if the WAKEUP instruction cannot execute. CONTINUE = specifies that the next instruction executes. QUIT = specifies that no subsequent instructions execute.

SELECT Statement

This section describes the parameters that you can use with the SELECT statement and the guidelines for creating SELECT statements:

- In addition to the RULE parameter, you must specify a minimum of one parameter for each SELECT statement that you define.
- Some parameter values are case sensitive and are noted as such. Case sensitive means that upper and lower case values are retained. For example, if you specify the Mailbox ID selection parameter as ID=testID01, a match only occurs for testID01, not TESTID01, Testid01, and so on.
- Specify each SELECT statement to point to up to eight rules. You can also specify multiple SELECT statements to reference a single rule.
- To ensure accurate processing, list the SELECT statements that contain detailed selection criteria before the SELECT statements that contain general selection criteria.

The following table lists all SELECT statement parameters and indicates the type of application agent rule that for which each is valid. Logging rules are further classified by log record type (RECTYPE=). O indicates the parameter is optional; R indicates the parameter is required.

SELECT Parameter	Application Agent Rule Type								
	CON	EOB	SCH	WKT	LOG (ACS)	LOG (ACD)	LOG (RCS)	LOG (RCD)	LOG (ACQ)
ACFUNC						O			
ACQREASON									O

1 CON=Console
EOB=End of Batch
SCH=Scheduler
WKT=Wake Up Terminate

2 LOG=Logging
ACS=Auto Connect Summary Log Record
ACD=Auto Connect Detail Log Record
RCS=Remote Connect Summary Log Record
RCD=Remote Connect Detail Log Record
ACQ=Auto Connect Queue Log Record

SELECT Parameter	Application Agent Rule Type								
	CON	EOB	SCH	WKT	LOG (ACS)	LOG (ACD)	LOG (RCS)	LOG (RCD)	LOG (ACQ)
BATCHID		O		O		O		O	
CALENDAR			O						
CASE_SENSITIVE		O		O	O	O	O	O	O
CICSPGMNM				O					
CICSTRANID				O					
DESCRIPTION			O						
FAILCODE					O	O		O	
ID		O		O		O		O	
LINE		O		O		O		O	
LISTNAME					O	O			O
LOGFUNC					O	O	O	O	O
MSG01	R								
MSG02–MSG96	O								
ORIGIN				O					
RCFUNC								O	
RECTYPE					R	R	R	R	R
REMOTE		O		O		O	O	O	
RTNCODE				O					
RTYPE					O	O	O	O	O
RULE	R	R	R	R	R	R	R	R	R
STATOR		O		O					
STATUS		O		O					
TIME			O						

1 CON=Console
EOB=End of Batch
SCH=Scheduler
WKT=Wake Up Terminate

2 LOG=Logging
ACS=Auto Connect Summary Log Record
ACD=Auto Connect Detail Log Record
RCS=Remote Connect Summary Log Record
RCD=Remote Connect Detail Log Record
ACQ=Auto Connect Queue Log Record

SELECT Parameter	Application Agent Rule Type								
	CON	EOB	SCH	WKT	LOG (ACS)	LOG (ACD)	LOG (RCS)	LOG (RCD)	LOG (ACQ)
ULTEXT01–ULTEXT96						O			
WILD_CARD		O		O	O	O	O	O	O
WILD_CARD_MULTI_CHAR		O		O	O	O	O	O	O
WILD_CARD_SINGLE_CHAR		O		O	O	O	O	O	O

1 CON=Console
EOB=End of Batch
SCH=Scheduler
WKT=Wake Up Terminate

2 LOG=Logging
ACS=Auto Connect Summary Log Record
ACD=Auto Connect Detail Log Record
RCS=Remote Connect Summary Log Record
RCD=Remote Connect Detail Log Record
ACQ=Auto Connect Queue Log Record

SELECT Statement Format

The following describes the SELECT statement format and parameters. Default values for parameters and subparameters are underlined. A description of each parameter and subparameter follows the SELECT statement format.

```
SELECT RULE=(xxxxxxxx[xxxxxxxx,....,xxxxxxxx])
  †RECTYPE=ACSUMMARY|ACDETAIL|ACQUEUE|RCSUMMARY|RCDETAIL,
  ACFUNC=(CONN,DISC,LOGON,RECV,SEND,SESSEN,SESSST,ULOG),
  ACQREASON=(LINE,ACTIVE,SESSION,THREAD),
  BATCHID='xxxx....xxxx'|"yyyy....yyyy",
  CALENDAR=xxxxxxxx
  CICSPGMNM=xxxxxxxx
  CICSTRANID=xxxx
  DESCRIPTION='xxx.xxx'
  FAILCODE=nnn|nnn-nnn|(nnn,...,nnn),
  ID=xxxxxxxx,
  LINE=xxxxxxxx,
  LISTNAME=xxxxxxxx,
  LOGFUNC=(NEW,UPDATE),
  MSG01=xxxxxxxx|(xxxxxxxx,xxxxxxxx...)
  MSG02=xxxxxxxx|(xxxxxxxx,xxxxxxxx...)
...
  MSG96=xxxxxxxx|(xxxxxxxx,xxxxxxxx...)
  ORIGIN=ALL|EOBRULES|EOBEXIT
  RCFUNC=(ADD,CONN,DEL,DIR,DISC,NOADD,REQ,SIGNON),
  REMOTE=xxxxxxxx
  RTNCODE=nnnn|nnnn-nnnn|(nnnn,...,nnnn)
  RTYPE=(BSC,FTP,SNA)
  STATOR=(ADDED,BSC,COLLECTED,DELETE,EBCDIC,EXTRACTED,
  FILE_STUCTURE,FTP,INCOMPLETE,MULTXMIT,
  NONTRANSMITTABLE,REQUESTABLE,SNA,SSL,
  TRANSPARENT,TRANSMITTED,UNEXTRACTABLE)
  STATUS=(ADDED,BSC,COLLECTED,DELETE,EBCDIC,EXTRACTED,
  FILE_STUCTURE,FTP,INCOMPLETE,MULTXMIT,
  NONTRANSMITTABLE,REQUESTABLE,SNA,SSL,
  TRANSPARENT,TRANSMITTED,UNEXTRACTABLE)
  TIME=hh:mm|(hh:mm,hh:mm...)
  ULTEXT01=xxxxxxxx|(xxxxxxxx,xxxxxxxx...)
...
  ULTEXT96=xxxxxxxx|(xxxxxxxx,xxxxxxxx...)
```

SELECT Statement Parameters

The following are required and optional parameters you can use in SELECT statements:

Parameters	Definition
RULE=(xxxxxxxx[xxxxxxxx, ...,xxxxxxxx])	Required. Valid for all rules. Specifies the rule names to process when a match occurs on all criteria specified in the SELECT statement. xxxxxxxx specifies the 1–8 character names of the rules to process. A minimum of one rule name must be specified, but up to eight rules can be listed. The rules are processed in the order specified.

Parameters	Definition
RECTYPE=ACSUMMARY | ACDETAIL | ACQUEUE | RCSUMMARY | RCDETAIL	Required. Valid only for Logging rules. Specifies the log record type to process. You can specify only one log record type. ACSUMMARY = Processes Auto Connect Summary records. ACDETAIL = Processes Auto Connect Detail records. ACQUEUE = Processes Auto Connect Queue records. RCSUMMARY = Processes Remote Connect Summary records. RCDETAIL = Processes Remote Connect Detail records.
ACFUNC=(SEND,RECV)	Optional. Valid only for Logging rules and when RECTYPE=ACDETAIL. Specifies the Auto Connect function to perform. **Note:** If this parameter is omitted, all function request types are assumed. To reduce internal storage and other resource utilization and overhead, it is recommended that you specify which auto connect functions to perform. CONN = Connect on control port (FTP only). DISC = Disconnect on control port (FTP only). LOGON = Processes USER and PASS commands (FTP only). RECV = Performs batch collection. SEND = Performs batch transmission. SESSEN = Ends script execution (FTP only). SESSST = Starts script execution (FTP only). ULOG = Processes USERLOG command to log user data (FTP only) **Note:** See *Connect:Enterprise for z/OS Administration Guide* for more information on host commands used in script processing and FTP auto connect sessions, including USERLOG.
ACQREASON=(LINE,ACTIVE, SESSION,THREAD)	Optional. Valid for Logging rules. Specifies the reason for queueing the Auto Connect request. If omitted, all Auto Connect queue reasons are assumed. Specify this parameter only if RECTYPE=ACQUEUE. ACTIVE = Selects Auto Connects queued because they were already initiated and running. LINE = Selects Auto Connects queued for BSC lines. SESSION = Selects Auto Connects queued because no SNA session was established. THREAD = Selects Auto Connect queued because no FTP threads were available.

Parameters	Definition
BATCHID='xxxx...xxxx' | "yyyy...yyyy"	Optional. Valid for End of Batch, Wake Up Terminate, and Logging rules. Specifies the specific or generic user batch ID of the batch or batches to process. This parameter is case sensitive. 'xxxx...xxxx' **=** Full user batch ID of the batch or batches to process. Ensure that the user batch ID is 1–64 characters long and conforms to the standards at your site. The user batch ID can contain blanks and you must enclose it in single quotes. "yyyy...yyyy" **=**Generic user batch ID used for selecting the batch or batches for processing. Ensure that the generic user batch ID is 1–63 characters in length and conforms to the standards at your site. The user batch ID can contain blanks and you must enclose it in double quotes. BID= is the short form of this parameter. **Note:** If you use this parameter for Logging rules, it is only valid for RECTYPE=ACDETAIL | RCDETAIL. **Note:** When selecting by batch ID, you may optionally use full wildcard search capability, instead of performing an exact string comparison on the 64-character Batch ID (or a generic Batch ID prefix). When full wildcard support is used, the BATCHID= value is treated as a pattern, in which wildcard characters can be used to mask out one or more portions of the Batch ID. To activate wildcard checking, the WILD_CARD=BID parameter must be specified. See the following parameters for a complete description on how to activate wildcard checking: ◆ CASE_SENSITIVE ◆ WILD_CARD= ◆ WILD_CARD_MULTI_CHAR= ◆ WILD_CARD_SINGLE_CHAR=
CALENDAR=xxxxxxxx	Optional. Valid only in a Scheduler Agent rule. Specifies the 1 to 8-character name of a calendar defined in the ODF. The calendar can specify dates or days of the week identified for activation or exclusion. When a time of day occurs that matches a value in the SELECT statement's TIME parameter, and the calendar identifies the current date or day of the week for activation, it is considered a match. If the time matches but the calendar identifies the current date or day of the week for exclusion (exception day or date), it is NOT considered a match. If omitted, only the time of day determines when a match occurs. For more information on defining calendars, refer to Chapter 7, *Configuring *CALENDAR Records*, in the *Connect:Enterprise for z/OS Administration Guide*.

Parameters	Definition
CASE_SENSITIVE=YES|NO	Optional. Valid for End of Batch, Wake Up Terminate, and Logging rules. Specifies whether characters are to be treated as case-sensitive when performing a wild card comparison. The default value is yes. YES = Treats the tested string and mask pattern as case-sensitive, that is, leaves the input value as is when performing the wildcard comparison. NO = Uppercases both the tested character string and the mask pattern, prior to performing the wild card comparison. **Note:** You must specify the WILD_CARD=BID parameter when using this parameter.
CICSPGMNM=xxxxxxxx	Optional. Valid only for Wake Up Terminate rules. Specifies the 8-byte CICS program name that was invoked as a result of the Wake Up notification. Specify a generic name with an asterisk, such as CICSPGMNM=PGM99*.
CICSTRANID=xxxx	Optional. Valid only for Wake Up Terminate rules. Specifies the 4-byte CICS transaction ID that was invoked as a result of the Wake Up notification. Specify a generic name with an asterisk, such as CICSTRANID=TR*.
DESCRIPTION='xxx...xxx'	Optional. Valid only for Scheduler Agents rules. Specifies a 1–50 character string, which can be used to provide a description of what the SELECT statement will process. The string can contain blanks, if enclosed in quotes. The description is displayed on the user interface Scheduler Selection List panel used to invoke a Scheduler SELECT statement.
FAILCODE=nnn | nnn–nnn | (nnn,.....,nnn)	Optional. Valid for Logging rules. Specifies a 1–3 digit Auto Connect or Remote Connect failure code or range of failure codes, as documented in *Connect:Enterprise for z/OS Messages and Codes Guide*. Leading zeros are not required. Can also include user-defined fail codes in the 240-255 range. **Note:** The FAILCODE value of each PUT NEW ACSUMMARY log record is 001 (connect process was started, but not completed). Therefore, RECTYPE=ACSUMMARY with LOGFUNC=NEW and FAILCODE=001 is not an error. This combination of SELECT criteria occurs each time an Auto Connect begins in Connect:Enterprise. nnn **=** Specifies a connect failure code nnn–nnn **=** Specifies a range of connect failure codes nnn,....,nnn **=** Specifies a list of up to 16 failure codes.

Parameters	Definition
ID=xxxxxxxx	Optional. Valid for End of Batch, Wake Up Terminate, and Logging rules. Specifies the 1–8 byte Mailbox ID. Specify a generic name with an asterisk, such as ID=BRCH*. The value of this parameter is case sensitive. **Note:** If used for Logging rules, this parameter is valid only for RECTYPE=ACDETAIL | RCDETAIL.
LINE=xxxxxxxx	Optional. Valid for End of Batch, Wake Up Terminate, and Logging rules. Specifies the 1–8 byte BSC line ID. Specify a generic name with an asterisk, such as LINE=LINE0*. The value of this parameter is case sensitive. **Note:** If used for Logging rules, this parameter is valid only for RECTYPE=ACDETAIL | RCDETAIL.
LISTNAME=xxxxxxxx	Optional. Valid only for Logging rules. Specifies the Auto Connect list name. Specify a generic name with an asterisk, such as LISTNAME=LIST0*. The value of this parameter is case sensitive. **Note:** This parameter is valid only for RECTYPE=ACSUMMARY | ACDETAIL | ACQUEUE.
LOGFUNC=(NEW,UPDATE)	Optional. Valid only for Logging rules. Specifies the log function type. If this parameter is omitted, both log function types are assumed. NEW **=** Specifies selection for a new log record. UPDATE **=** Specifies selection for an update log record. **Note:** To understand what function to base your selection on, review the *Log Exit Requirements* on page 150.
MSG01=xxxxxxxx|(xxxxxxxx, xxxxxxxx...)	Required. Valid only for in a Console Application Agent rules. Specifies the first blank-delimited word in the message and is commonly known as the message ID (MSGID). Specify 1—16 words (up to 125 bytes each), separated by commas and enclosed in parentheses if more than one word is specified. The comparison is not case sensitive. No wildcards are supported.

Parameters	Definition
MSG02–MSG96=xxxxxxxx| (xxxxxxxx,xxxxxxxx...)	Optional. Valid only in Console Application Agent rules. Specifies the 2nd – 96th words in the message. Specify 1–16 words (up to 125 bytes each), separated by commas and enclosed in parentheses if more than one word is specified. The comparison is case-sensitive. Wildcards are supported. An asterisk (*) represents any 0-125 byte string. A percent (%) represents any one character. The rule is selected if, for every MSGnn parameter specified, one of the parameter's values matches the associated word in the message. If any MSGnn parameter is not specified, the associated word in the message is not part of the selection process. It is not necessary to specify contiguous MSGnn parameters; for example, you can specify MSG03 and either specify or omit MSG02. However, MSG01 must always be specified. Message words matched to MSG02-96 can be delimited by the following characters in addition to blanks: period, comma, equal sign, open parenthesis and close parenthesis. These same delimiters are used to break up the symbolic &MSG into symbolics &MSG02 – &MSG96. **Note:** When the SELECT statement is processed, the first blank delimited word in the message text string is upper cased and set into MSG01. The remainder of MSG is then translated so that the following characters are converted to blanks: period, comma, equals sign, open parenthesis, and close parenthesis. Lastly, the translated string is broken into blank-delimited words, with MSG02 taking the value of the word after MSG01, and so on, until MSG96.
ORIGIN=ALL | EOBRULES | EOBEXIT	Optional. Valid only for Wake Up Terminate rules. Specifies the origin of the Wake Up Initiate transaction. Use this parameter to select Wake Up Terminate rules based on the origin of the Wake Up Initiate request. ALL**=** Specifies all points of Wake Up Initiate origin. This value includes both End of Batch application agent rules processing and End of Batch user exit. EOBRULES **=** Specifies the Wake Up Initiate request originated as a result of End of Batch rules processing (in other words, generated by the WAKEUP instruction in the End of Batch rules). EOBEXIT **=** Specifies the Wake Up Initiate request originated as a result of the End of Batch user exit. For example, the request was generated by the data repository as a result of the user exit setting the action code indicator to ‘W’, issue Wake Up Initiate.

Parameters	Definition
RCFUNC=(ADD,CONN,DEL, DIR,DISC,NOADD,REQ, SIGNON)	Optional. Valid only for Logging rules and when RECTYPE=RCDETAIL. Specifies the type of function requested by the remote node. **Note:** If this parameter is omitted, all function request types are assumed. To reduce internal storage and other resource utilization and overhead, it is recommended that you specify which remote connect functions to perform. ADD = Batch containing a $$ADD record CONN = Connect DEL = $$DELET request from the remote. DIR = $$DIRECTORY request from the remote DISC = Disconnect NOADD = Batch without a $$ADD record REQ = $$REQUEST from the remote SIGNON = BSC signon request from the remote
REMOTE=xxxxxxxx	Optional. Valid for End of Batch, Wake Up Terminate, and Logging rules. Specifies the 1–8 byte name of the remote site. Specify a generic name with an asterisk, such as REMOTE=RMT*. The value of this parameter is case sensitive. **Note:** If used for Logging rules, this parameter is valid only for RECTYPE=ACDETAIL | RCDETAIL | RCSUMMARY.
RTNCODE=nnnn | nnnn–nnnn | (nnnn,.....,nnnn)	Optional. Valid for Wake Up Terminate rules. Specifies the 4-digit hex return code or range of return codes set by CICS in the Wake Up Terminate IPS header as specified in *Connect:Enterprise for z/OS Messages and Codes Guide*. Leading zeros are required. nnnn = specifies a 4-digit hex return code. nnnn–nnnn = specifies a range of 4-digit hex return codes. nnnn,.....,nnnn = specifies a list of up to 16 return codes.
RTYPE=(BSC,FTP,SNA)	Optional. Valid only for Logging Application Agent rules. Specifies the type of remote connection. BSC = specifies a BSC connection FTP = specifies a FTP connection SNA = specifies a SNA connection

Parameters	Definition
STATOR=(ADDED,BSC, COLLECTED,DELETE,EBCDIC, EXTRACTED, FILE_STRUCTURE, FTP,INCOMPLETE,MULTXMIT, NONTRANSMITTABLE, REQUESTABLE,SNA,SSL, TRANSPARENT, TRANSMITTED, UNEXTRACTABLE)	Optional. Valid for End of Batch and Wake Up Terminate rules. Specifies one or more status flags for batches selected for processing. Each batch with any of the specified STATOR flags is processed. STATUS and STATOR are mutually exclusive. ADDED = batch was added offline. BSC = batch was collected through a BSC transmission. COLLECTED = batch was collected online. DELETE = batch is flagged for deletion. EBCDIC = batch was collected through the APPC API. EXTRACTED = batch was extracted. FILE_STRUCTURE = batch is stored as non-record oriented. FTP = batch was collected through an FTP transmission. INCOMPLETE = batch is incomplete (not successfully collected). MULTXMIT = batch is available for multiple transmission. NONTRANSMITTABLE = batch cannot be transmitted. REQUESTABLE = batch is available for online requests by remote sites or for transmission by host-initiated Auto Connect sessions. SNA = batch was collected through an SNA transmission. SSL = batch was collected over a secured connection using SSL or TLS. TRANSMITTED = batch was transmitted. TRANSPARENT = batch contains transparent data. UNEXTRACTABLE = batch cannot be extracted. The abbreviated form for this parameter is: STATOR=(A,B,C,D,EB,EX,FI,FTP,I,M,N,R,SN,SS,TRANSP,TRANSM,U)

Parameters	Definition
STATUS=(ADDED,BSC, COLLECTED,DELETE,EBCDIC, EXTRACTED, FILE_STRUCTURE, FTP,INCOMPLETE,MULTXMIT, NONTRANSMITTABLE, REQUESTABLE,SNA,SSL, TRANSPARENT, TRANSMITTED, UNEXTRACTABLE)	Optional. Valid for End of Batch and Wake Up Terminate rules. Specifies one or more status flags for batches selected for processing. Only those batches with all the specified batch status flags are processed. STATUS and STATOR are mutually exclusive. ADDED = batch was added offline. BSC = batch was collected through a BSC transmission. COLLECTED = batch was collected online. DELETE = batch was flagged for deletion. EBCDIC = batch was collected through the APPC API. EXTRACTED = batch was extracted. FILE_STRUCTURE = batch is stored as non-record oriented. FTP = batch was collected through an FTP transmission. INCOMPLETE = batch is incomplete (not successfully collected). MULTXMIT = batch is available for multiple transmission. NONTRANSMITTABLE = batch cannot be transmitted. REQUESTABLE = batch is available for online requests by remote sites or for transmission by host-initiated Auto Connect sessions. SNA = batch was collected through an SNA transmission. SSL = batch was collected over a secured connection using SSL or TLS. TRANSPARENT = batch contains transparent data. TRANSMITTED = batch was transmitted. UNEXTRACTABLE = batch cannot be extracted. The abbreviated form for this parameter is: STATUS=(A,B,C,D,EB, EX,FI, FTP,I,M,N,R,SN,SS,TRANSP,TRANSM,U)
TIME=hh:mm | (hh:mm,hh:mm…)	Required. Valid only in Scheduler Application Agent rules. Specifies the particular times, in universal (military) hh:mm format, to schedule one or more rules to be processed. Specify 1 to 128 different time values, separated by commas and enclosed in parentheses if more than one value is specified. A calendar can also be specified in conjunction with the TIME= parameter to schedule one or more rules to be processed at a specified time and day/date. For information, refer to the description for the parameter, *CALENDAR* on page 74.

Parameters	Definition
ULTEXT01–ULTEXT96=xxxxxxxx|(xxxxxxxx,xxxxxxxx...)	Optional. Valid only in Log Application Agent rules for ACDETAIL record type. Specifies the 1st – 96th words in the user log text. Specify 1–16 words (up to 255 bytes each) in each ULTEXTnn keyword parameter, separated by commas and enclosed in parentheses if more than one word is specified. The comparison is case-sensitive. Wildcards are supported. An asterisk (*) represents any 0-255 byte string. A percent (%) represents any one character. The rule is selected if, for every ULTEXTnn parameter specified, one of the parameter's values matches the associated word in the message. If any ULTEXTnn parameter is not specified, the associated word in the message is not part of the selection process. It is not necessary to specify contiguous ULTEXTnn parameters; for example, you can specify ULTEXT03 and either specify or omit ULTEXT02. User text words matched to ULTEXT01–ULTEXT96 can be delimited by the following characters in addition to blanks: period, comma, equal sign, open parenthesis and close parenthesis.
WILD_CARD=BID	Optional. Valid for End of Batch, Wake Up Terminate, and Logging rules. Turns on wildcard checking using the specified BATCHID as the criterion for selecting batches. BID = Performs wildcard checking on the User Batch ID You can place wildcard mask characters (multi character and single character) anywhere in the selection pattern. To turn case-sensitivity off, you must use the CASE_SENSITIVE=NO parameter with this parameter; otherwise, the default is to treat the tested string and mask pattern as case-sensitive. For more information, see page 79. **Note:** For more information on using wildcards when selecting BIDs, see *Connect:Enterprise for z/OS Release Notes.*
WILD_CARD_MULTI_CHAR=* | x[xxxxxxx]	Optional. Valid for End of Batch, Wake Up Terminate, and Logging rules. Specifies the pattern of 1-8 special characters used to represent zero or more characters in the field being compared with the specified BATCHID value as the criterion for selecting batches. A contiguously repeating multi-wildcard character has no additional effect, for example, 'A*' is identical to 'A**' , 'A***', and so on. The default value is '*'. **Note:** You must specify the WILD_CARD=BID parameter when using this parameter.

Parameters	Definition
WILD_CARD_SINGLE_ CHAR=% | x[xxxxxxx]	Optional. Valid for End of Batch, Wake Up Terminate, and Logging rules. Specifies a single character used to represent exactly one character in the field being compared with the specified BATCHID value as the criterion for selecting batches. The default value is '%'. **Note:** You must specify the WILD_CARD=BID parameter when using this parameter.

Verifying Application Agent Rule Sets

The Rules Offline Verification utility verifies the syntax of an application agent rule set. Execute the utility as a stand-alone offline batch job. Run the Rules Verification utility whenever you create or modify a rule set.

The following JCL executes the Rules Verification utility. It is located in the Connect:Enterprise sample library shipped on the distribution tape in the member name VERIFYRL

```
//VERIFYRL JOB as required by your site
//*****************************************************
//*    APPLICATION AGENT RULES VERIFICATION UTILITY    *
//*****************************************************
//VERIFYRL EXEC PGM=STMC99,REGION=2000K,TIME=1440
//STEPLIB    DD       DISP=SHR,DSN=ENTPRS.LOAD
//SYSPRINT   DD       SYSOUT=*
//SNAPOUT    DD       SYSOUT=*,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=1632)
//SYSUDUMP   DD       SYSOUT=*
//RULES      DD       DISP=SHR,DSN=ENTPRS.RULES
//SYSIN      DD       *,DLM=ZZ
 RULESCON=XXXXXXXX
 RULESEOB=XXXXXXXX
 RULESLOG=XXXXXXXX
 RULESSCH=XXXXXXXX
 RULESWKT=XXXXXXXX
 TRACE=NO|YES
ZZ
```

Offline Rules Verification Utility Files

The offline Rules Verification utility uses the following input and output files at execution time:

Files	Definition
RULES	The input file containing the application agent rules members. Define the file as an 80-byte fixed blocked PDS (partitioned data set).

Files	Definition
SNAPOUT	Contains the trace data if tracing is specified.
SYSUDUMP	Contains diagnostic information if internal errors occur.
SYSPRINT	Contains a printout of all 80-character SYSIN control records, followed by the input RULES records. Columns 73–80 are ignored and set to blanks when written to the SYSPRINT file. However, on each SELECT statement, columns 73–80 contain a sequence number generated by the Rules Verification utility. If the verification program detects any errors, error messages are also written to this file. All such error messages begin with CMRnnnx, where *nnn* is the specific message number and *x* denotes severity. These messages are documented in *Connect:Enterprise for z/OS Messages and Codes Guide.*
SYSIN	Contains 80-character input control records that specify the rule member names to verify. Comment records are allowed by specifying an asterisk in column one. SYSIN control records are as follows: ◆ RULESCON=xxxxxxxx = Console rules member to verify. ◆ RULESEOB=xxxxxxxx = End of Batch rules member to verify. ◆ RULESLOG=xxxxxxxx = Log rules member to verify. ◆ RULESSCH=xxxxxxxx = Scheduler rules member to verify. ◆ RULESWKT=xxxxxxxx = Wake Up Terminate rules member to verify. ◆ TRACE=NO | YES = Trace the rules control blocks. **Note:** If you specify YES, all of the rules control blocks are written to the SNAPOUT DD file. TRACE is only for debugging and is not required. Specify any of the previous control statements for one execution, but only one of each type. No multiple occurrences of a given control record are allowed.

Chapter 3

Implementing Application Agent Rules

This chapter describes the general procedure for implementing application agents, changing rule sets during Connect:Enterprise for z/OS processing, debugging rule sets using the trace facility, and it provides sample application agent implementations.

Implementing Application Agents

After you define and verify the rule sets for processing, you are ready to implement the application agents. This section outlines the general procedure for implementing application agents and explains Connect:Enterprise JCL and Options Definition File (ODF) requirements. See *Connect:Enterprise JCL and ODF Configuration for Application Agents* on page 92 for examples of the JCL and ODF.

To implement application agents:

1. Allocate the //RULES DD file. The //RULES DD file is a PDS with LRECL=80,RECFM=FB.
2. Create the RULES member or members in the //RULES DD file that you allocated in the previous step. Refer to Chapter 2, *Creating and Verifying Application Agent Rules*, for information about defining a rule set.

   An example of each rule set is in the example library, ENTPRS.EXAMPLE, on the Connect:Enterprise release tape. The example rule member names are as follows:

   - $RCON—Console application agent rules
   - $REOB—End of Batch application agent rules
   - $RLOG—Logging application agent rules
   - $RSCH—Scheduler application agent rules
   - $RWKT—Wake Up Terminate application agent rules
3. Create any required JCL member or members in the //RULESJCL file that you allocated in Step 1 above.
4. If the SUBMIT instruction is in your rules, allocate the //RULESJCL DD file. The //RULESJCL DD file is a PDS with LRECL=80,RECFM=FB.

5. If you need to trace the flow of application agent processing, allocate the //RULTRACE DD file. The //RULTRACE DD file is a sequential file or SYSOUT with LRECL=133, RECFM=FBA.
6. Update the Connect:Enterprise JCL with the following statements, using the table for reference:
   - //JESRDR DD
   - //RULES DD
   - //RULESJCL DD
   - //RULTRACE DD (required only for using the trace feature)

DD Statement	Description
//JESRDR DD	Enables the application agent to submit jobs. This DD statement is for the internal reader. This file is required if the SUBMIT instruction is included in your rules file.
//RULES DD	Identifies the data set where the application agent rules reside. Allocate this file as a partitioned data set with a record format of FB (fixed block) and a record size of 80. This file is required.
//RULESJCL DD	Identifies the data set where the JCL members, used by the SUBMIT instruction, reside. Allocate this file as a partitioned data set with a record format of FB (fixed block) and a record size of 80. This file is required.
//SYSPRINT DD	Required. Contains a listing of the rules members read during Connect:Enterprise startup or rules refresh, along with any error messages resulting from incorrectly specified RULE statements. Columns 73–80 are ignored and set to blanks when written to the SYSPRINT file. However, on each SELECT statement, columns 73–80 contain a sequence number. This number uniquely identifies each SELECT statement and is useful when tracing application agent requests. Allocate this file as a sequential data set with a record format of FB (fixed block) and a record size of 133. You can also specify SYSOUT.
//RULTRACE DD	Identifies the data set to write the application agent trace records to. Allocate this file as a sequential data set with a record format of FBA (fixed block) and a record size of 133. This file is only required when application agent tracing is turned on.

7. Activate application agent processing and individual application agents by updating the Options Definition File using this table for reference:
   - RULES=YES (required to activate application agent processing)
   - MAXRP=nn
   - RULESCON=*xxxxxxxx* (member name from //RULES file)
   - RULESEOB=*xxxxxxxx* (member name from //RULES file)

- RULESLOG=*xxxxxxxx* (member name from //RULES file)
- RULESSCH=*xxxxxxxx* (member name from //RULES file)
- RULESWKT=*xxxxxxxx* (member name from //RULES file)
- RULES_IR=YES|NO
- RULES_RECURSION_MAX=nnnnnnnnnn

Application Agent Parameters	Definition
RULES=YES	Activates the application agent processing environment within the data repository. **Note:** You must specify RULES=YES to activate application agent processing. You can activate individual application agents by specifying the member name for each application agent.
MAXRP=nn	Specifies the maximum number of Rules Processor subtasks allowed to concurrently process requests. Specify a number between 1 and 99; however, the number permitted is limited by the storage available to your system. The default value is 2. **Note:** Based on the MAXRP value and the amount of other activity defined in Connect:Enterprise, you may need to increase your region value. See *Connect:Enterprise for z/OS Administration Guide* for more information about the MAXRP parameter and all other parameters.
RULESCON=xxxxxxxx	Specifies the Console application agent rules PDS member. This member must reside in the //RULES DD file. Specify this parameter to activate the Console application agent.
RULESEOB=xxxxxxxx	Specifies the End of Batch application agent rules PDS member. This member must reside in the //RULES DD file. Specify this parameter to activate the End of Batch application agent.
RULESLOG=xxxxxxxx	Specifies the Logging application agent rules PDS member. This member must reside in the //RULES DD file. Specify this parameter to activate the Logging application agent.
RULESSCH=xxxxxxxx	Specifies the Scheduler application agent rules PDS member. This member must reside in the //RULES DD file. Specify this parameter to activate the Scheduler application agent.
RULESWKT=xxxxxxxx	Specifies the Wake Up Terminate application agent rules PDS member. This member must reside in the //RULES DD file. Specify this parameter to activate the Wake Up Terminate application agent.
RULES_IR=YES|NO	Specifies whether each RP task subtask allocates its own JES2 internal reader, and does not use //JESRDR when submitting JCL from //RULESJCL.

Application Agent Parameters	Definition
RULES_RECURSION_MAX=nnnnnnnnnn | 5	Defines number of times console application agent-to-console application agent recursions (C2C recursions) are allowed to loop. 5 = Default; allows some recursion while preventing a loop from overwhelming the job or system log with messages. 0 = No C2C recursion is allowed. nnnnnnnnnn = 2147483647. For practical purposes, sets no limit. **Note:** See *Detecting Application Agent Loops* on page 97 for more information on this parameter, which is used when troubleshooting and testing application agents.

8. Start Connect:Enterprise.

Connect:Enterprise JCL and ODF Configuration for Application Agents

The following example illustrates the Connect:Enterprise JCL and Options Definition File configured for application agent processing. Modifications required for application agents processing are highlighted in bold text.

```
//jobcard JOB ...... as required by your site //*****************************************************************
 //*                          Connect:Enterprise for z/OS                        *
 //*************************************************************
 //MAILBOX  EXEC PGM=STMAIN,REGION=6000K,TIME=1440,PARM=xxxx
 //STEPLIB  DD DISP=SHR,DSN=ENTPRS.LOAD
 //SYSPRINT DD SYSOUT=*
 //SNAPOUT  DD SYSOUT=*,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=1632)
 //BTSNAP   DD SYSOUT=*,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=1632)   //SYSUDUMP DD SYSOUT=*
 //JESRDR   DD SYSOUT=(A,INTRDR)
 //RULES    DD DISP=SHR,DSN=ENTPRS.RULES
 //RULESJCL DD DISP=SHR,DSN=ENTPRS.RULES.CNTL
 //RULTRACE DD DISP=SHR,DSN=ENTPRS.RULES.TRACE
 //SYSABEND DD SYSOUT=*,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=1632)
 //OPTDEF   DD *,DLM=ZZ
 *OPTIONS
   VPF='ENTPRS.VPF'
   VBQROTAT=10
   VBQPCT=90
   VSESSLIM=0
   VTAM=YES
   ACQDEFAULT=YES
   PASSWORD=BANANA
   APPLID=LU050
 **----- APPLICATION AGENT RULES -----
  RULES=YES
  RULES_IR=YES
  RULES_RECURSION_MAX=nnnnnnnnnn
  MAXRP=nn
  RULESCON=XXXXXXXX
  RULESEOB=XXXXXXXX
  RULESLOG=XXXXXXXX
  RULESSCH=XXXXXXXX
  RULESWKT=XXXXXXXX
 **---- CICS ------------------------
   MAXCP=2
   APPC=YES
   APPCAPPL=LU051
   CICSMODE=TESTLU62
   CICSAPPL=CICS0001
   CICSTR1=CM62
 **-----------------------------------
   CONSLOG=YES
   LOGONMSG='SUCCESSFUL LOGON TO Connect:Enterprise SYSTEM'
   CMB001I='Connect:Enterprise PRODUCTION SYSTEM'
 *CONNECT
   LISTNAME=LIST1
     TYPE=LU1RJE
     DISCINTV=15
     ACSESS#=1
     RETRY=1
     TIME=19:35
     RMTA001 IDLIST=TEST2 TEST111 TEST1111
 *REMOTES
   NAME=RMTA001
     TYPE=LU1RJE
     CONSOLE=NO
     RMTACB=LU002
     USERDATA='RMTB001'
     COMPRESS=NO
     BLKSIZE=256
```

Refreshing Application Agent Rules

The $$REFRESH Console command enables you to change the application agent rules while Connect:Enterprise is processing. You can apply the $$REFRESH command to all agents or to individual application agents. Enter the $$REFRESH command as a system console command as follows:

```
$$REFRESH RULES=ALL
                CON
                EOB
                LOG
                SCH
                WKT
```

You can also enter the $$REFRESH command through the CICS or ISPF interface.

When you issue a $$REFRESH command, active transactions in progress complete using the old rules. If the refresh rules are parsed successfully, the new rules are rolled in and new transactions begin using the new rules. The SYSPRINT file for the Connect:Enterprise job is updated with a listing of the new rules, including a time and date stamp for the refresh.

For $$REFRESH RULES=ALL, if the refresh parsing detects any errors, none of the new rules are rolled in. Refer to the SYSPRINT file for rules error messages.

Troubleshooting Application Agent Requests

To test and debug rule sets, use the trace facility that records the flow of application agent processing. If you suspect a logic problem in the execution of a rule set, see *Detecting Application Agent Loops* on page 97.

Tracing Application Agent Requests

Tracing allows you to see which rule is processed for each application agent request. The trace facility writes the trace messages to the //RULTRACE DD file specified in your Connect:Enterprise JCL. The presence of the //RULTRACE DD statement indicates that the trace facility can trace application agent processing flow. Tracing for specific agents is activated by options set in the ODF *OPTIONS section, by operator console commands, or by requests from the CICS/ISPF interfaces.

To activate application agent flow traces, use one of the following:

- ODF (Options Definition File)
  - *OPTIONS
    - TRACE=RPCON activates Console application agent traces.
    - TRACE=RPEOB activates End of Batch application agent traces.
    - TRACE=RPLOG activates Logging application agent traces.
    - TRACE=RPSCH activates Scheduler application agent traces.
    - TRACE=RPWKT activates Wake Up Terminate application agent traces.
- Operator Console Commands
  - To Activate
    - $$TRACE RPCON
    - $$TRACE RPEON

- $$TRACE RPLON
- $$TRACE RPSON
- $$TRACE RPWON

◆ To Deactivate

- $$TRACE RPCOFF
- $$TRACE RPEOFF
- $$TRACE RPLOFF
- $$TRACE RPSOFF
- $$TRACE RPWOFF

✦ ISPF/CICS Trace Management Screen

◆ Select one of the following:

- RPCON
- RPEOB
- RPLOG
- RPSCH
- RPWKT

When Connect:Enterprise sends an application agent request to the Rules Processor, the Rules Processor checks the SELECT statements for a match against the request data and processes the rules pointed to by the first matching SELECT statement.

Each trace entry contains information about the application agent request, the chosen SELECT statement and rule, and the associated instructions executed within that rule. The following two examples illustrate the trace entry layout of both a matched and nonmatched request.

Trace messages begin with the message ID followed by a 3-digit number and date and time stamp. The 3-digit number represents the specific rules processor task that processed the request. Usually the trace messages are grouped together by processor task number. Heavy system activity that drives more than one rules processor task interleaves the traces. In this case, the 3-digit number enables you to find related traces.

Example 1—End of Batch Application Agent Request Trace Entry (Match)

The following is an example of an End of Batch application agent request trace entry that found a match during the selection processing:

```
CMR701I - 001 yyddd-17:48:05 EOB Rules processing started, using "RULE001 " rule dated
yyddd-17:46:21
CMR702I - 001 yyddd-17:48:05 SELECT statement #00000001 matched. EOB Application
Agent            request data:
CMR730I - 001 yyddd-17:48:05 ID=TESTID01  B#=0000008 BATCHID='TESTID01 BATCHID '
REMOTE=none      LINEID=none
CMR731I - 001 yyddd-17:48:05    STATUS=A,M,R
CMR708I - 001 yyddd-17:48:05 EXECUTE program SV$BR14
CMR706I - 001 yyddd-17:48:07 COMMAND sent to console: DNET,ID=SBLDUB97
CMR707I - 001 yyddd-17:48:07 MESSAGE sent to console: THIS IS AN APPLICATION AGENT
TEST MSG
CMR711I - 001 yyddd-17:48:07 NOP
CMR709I - 001 yyddd-17:48:07 SUBMIT member LISTA
CMR715I  - 001 yyddd-17:48:07 STATFLG ONFLAGS=RM OFFFLAGS=none STATUS before: A
after: A   R  M
CMR710I - 001 yyddd-17:48:07 WAKEUP  CICSSYSID=C212  CICSTRANID=CMS1
CMR720I - 001 yyddd-17:48:07 EOB Rules processing complete
```

The following is detailed information concerning the preceding example:

Trace Message ID #	Description
CMR701I	Lists the rule selected for processing and includes a date and time stamp. This trace message is important if you REFRESH the rules while Connect:Enterprise is online, because it indicates which rules were used by the date and time stamp on the rules SYSPRINT header.
CMR702I	Lists the sequence number of the SELECT statement that matched the request. This sequence number is the same number listed in columns 73–80 on the SYSPRINT listing.
CMR730I and CMR731I	Displays information from the application agent request
CMR708I, CMR706I, CMR707I, CMR711I, CMR709I, CMR715I, and CMR710I	Lists the instructions in the order they execute
CMR720I	Indicates processing is complete for this End of Batch application agent request

Example 2—End of Batch Application Agent Request Trace Entry (No Match)

The following is an example of End of Batch application agent request trace entry that did not find a match, during the selection processing:

```
CMR703I-001 yyddd-17:48:34 EOB Rules processing started, no selection criteria matched
CMR730I-001 yyddd-17:48:34 ID=TESTID01 B#=0000010 BATCHID='ENCRYPTED' REMOTE=none LINEID=none
CMR731I-001 yyddd-17:48:34    STATUS=C,EB,R,TRANSP
CMR720I-001 yyddd-17:48:34 EOB Rules processing complete
```

The following is detailed information concerning the preceding example:

Trace Message ID #	Description
CMR703I	Indicates that no SELECT statement was chosen for this request
CMR730I and CMR731I	Displays information from the application agent request
CMR720I	Indicates processing is complete for this End of Batch application agent request

Detecting Application Agent Loops

When enabling application agents, care should be taken to avoid creating a loop. A loop results when one agent triggers another (a "recursion"), and that agent triggers another, and so on without end. One or more recursions are a necessary, but insufficient condition for a loop—a loop also requires that the last recursion in a chain (which can be any length) trigger a previous part of the chain. Application Agent rules can cause the following types of recursions:

- An agent could issue a MESSAGE instruction with CONEVENT=YES, which would directly trigger a Console Agent.
- An agent could issue a COMMAND instruction, which causes a new agent to be triggered. Such commands include:
  - $$INVOKE, which directly triggers an EOB or Console agent.
  - $$CONNECT, which can cause a batch to be received which triggers an EOB agent.
  - Any $$ command which causes the Connect:Enterprise main task to issue a message which triggers a Console agent.
- An agent could issue a SUBMIT instruction whose batch job causes a new agent to be triggered.
  - The batch job could issue the same commands described above in the COMMAND instruction, and subsequently trigger an agent.
  - The batch job could execute a STOUTL or Cross System Client utility, which could cause the Connect:Enterprise main task to issue a message which triggers a Console Application Agent.
- An agent could issue an EXECUTE instruction, which causes a user-written program to be executed, which in turn causes an agent to be triggered.

- The program could issue an operator command, which would have the same effect as using the Application Agent COMMAND instruction.
- The program could submit a batch job, which would have the same effect as using the Application Agent SUBMIT instruction.

✦ An agent could issue a WAKEUP instruction, which could cause CICS to run a transaction which triggers an agent.

- The CICS transaction could cause a batch to be received, which triggers an End of Batch application agent.
- The CICS transaction could issue the same commands described above in the COMMAND instruction, and so trigger an agent.
- The CICS transaction could submit a batch job which triggers an agent as described in the SUBMIT instruction above.

Once a loop is started, it can be stopped manually in one of two ways:

✦ Stop any application agent that plays a part in the loop. For example, to stop the Console agent, use this console command, $$STOP RULES=CON.

✦ Shut down the Connect:Enterprise main task—it is recommended that you use the immediate option, that is, $$SHUTDOWN,I.

After stopping the loop, you can change the Application Agent rules to eliminate the loop. Then you can either:

✦ Refresh the rules ($$REFRESH RULES=ALL) and restart the stopped Application Agent. For example, start the Console agent, use this console command, $$START RULES=CON.

✦ Restart the Connect:Enterprise main task

A loop can be stopped automatically as long as it is completely consists of console application agent-to-console application agent recursions (referred to as “C2C recursion”) . Once any part of the loop occurs outside of console application agent processing, it is impossible to determine whether it is a recursion or not, and so such loops must be stopped manually as described above. The maximum depth of C2C recursion that is allowed is specified by the ODF parameter RULES_RECURSION_MAX. (See the chapter on the ODF in the *Connect:Enterprise for z/OS Administration Guide* for more information on this parameter.) If this parameter is set to 0, no C2C recursion is allowed. It can be set to any number up to 2,147,483,647, which for all practical purposes is no limit at all. The default of 5 allows some room for deliberate C2C recursion while preventing a loop from swamping the job or system log with messages.

In this example, the C2C loop is limited to MSGID CMB087I:

```
RULE NAME=RULE1
     MESSAGE TEXT='CMB087I RULE1',
             ROUTCODE=(11),
             DESCCODE=(7),
             CONEVENT=YES
SELECT RULE=(RULE1),
      MSG01=(CMB087I)
```

In this example, the C2C loop can happen for either CMB087I or CMB088I. Note the use of the Console application agent variable '&MSG01' in the first word in the TEXT parameter. The same loop would result if it were '@MSG01', '&MSG', or '@MSG'.

```
RULE NAME=RULE2
     MESSAGE TEXT='&MSG01 RULE2',
             ROUTCODE=(11),
             DESCCODE=(7),
             CONEVENT=YES
SELECT RULE=(RULE2),
      MSG01=(CMB087I,CMB088I)
```

Sample End of Batch and Wake Up Terminate Rules Implementations

The following example illustrates how to use the End of Batch and Wake Up Terminate application agents to process collected batches.

Site Requirements Example

Whenever Connect:Enterprise collects a batch with a mailbox ID beginning with either MBOX1 or MBOX2, it performs the following actions:

- Issues an Auto Connect for LISTNAME=DEST1 to transmit the newly collected batch.
- Extracts the batch to a PDS. The member name of the extracted batch is Bnnnnnnn, where nnnnnnn is the 7-digit batch number.
- Issues a Wake Up transaction to notify CICS of the collected batch. CICS then STARTs transaction TRN1 on system SYS1.
- If CICS does not successfully START transaction TRN1, Connect:Enterprise issues a message to the operator as follows:

```
"CICS ALERT - WAKEUP TRN1 FAILED FOR ID=xxxxxxxx BID=xxxx....xxxx BATCHNO=nnnnnnn"
```

  - ID—8-character Mailbox ID
  - BID—64-character user batch ID
  - BATCHNO—7-digit batch number

The following End of Batch rules, EXTRACT job, and Wake Up Terminate rules accomplish the processing described in the preceding site requirements.

End of Batch Application Agent Rules Example

The following example illustrates an End of Batch application rule that processes batches collected for the site requirements:

```
RULE NAME=RULE1
   COMMAND TEXT='F &STCNAME,$$CONNECT LISTNAME=DEST1 ID=&IDFIELD BATCHID=BATCH#'
   SUBMIT  MEMBER=EXTR001
   WAKEUP CICSDEFN=TRANSACTION,
          CICSSYSID=SYS1,
         CICSTRANID=TRN1
 SELECT RULE=RULE1,ID=MBOX1*
 SELECT RULE=RULE1,ID=MBOX2*
```

RULESJCL DD Member Example

The following example illustrates EXTR001, which is the RULESJCL DD member used in the SUBMIT instruction in the preceding example:

```
//jobname  JOB  as required by your site
//EXTRACT  EXEC PGM=STOUTL,PARM='xxxx',REGION=2M
//STEPLIB  DD   DISP=SHR,DSN=ENTPRS.LOAD
//JESRDR   DD   SYSOUT=(A,INTRDR)
//SYSTERM  DD   SYSOUT=*,DCB=(RECFM=FA,LRECL=133,BUFNO=0)
//BTSNAP   DD   SYSOUT=*,DCB=(RECFM=FA,LRECL=133,BUFNO=0)
//PRINT    DD   SYSOUT=*,DCB=(RECFM=FA,LRECL=133,BUFNO=0)
//SYSPRINT DD   SYSOUT=*,DCB=(RECFM=FA,LRECL=133,BUFNO=0)
//REPORTS  DD   SYSOUT=*,DCB=(RECFM=FA,LRECL=133,BUFNO=0)
//OUTFILE  DD   DISP=OLD,DSN=ENTPRS.EXTRACT(B&BATCH#)
//SYSIN    DD   DATA,DLM=ZZ
   EXTRACT
   VPF='ENTPRS.VPF
   BATCHNUM=&BATCH#
ZZ
```

Wake Up Terminate Application Agent Rules Example

The following is an example of Wake Up Terminate application agent rules:

```
RULE NAME=RULE1
    MESSAGE TEXT='CICS ALERT - WAKEUP TRN1 FAILED FOR ID=&IDFIELD BID-
                 =&BID64 BATCHNO=&BATCH#'
 SELECT RULE=RULE1,ID=MBOX1*,RTNCODE=0001-FFFF
 SELECT RULE=RULE1,ID=MBOX2*,RTNCODE=0001-FFFF
```

Sample Log Rules Implementations

These examples demonstrate how the Connect:Enterprise Logging application agent can be used to process specific logging events—the first involving a message being sent to the operator console whenever a successful batch collection is logged during connect sessions and the second involving user log text to be included in reports and user interface screens upon a failed logon attempt.

Site Requirements Example for Successful Batch Collection

Whenever Connect:Enterprise logs a successful batch collection during an Auto Connect, write the following message to the operator console:

```
BATCH COLL VIA AUTO CONNECT:  ID=xxxxxxxx
Batchid=xxxxxxxxxxxxxxxxxxxxxxxx
Batch#=nnnnnnn Listname=xxxxxxxx
```

Whenever Connect:Enterprise logs a successful batch collection during a Remote Connect, write the following message to the operator console:

```
BATCH COLL VIA REMOTE CONNECT:  ID=xxxxxxxx
Batchid=xxxxxxxxxxxxxxxxxxxxxxxx
Batch#=nnnnnnn Remote=xxxxxxxx
```

- ID—8-character Mailbox ID
- Batchid—24-character user batch ID
- Batch#—7-digit batch number
- Listname—8-character Auto Connect listname
- Remote—8-character remote name

Logging Application Agent Rules Example

The following example illustrates the Logging application agent rules to write to log a successful batch collection:

```
RULE   NAME=RULE001
       MESSAGE TEXT='BATCH COLL VIA AUTO CONNECT:  ID=@IDFIELD Batch-
                    id=@* BID24 Batch#=@BATCH# Listname=@LISTNAM'
RULE   NAME=RULE002
      MESSAGE TEXT='BATCH COLL VIA REMOTE CONNECT:ID=@IDFIELD Batch-
      id=@* BID24 Batch#=@BATCH# Remote=@RMTNAME'
SELECT RULE=RULE001,
       RECTYPE=ACDETAIL,      AUTO CONNECT DETAIL
       ACFUNC=RECV,           COLL WITH $$ADD, WITHOUT $$ADD
       LOGFUNC=(NEW,UPDATE),
       FAILCODE=000
SELECT RULE=RULE002,
       RECTYPE=RCDETAIL,      REMOTE CONNECT DETAIL
       RCFUNC=(ADD,NOADD),COLL WITH $$ADD, WITHOUT $$ADD
       LOGFUNC=(NEW,UPDATE),
       FAILCODE=000
```

Site Requirements Example for Failed Logon Attempt

Whenever Connect:Enterprise logs an unsuccessful logon attempt during an FTP Auto Connect, write the following fail code and user log text to the ACDETAIL report and the related ISPF and CICS user interface screens:

```
Script FTPLOGON failed on PASS command, FC=240 for USER FTPRMTB.
```

Fail code 240 through 255 are user-defined CONNECT fail codes in the STUAMMT table. Text from these user log fail codes is listed along with the text predefined for fail code 240. (See *Connect:Enterprise for z/OS User's Guide* for more information about customizing text for user-defined fail codes.)

```
RULE NAME=RULE240
     MESSAGE TEXT='RULE240 ACFUNC=ULOG ULFC=240 SELECTED'
RULE NAME=RULE25X
     MESSAGE TEXT='RULE25X ACFUNC=ULOG ULFC=250-255 SELECTED'
SELECT RULE=RULE240,RECTYPE=ACDETAIL,
       ACFUNC=ULOG,LISTNAME=LFTP1,REMOTE=FTPRMT1,FAILCODE=240,
       ULTEXT02=('emergency','failure','error'),
       ULTEXT08='call',
       ULTEXT10='Helpdesk',
SELECT RULE=RULE25X,RECTYPE=ACDETAIL,
       ACFUNC=ULOG,LISTNAME=LFTP1,REMOTE=FTPRMT1,FAILCODE=250-255,
       ULTEXT01=(U250,U251,U252,U253,U254,U255)
```

Sample Console Rules Implementation

This example illustrates a Console application agent that handles certain C:E FTP message traffic.

Site Requirements Example

Whenever Connect:Enterprise issues a CMB2103I message (or DUMMY001 or DUMMY003) whose fifth word is "SERVER", it performs the following actions:

- ✦ Executes a program
- ✦ Issues an operator command
- ✦ Reformats the CMB2103I message and issues it to the operator console
- ✦ Captures the number of FTP threads in the message and sends an SNMP trap to a Network Manager.
- ✦ Submits a batch job

Console Application Agent Rules Example

The following example illustrates the application agent rule set that implements the site requirements.

```
*********************************************************************
*  CONSOLE APPLICATION AGENT RULES EXAMPLE                          *
*********************************************************************
RULE NAME=RULE001
 EXECUTE  PROGRAM=IEFBR14,
          ERROR=CONTINUE
 COMMAND  TEXT='D NET,ID=RDXSA064',
          ERROR=CONTINUE
 MESSAGE  TEXT='CON RULE001 &MSG04 &MSG05 &MSG11 &MSG12 &MSG02 &MSG09',
          ERROR=CONTINUE
 NOP
 SNMPTRAP TEXT='CON RULE001 &MSG',
          IPADDR=MYCOMPANY.COM,
          PORT=162,
          GROUP1=STATUS,
          GROUP2=2222,
          ERROR=CONTINUE
 SUBMIT   MEMBER=IEFBR14
SELECT  RULE=(RULE001),MSG01=(DUMMY001,CMB2103I,DUMMY003),MSG05=(SERVER)
```

Console Application Agent Request Operator Messages Example

Following is an example of system log output when Connect:Enterprise is started with the above rule. The first CMB2103I matches the rule, the second does not. The rest of the messages are generated by the matching rule.

```
16.19.29 STC00469  CMB2103I - C:E FTP SERVER THREAD INITIALIZATION COMPLETE. 0002
SESSION THREADS ALLOCATED.
16.19.29 STC00469  CMB2103I - C:E FTP CLIENT THREAD INITIALIZATION COMPLETE. 0002
SESSION THREADS ALLOCATED.
16.19.29 STC00469  IEA630I  OPERATOR MBXNAMEA NOW ACTIVE,   SYSTEM=CSGA    ,
LU=MBXNAMEA
16.19.29 STC00469  D NET,ID=RDXSA064
16.19.29 STC00469  IEA631I  OPERATOR MBXNAMEA NOW INACTIVE, SYSTEM=CSGA    ,
LU=MBXNAMEA
16.19.29 STC00469  CON RULE001 FTP SERVER THREADS ALLOCATED - 0002
```

Console Application Agent Request Trace Entry (Match and No Match) Example

The following is an example of application agent RULTRACE DD output when C:E is started with the above rule:

```
CMR750I - 001 2004034-16:19:29 CON Selection processing started
CMR751I - 001 2004034-16:19:29   Match on SELECT stmt 00000001 MSG01 CMB2103I
and list word 02 CMB2103I
CMR751I - 001 2004034-16:19:29   Match on SELECT stmt 00000001 MSG05 SERVER
and list word 01 SERVER
CMR753I - 001 2004034-16:19:29 CON Selection processing ended

CMR702I - 001 2004034-16:19:29 SELECT statement #00000001 matched.  CON Application
Agent request data:
CMR754I - 001 2004034-16:19:29    MSG=CMB2103I - C:E FTP SERVER THREAD INITIALIZATION
COMPLETE. 0002 SESSION THREADS ALLOC
CMR701I - 001 2004034-16:19:29 CON Rules processing started, using "RULE001 " rule
dated 2004034-16:19:23
CMR708I - 001 2004034-16:19:29 EXECUTE program IEFBR14
CMR706I - 001 2004034-16:19:29 COMMAND sent to console: D NET,ID=RDXSA064
CMR707I - 001 2004034-16:19:29 MESSAGE sent to console: CON RULE001 &MSG04 &MSG05
&MSG11 &MSG12 &MSG02 &MSG09
CMR711I - 001 2004034-16:19:29 NOP
CMR712I - 001 2004034-16:19:29 SNMPTRAP Description:
CMR713I - 001 2004034-16:19:29    TEXT (U) : CON RULE001 &MSG
CMR713I - 001 2004034-16:19:29    TEXT (R) : CON RULE001 CMB2103I - C:E FTP SERVER
THREAD INITIALIZATION COMPLETE.
                                                                          Continued


CMR713I - 001 2004034-16:19:29      "  "   : 0002 SESSION THREADS ALLOCATED.
CMR713I - 001 2004034-16:19:29    IPADDR   : MYCOMPANY.COM (10.20.4.1)
CMR713I - 001 2004034-16:19:29    PORT     : 162
CMR713I - 001 2004034-16:19:29    GROUP1   : 2
CMR713I - 001 2004034-16:19:29    GROUP2   : 2222
CMR713I - 001 2004034-16:19:29    TRAP OID : 1.3.6.1.4.1.1733.4.2.1.5.2222
CMR713I - 001 2004034-16:19:29    TRAPDATA :  b          %  xb j          b d
CMR713I - 001 2004034-16:19:29     "  "   :
|+   <      (
CMR713I - 001 2004034-16:19:29     "  "   :              +   < !   |+  |(&<
|+          <<|
CMR713I - 001 2004034-16:19:29    "  "  :                 &
|+
CMR713I - 001 2004034-16:19:29    "  "  :  |+                      <
(
CMR709I - 001 2004034-16:19:29 SUBMIT member IEFBR14
CMR720I - 001 2004034-16:19:29 CON Rules processing complete
CMR750I - 001 2004034-16:19:29 CON Selection processing started
CMR751I - 001 2004034-16:19:29   Match on SELECT stmt 00000001 MSG01 CMB2103I
and list word 02 CMB2103I
CMR752I - 001 2004034-16:19:29 No Match on SELECT stmt 00000001 MSG05 CLIENT
CMR753I - 001 2004034-16:19:29 CON Selection processing ended

CMR703I - 001 2004034-16:19:29 CON Rules processing started, no selection criteria
matched
CMR754I - 001 2004034-16:19:29    MSG=CMB2103I - C:E FTP CLIENT THREAD INITIALIZATION
COMPLETE. 0002 SESSION THREADS ALLOC
CMR720I - 001 2004034-16:19:29 CON Rules processing complete
```

The following is detailed information concerning the previous example:

Trace Message ID #	Description
CMR750I	Indicates the beginning of Selection processing for the first CMB2103I message.
CMR751I (#1)	Indicates a successful match of the first criterion in the SELECT statement. The number of the SELECT statement being checked is listed as well as the matching value in the message and the match criterion. In this case, the message's MSG01 is CMB2103I, and it matches the second word in the list specified by the SELECT statement's MSG01 parameter. The MSG01 parameter is also shown since it could contain wildcard characters.
CMR751I (#2)	Indicates a successful match of the second criterion in the SELECT statement.
CMR753I	Indicates the end of Selection processing for the request. At this point, there is no indication whether all criteria matched or not.
CMR702I	Indicates that the request matched all the criteria for a SELECT statement, and gives the number of that SELECT statement. This sequence number is the same number listed in columns 73-80 on the SYSPRINT listing.
CMR754I	Lists the message of the request that matched the SELECT statement.
CMR701I	Lists the rule selected for processing and gives the rule's date and time stamp. This trace message is important if you REFRESH the rules while Connect:Enterprise is online, because it indicates which rules were used by the date and time stamp on the rules SYSPRINT header.
CMR708I, CMR706I, CMR707I, CMR711I, CMR712I, and CMR709I	List the instructions in the order they execute.
CMR712I	Indicates that an SNMP trap is being built and its description will follow.
CMR713I	Describes the SNMP trap in detail. TEXT (U) = Lists the unresolved message text as it appears in the rule. It may be continued onto multiple lines. TEXT (R) = Lists the resolved message text, that is, the text after variable substitution. It may be continued onto multiple lines. It will be converted to ASCII, then encoded using ASN.1 as the value of the ceSNMPTRAPtext varbind pair. IPADDR = Lists the IP address as it appears in the rule, followed by the Hostname or Reverse Hostname lookup value in parentheses. PORT = Lists the port number as it appears in the rule. Combined with the IP address, this indicates where the trap will be sent. GROUP1 = Lists the GROUP1 number. If 1, the rule specified GROUP1=ALARM. If 2, the rule specified GROUP1=STATUS. GROUP2 = Lists the GROUP2 number as it appears in the rule. GROUP1 and GROUP2 give partial user control over the TRAP OID. TRAP OID = Lists the OID generated for the entire trap. It becomes the value of the snmpTrapOID varbind pair.

Trace Message ID #	Description
TRAPDATA	Lists the encoded string which comprises the entire SNMP trap. This is the data in the UDP packet sent to IPADDR/PORT.
CMR720I	Indicates processing is complete for the first CMB2103I message.
CMR750I	Indicates the beginning of Selection processing for the second CMB2103I message.
CMR751I	Indicates a successful match of the first criterion in the SELECT statement.
CMR752I	Indicates an unsuccessful match of the second criterion in the SELECT statement.
CMR753I	Indicates the end of Selection processing for the request. At this point, there is no indication whether all criteria matched or not.
CMR703I	indicates that the request did not matched all the criteria for any single SELECT statement.
CMR754I	Lists the message of the request that matched no SELECT statement(s).
CMR720I	Indicates processing is complete for the second CMB2103I message.

Sample Scheduler Rules Implementation

The following example illustrates a Scheduler application agent that handles certain time events.

Site Requirements Example

Whenever certain times of day occur on weekdays, Connect:Enterprise performs the following actions:

- Executes a program
- Issues an operator command
- Issues a message to the operator console
- Sends an SNMP trap to a Network Manager.
- Submits a batch job

Scheduler Application Agent Rules Example

The following example illustrates the Scheduler application agent rule set that implements the site requirements.

```
*********************************************************************
*  SCHEDULER APPLICATION AGENT RULES EXAMPLE                        *
*********************************************************************
RULE NAME=RULE001
 EXECUTE  PROGRAM=IEFBR14,
          ERROR=CONTINUE
 COMMAND  TEXT='D NET,ID=RULE001',
          ERROR=CONTINUE
 MESSAGE  TEXT='SCH RULE001 MSG',
          ERROR=CONTINUE
 NOP
 SNMPTRAP TEXT='SCH RULE001 TRAP',
          IPADDR=EPETERS2K.CSG.STERCOMM.COM,
          PORT=162,
          GROUP1=STATUS,
          GROUP2=2222,
          ERROR=CONTINUE
 SUBMIT   MEMBER=IEFBR14
RULE NAME=RULE002
 MESSAGE  TEXT='SCH RULE002 MSG'
RULE NAME=RULE003
 MESSAGE  TEXT='SCH RULE003 MSG'
RULE NAME=RULE004
 MESSAGE  TEXT='SCH RULE004 MSG'
SELECT  RULE=RULE001,TIME=(14:05,14:06,14:07),CALENDAR=WEEKDAY
SELECT  RULE=RULE002,TIME=(14:05,14:06,14:07),CALENDAR=WEEKEND
SELECT  RULE=(RULE003,RULE004),TIME=(14:06,14:08)
```

Scheduler Application Agent – Calendar Example

The following example illustrates the ODF calendar definitions that implements the site requirements.

```
*CALENDAR
**
NAME=WEEKEND
    EXDAYS=MON TUE WED THU FRI
NAME=WEEKDAY
    EXDAYS=SAT SUN
```

When Connect:Enterprise is started with the above rule and calendar on a Wednesday:

- At 14:05, Select statement 1 is processed.
- At 14:06, Select statements 1 and 3 are processed.
- At 14:07, Select statement 1 is processed.
- At 14:08, Select statement 3 is processed.

Select statement 2 is not processed because Wednesday is excluded in the WEEKEND calendar.

Scheduler Application Agent Request Operator Messages Example

The following is an example of system log output from the above processing:

```
14.05.00 STC04005  IEA630I  OPERATOR MBXNAMEA NOW ACTIVE,   SYSTEM=CSGA    ,
LU=MBXNAMEA
14.05.00 STC04005  D NET,ID=RULE001
14.05.00 STC04005  IEA631I  OPERATOR MBXNAMEA NOW INACTIVE, SYSTEM=CSGA    ,
LU=MBXNAMEA
14.05.00 STC04005  SCH RULE001 MSG
14.06.00 STC04005  SCH RULE003 MSG
14.06.00 STC04005  SCH RULE004 MSG
14.06.00 STC04005  IEA630I  OPERATOR MBXNAMEA NOW ACTIVE,   SYSTEM=CSGA    ,
LU=MBXNAMEA
14.06.00 STC04005  D NET,ID=RULE001
14.06.00 STC04005  IEA631I  OPERATOR MBXNAMEA NOW INACTIVE, SYSTEM=CSGA    ,
LU=MBXNAMEA
14.06.00 STC04005  SCH RULE001 MSG
14.07.00 STC04005  IEA630I  OPERATOR MBXNAMEA NOW ACTIVE,   SYSTEM=CSGA    ,
LU=MBXNAMEA
14.07.00 STC04005  D NET,ID=RULE001
14.07.00 STC04005  IEA631I  OPERATOR MBXNAMEA NOW INACTIVE, SYSTEM=CSGA    ,
LU=MBXNAMEA
14.07.00 STC04005  SCH RULE001 MSG
14.08.00 STC04005  SCH RULE003 MSG
14.08.00 STC04005  SCH RULE004 MSG
```

Scheduler Application Agent Request Trace Entry Example

The following is an example of application agent RULTRACE DD output when Connect:Enterprise is started with the above rule:

```
CMR702I - 001 2004035-14:05:00 SELECT statement #00000001 matched.  SCH Application
Agent request data:

CMR701I - 001 2004035-14:05:00 SCH Rules processing started, using "RULE001 " rule
dated 2004035-14:04:40
CMR708I - 001 2004035-14:05:00 EXECUTE program IEFBR14
CMR706I - 001 2004035-14:05:00 COMMAND sent to console: D NET,ID=RULE001
CMR707I - 001 2004035-14:05:00 MESSAGE sent to console: SCH RULE001 MSG
CMR711I - 001 2004035-14:05:00 NOP
CMR712I - 001 2004035-14:05:00 SNMPTRAP Description:
CMR713I - 001 2004035-14:05:00    TEXT (U) : SCH RULE001 TRAP
CMR713I - 001 2004035-14:05:00    TEXT (R) : SCH RULE001 TRAP
CMR713I - 001 2004035-14:05:00    IPADDR   : EP2K.CSG.STERCOMM.COM (10.20.4.1)
CMR713I - 001 2004035-14:05:00    PORT     : 162
CMR713I - 001 2004035-14:05:00    GROUP1   : 2
CMR713I - 001 2004035-14:05:00    GROUP2   : 2222
CMR713I - 001 2004035-14:05:00    TRAP OID : 1.3.6.1.4.1.1733.4.2.1.4.2222
CMR713I - 001 2004035-14:05:00   TRAPDATA :  b .       %  xb          b
u                              j
CMR713I - 001 2004035-14:05:00    " "  :
<        &
CMR713I - 001 2004035-14:05:00       "  "   :                            &
CMR713I - 001 2004035-14:05:00    " "  :                           <
(
CMR709I - 001 2004035-14:05:00 SUBMIT member IEFBR14
CMR720I - 001 2004035-14:05:00 SCH Rules processing complete

CMR702I - 001 2004035-14:06:00 SELECT statement #00000003 matched.  SCH Application
Agent request data:

CMR701I - 001 2004035-14:06:00 SCH Rules processing started, using "RULE003 " rule
dated 2004035-14:04:40
CMR707I - 001 2004035-14:06:00 MESSAGE sent to console: SCH RULE003 MSG

CMR701I - 001 2004035-14:06:00 SCH Rules processing started, using "RULE004 " rule
dated 2004035-14:04:40
CMR707I - 001 2004035-14:06:00 MESSAGE sent to console: SCH RULE004 MSG
CMR720I - 001 2004035-14:06:00 SCH Rules processing complete
CMR702I - 001 2004035-14:06:00 SELECT statement #00000001 matched.  SCH Application
Agent request data:

CMR701I - 001 2004035-14:06:00 SCH Rules processing started, using "RULE001 " rule
dated 2004035-14:04:40
CMR708I - 001 2004035-14:06:00 EXECUTE program IEFBR14
CMR706I - 001 2004035-14:06:00 COMMAND sent to console: D NET,ID=RULE001
CMR707I - 001 2004035-14:06:00 MESSAGE sent to console: SCH RULE001 MSG
CMR711I - 001 2004035-14:06:00 NOP
CMR712I - 001 2004035-14:06:00 SNMPTRAP Description:
CMR713I - 001 2004035-14:06:00    TEXT (U) : SCH RULE001 TRAP
CMR713I - 001 2004035-14:06:00    TEXT (R) : SCH RULE001 TRAP
CMR713I - 001 2004035-14:06:00    IPADDR   : EP2K.CSG.STERCOMM.COM (10.20.4.1)
```

Continued

```
CMR713I - 001 2004035-14:06:00    PORT     : 162
CMR713I - 001 2004035-14:06:00    GROUP1   : 2
CMR713I - 001 2004035-14:06:00    GROUP2   : 2222
CMR713I - 001 2004035-14:06:00    TRAP OID : 1.3.6.1.4.1.1733.4.2.1.4.2222
CMR713I - 001 2004035-14:06:00  TRAPDATA : b .     % xb       b
j
CMR713I - 001 2004035-14:06:00    "  "   :
<        &
CMR713I - 001 2004035-14:06:00      "   "    :                              &
CMR713I - 001 2004035-14:06:00    "  "  :                            <
(
CMR709I - 001 2004035-14:06:00 SUBMIT member IEFBR14
CMR720I - 001 2004035-14:06:00 SCH Rules processing complete

CMR702I - 001 2004035-14:07:00 SELECT statement #00000001 matched.  SCH Application
Agent request data:

CMR701I - 001 2004035-14:07:00 SCH Rules processing started, using "RULE001 " rule
dated 2004035-14:04:40
CMR708I - 001 2004035-14:07:00 EXECUTE program IEFBR14
CMR706I - 001 2004035-14:07:00 COMMAND sent to console: D NET,ID=RULE001
CMR707I - 001 2004035-14:07:00 MESSAGE sent to console: SCH RULE001 MSG
CMR711I - 001 2004035-14:07:00 NOP
CMR712I - 001 2004035-14:07:00 SNMPTRAP Description:
CMR713I - 001 2004035-14:07:00    TEXT (U) : SCH RULE001 TRAP
CMR713I - 001 2004035-14:07:00    TEXT (R) : SCH RULE001 TRAP
CMR713I - 001 2004035-14:07:00    IPADDR   : EP2K.CSG.STERCOMM.COM (10.20.4.1)
CMR713I - 001 2004035-14:07:00    PORT     : 162
CMR713I - 001 2004035-14:07:00    GROUP1   : 2
CMR713I - 001 2004035-14:07:00    GROUP2   : 2222
CMR713I - 001 2004035-14:07:00    TRAP OID : 1.3.6.1.4.1.1733.4.2.1.4.2222
CMR713I - 001 2004035-14:07:00  TRAPDATA : b .     % xb       b
j
CMR713I - 001 2004035-14:07:00    "  "   :
<        &
CMR713I - 001 2004035-14:07:00      "   "    :                              &
CMR713I - 001 2004035-14:07:00    "  "  :                            <
(
CMR709I - 001 2004035-14:07:00 SUBMIT member IEFBR14
CMR720I - 001 2004035-14:07:00 SCH Rules processing complete

CMR702I - 001 2004035-14:08:00 SELECT statement #00000003 matched.  SCH Application
Agent request data:

CMR701I - 001 2004035-14:08:00 SCH Rules processing started, using "RULE003 " rule
dated 2004035-14:04:40
CMR707I - 001 2004035-14:08:00 MESSAGE sent to console: SCH RULE003 MSG

CMR701I - 001 2004035-14:08:00 SCH Rules processing started, using "RULE004 " rule
dated 2004035-14:04:40
CMR707I - 001 2004035-14:08:00 MESSAGE sent to console: SCH RULE004 MSG
CMR720I - 001 2004035-14:08:00 SCH Rules processing complete
```

The following is detailed information concerning the previous example:

Trace Message ID #	Description
CMR702I	Indicates that the request matched all the criteria for a Select statement, and gives the number of that SELECT statement. This sequence number is the same number listed in columns 73-80 on the SYSPRINT listing.
CMR701I	Lists the rule selected for processing and gives the rule's date and time stamp. This trace message is important if you REFRESH the rules while Connect:Enterprise is online, because it indicates which rules were used by the date and time stamp on the rules SYSPRINT header.
CMR708I, MR706I, CMR707I, CMR711I, CMR712I, CMR713I, and CMR709I	List the instructions in the order they execute.
CMR712I	Indicates that an SNMP trap is being built and its description will follow.
CMR713I	Describes the SNMP trap in detail. See *CMR713I* on page 105 for more details.
CMR720I	Indicates processing is complete for the Select statement.

Chapter 4

Using Connect:Enterprise Online Exits

Several optional user exits are provided to customize the online execution of Connect:Enterprise. Connect:Enterprise calls these online exits at the appropriate time during Connect:Enterprise processing and can set an action code before returning to alter the standard Connect:Enterprise processing. Connect:Enterprise only provides the linkage to the online exits. You must define, code, assemble, link, and test your own exits. Online exit programs are optional and the system default is no exits.

You can supply several user exit programs that are called by online Connect:Enterprise. The exits available are as follows:

- Input exit
- Session security exit (FTP)
- Security exit one (BSC, SNA)
- Security exit two (BSC, SNA)
- Output exit
- End Of Batch exit
- Initialization exit
- Termination exit
- Log exit
- APPC security exit
- CICS Wake Up Initiate exit
- CICS Wake Up Terminate exit

The online exits are called by Connect:Enterprise using standard CALL linkage. Therefore, when the exits are in control, Connect:Enterprise is not. Define, assemble, link, and test your programs carefully because, when an ABEND occurs or the system goes into a loop, the entire Connect:Enterprise system is affected.

An exit trace feature is provided to aid in testing and debugging of online exits. This trace feature can be invoked to snap the information passed to and from the exits before and after each CALL.

This chapter describes the different types of online exits and their uses.

Note: You can use the sample member, ASMLKXIT, to assemble and link your user exits.

How Connect:Enterprise Uses Online User Exits

All Connect:Enterprise online exits are called using a standard z/OS CALL issued by an assembler language program. Sample user exit programs demonstrating the necessary entry and return coding requirements are provided for each exit.

An exit parameter list is passed to each online exit. The parameter list points to all of the data that can be examined and modified by the exit programs. The calling parameters and action codes for each of the user exit programs are documented with the descriptions for each type of user exit. The parameter list is a series of 4-byte addresses that point to the actual data field. The data field contents description uses letter codes to describe the field format. C is character data; X is hexadecimal data.

The Connect:Enterprise release tape includes several macros containing the parameter lists:

Macro	Description
M$XPARM	DSECTs for the input, security one, security two, and output exits.
M$SECMAP	DSECTs for the session security exit (FTP).
M$OUXCB	DSECT for the offline utility exits.
M$ACREC, M$LOGB, and M$DCREC	DSECTs for the Auto Connect record, remote connect records, and Auto Connect queue records. Use these macros when coding a log exit.
M$BCREC	DSECT for the VCF batch control record used by End Of Batch exit.

DSECTS are not provided for the other exits. Use of the DSECTs for coding assembler language user exit programs is demonstrated in the sample exit programs provided on the release tape.

Assembler language programs must save and restore the calling program's registers. At entry to the user exit, the parameter list address is in register 1.

COBOL programs must provide a LINKAGE SECTION that describes the parameter list passed to the user exit. A USING for the parameter list must be included in the PROCEDURE DIVISION. A single-byte hexadecimal action code is passed back to Connect:Enterprise, as demonstrated in the STCOBOL sample user exit.

Implementing Online Exits

Online exits are activated by control records in the Options Definition File (ODF) when you are using the online Connect:Enterprise. A keyword parameter in the *OPTIONS section indicates the exit is being used and supplies the name of the load module to be loaded and executed.

For example, to initiate these exit programs (STXINIT, STINP, STSECFTP, STSEC1, STSEC2, STOUT, STEOBX, STTERM, and STLOGX), use the following *OPTIONS parameters:

```
*OPTIONS
    ...
    ...
    XINIT=STINIT
    XINPUT=STINPS
    X_SECURE=STSECFTP
    XSECUR1=STSEC1
    XSECUR2=STSEC2
    XOUTPUT=STOUT
    XENDOFB=STEOBX
    XTERM=STTERM
    XLOG=STLOGX
```

These parameters along with others related to implementing user exits are discussed in the *Connect:Enterprise for z/OS Administration Guide*. See the chapters related to the ODF in that book for more information. Connect:Enterprise loads these exits and gets storage for exit-related control blocks during system initialization. If the programs cannot be found in your JOBLIB or STEPLIB, Connect:Enterprise issues an appropriate console message and terminates.

Coding User Exits

The following requirements apply to all online exits. You must follow these rules for proper execution of all exits.

- For maximum performance, code your exits to run in 31-bit addressing mode. Existing exits that are coded to run in 24-bit addressing mode *are* supported; however, you must upgrade them to 31-bit addressing mode to avoid performance degradation.
- When changing parameters passed to your exit programs, always use values that are valid for use by Connect:Enterprise. Failure to use valid data can cause system ABENDs or unpredictable results. All character fields are passed as left-justified, blank-filled, and should be returned in the same format.
- The passed parameter list always contains valid addresses for all fields defined for each user exit. However, the data field pointed to by a parameter address sometimes does not contain data and may be set to blanks. The data fields that are filled in depend on the transaction type being used.

  For example, a $$ADD transaction does not supply a password, so the password data field is not used. For details on the setting of data fields, you may want to use a dummy exit program as described in this chapter.
- Set a valid action code. The default action code is X‘00’, which indicates that Connect:Enterprise should continue processing as normal. If an invalid action code is set, Connect:Enterprise forces an action code of X‘00’.
- Connect:Enterprise provides one 256-byte user work area for each remote site that logs onto your Connect:Enterprise system. The values you save in this area are related to *individual remotes* being processed by the system, and are not tied to a specific user exit. Therefore, you can use the user work area to pass data for any remote between online exits. It also means that your online exits, when properly written, can handle many simultaneously-active transactions on many different lines defined to Connect:Enterprise. The exit programs *do not* need to be

true re-entrant programs since they are not interrupted for new calls while they are processing a call. They should, however, use the user work area to pass information between exits or to save information for future entries to the same exit for each remote session.

- The FTP session security exit must be coded re-entrant, since multiple FTP threads may concurrently invoke your exit.
- If you want to reinitialize the user work area, you must do so in your online exits. The user work area is initialized to binary zeros at system startup, but is never again referenced or used by Connect:Enterprise. If you require more than 256 bytes, set the GETMAIN storage for each individual user work area and save the address of the GETMAIN storage in the user work area.

  The 256-byte user work area is not available to Initialization, Termination, Log, or End Of Batch exits.

Note: While your online exits are executing, Connect:Enterprise is waiting for them to return from the CALL and is not processing other transactions. For this reason, avoid writing exits that will excessively slow down the system. This is particularly important when using the Input exit or Output exit, which is called for every I/O to a remote site.

- Use the Initialization and Termination exits to allocate and open special files to be used by all sessions or to GETMAIN storage to pass system level information between sessions and/or online exits. These exits provide you with a method of passing global system information between sessions and online exits.
- Use an address in the one-word (4-byte) area available for storing information from the Initialization exit. This address can then be the address of a GETMAIN area, which can be further interrogated and modified by other exits or sessions.

  The value specified in this one-word area word is passed to each user exit and to the Termination exit. You *must* close any open files, deallocate any allocated files, and FREEMAIN any storage acquired by the Initialization exit. It is recommended that these functions be performed in the Termination exit.

Testing Online Exits

Fully test online exits before you implement them into a production Connect:Enterprise system. An ABEND or loop in a user exit affects all of Connect:Enterprise, not just your exit load module. Use the following guidelines when testing online exits:

- Test online Connect:Enterprise with the exits, then test all offline utilities to ensure that they execute properly. Testing the EXTRACT utility is particularly important if you use the Input exit to modify input data. The EXTRACT offline utility relies on the presence of proper control characters when it extracts batches. If you use the Log exit to modify Auto Connect or remote connect log records, test the REPORT offline utility.
- If you access the BATCHID field in your exits, test the many different forms that it can take. BATCHID can be omitted entirely (and set to blanks); or it can be a character value; a generic value where the first byte indicates the length of generic BATCHID; or the Connect:Enterprise-assigned 7-digit packed batch number.
- Review the system console and data set for warning messages if you experience ABENDs during your testing. Connect:Enterprise sets a flag when it passes control to your exit, then resets the flag upon return. Connect:Enterprise also uses the ESTAE system macro to trap

ABENDs that occur in the system. If an ABEND occurs and the flag that indicates an exit is in control is set, Connect:Enterprise issues a message to the system console and creates a small snapshot dump in the SNAPOUT data set. The following messages displays on the system console:

```
CMB079W -WARNING: ABEND DURING USER EXIT (xxxxxxxx).  PROBABLE USER EXIT ERROR.
(xxxxxxxx = XINPUT, XOUTPUT, XSECUR1 XSECUR2, XENDOFB, XINIT, XTERM, or XLOG)
```

The SNAPOUT data set contains a small snap with the following title:

```
STMC01   ABEXIT - SCB - ABEND DURING USER EXIT
```

Tracing the User Exits

Connect:Enterprise provides many TRACE capabilities for the online system through the use of TRACE parameters in the ODF *OPTIONS section or by using the TRACE console command. One of these values may be helpful in testing your online exits.

Creating an Options Definition Trace Record

Specify TRACE=EXITS in the ODF to cause Connect:Enterprise to create a small snapshot dump on the standard Snapshot data set. Use the following in your ODF within the *OPTIONS section:

```
TRACE=EXITS
```

Using the Console Trace Command

The trace command can be entered as a console command:

```
$$TRACE EXITON .....  starts trace
$$TRACE EXITOFF .....  stops trace
```

Note: The exit trace feature is not available for the Initialization exit or the Termination exit. No trace data is provided for these exits.

Specifying Trace Output

When specified, Connect:Enterprise snaps different data depending on the exit, as follows:

- The Exit Control Block (XCB) is snapped before and after each CALL to the Input exit, Output exit, Security exit one and Security exit two. Passwords used by the security interface in the XCB are not snapped; they are replaced with question marks.
- The batch control record that was just added is snapped for the End Of Batch exit.
- The calling parameter list and the log record are snapped before and after each call to the Log exit.

Understanding the Exit Control Block

The XCB contains most of the important data passed to and from the exits. The data in the XCB includes the following:

- User exit work area
- Calling parameter data field values which are actually passed to the online exits. You can code your exits to change some of these values.
- Calling parameter list

Character values indicate the start of many of the sections in the snapshot data set.

For example, the words XCB USER AREA appear just before the start of the user exit work area. If you want more details on the exact layout of the Exit Control Block, refer to its DSECT in the M$XCB macro supplied on the Connect:Enterprise release tape.

The snapshot dumps have appropriate titles that indicate whether they were obtained before or after the CALL to your exit program.

One area that is not contained in the XCB is the Connect:Enterprise input and output buffers, which you may want to examine if you are testing an Input exit or Output exit. To include this area in the Snapshot Data Set (interleaved with your exit trace), use the TRACE=ALLTP or TRACEID=xxxxxxxx parameters in the *OPTIONS records or use the corresponding trace console commands.

For APPC activity, the TRACE feature also snaps the IPS passed to and from the exits. However, when these traces are used, massive output is generated to the snapshot data set.

Using the Exit Trace for Log Exit

If you specify XLOG=xxxxxxxx and TRACE=EXITS in the ODF, Connect:Enterprise snaps the calling parameter list and the log record before and after each call to the Log exit program. The snap dump title contains the words PARMS, LOGREC to identify the snap contents. The first seven full words of the snap data contain the calling parameters passed to the Log exit. These seven full words are followed by:

- 1-byte exit type field (6=Log exit)
- 1-byte return code field (X‘00’)
- 1-byte log function type (1=PUT NEW 2=UPDATE)
- 18-byte log record key
- 256-byte area that contains log record data, blank-filled
- 2-byte log data length
- 4-byte initialization exit word

Note: Some of the data that follows the seven fullword calling parameters is a COPY of the actual data. This data was moved adjacent to the calling parameters to reduce the number of snaps required to display all the data. Therefore, the calling parameters may not always point to the data displayed in the snap.

Using a Dummy Exit Program

If you are unsure of the possible values set by Connect:Enterprise in some of the data fields passed to the exits, a dummy exit program combined with the use of TRACE=EXITS may answer many of your questions.

Code, assemble, and link a small dummy exit program, as shown in the following example, which simply returns immediately to the caller:

```
//DUMMYX              JOB ...as required for your shop
 //*****************************************************
 //* ASSEMBLE AND LINK DUMMY EXIT PROGRAM
 //*****************************************************
 DUMMYX EXEC  ASMHFCL,PARM.LKED=(NCAL,LIST,XREF)
 //ASM.SYSIN DD *
          BR 14
          END
```

After creating the load module, named X$DUMMY in this example, execute Connect:Enterprise using the dummy exit load module name as one or all of the user exit names. Be sure to specify TRACE=EXITS.

For example, to examine all data passed to and from Security exit one and Security exit two, specify the following records in the *OPTIONS section:

```
*OPTIONS
    ...
    XSECUR1=X$DUMMY
    XSECUR2=X$DUMMY
    TRACE=EXITS
```

Use a remote site to send a variety of $$ commands to Connect:Enterprise. Try a wide range of parameters on the $$ commands and create some security violations to cause entry to both dummy exits. Then bring down Connect:Enterprise, and print and examine the SNAPOUT data set, referring to the XCB DSECT in the M$XCB macro supplied with Connect:Enterprise.

The SNAPOUT data set can be directed to SYSOUT for print or can be printed from a file.

Sample Online Exits

The Connect:Enterprise release tape contains several sample user exit programs. All programs except STCOBOL are written in assembler language. STCOBOL shows how to use the online exits if you are writing them in COBOL. Use the sample programs as guidelines in coding your own online exits:

Sample User Exit Programs	Description
STINP	Input exit (BSC)
STINPS	Input exit (SNA)
STSECFTP	Session security exit (FTP)
STSEC1	Security exit one (BSC, SNA)
STSEC2	Security exit two (BSC, SNA)
STTERM	Termination exit
STEOBX	End Of Batch exit
STEOBX2	End Of Batch exit (2nd example)
STXINIT	Initialization exit
STLOGX	Log exit
STOUT	Output exit
STCSEC	APPC Security exit
STCWI	CICS Wake Up Initiate exit
STCWT	CICS Wake Up Terminate exit
STCOBOL	Security exit one (COBOL sample)

Using the Input Exit

The Input user exit is called after every input completion, when Connect:Enterprise is receiving data from a remote site. The Input exit can examine and even change data in the input buffer, as long as the maximum buffer size is not exceeded. If data is changed, you *must* still retain the proper line and record control characters.

The Input exit can set an action code that requests Connect:Enterprise to continue normally, to use the data as changed by the exit, or to ignore the data and set up to receive more input from the sender.

Input Exit Parameters

Parameters passed to the Input exit are addressed by the X3$DSECT in the M$XPARM macro. A listing of the parameters and the contents of the data field pointed to by the parameters is in the following table:

DSECT Label	Contains Address of	Data Field Contents
X3$XTYPE	Exit Type Code	1-byte Exit Type Code C‘3’ = Input exit
X3$ACODE	Area to return Action Code	1-byte exit action code X‘00’ = Process as Normally X‘04’ = Input Area was Changed X‘08’ = Ignore this Input Block
X3$WORKA	User Work Area	Address of 256-byte user work area
X3$LINID	Remote Name	8-byte Remote Name from which the input data was received
X3$ICTYP	Input Completion Type	1-byte Input Completion Type If BSC: C‘1’ = Initial Input with normal completion C‘2’ = Subsequent Input with normal completion C‘3’ = Abnormal Input or no data If SNA: C‘1’ = All Input with normal completion C‘3’ = Abnormal Input or no data
X3$IMAXL	Maximum Length of Input Area	2-byte Maximum Length of Input Area (hexadecimal)
X3$INPTL	Length of Input Data	2-byte Length of Input Data received, including all control characters (hexadecimal)
X3$INPTA	Input Area	Input data received from remote. Maximum size in X3$IMAXL, Actual size in X3$INPTL.
X3$LNTYP	Line Type	1-byte Line Type Code indicator: X‘01’= BSC Switched X‘02’= BSC Non switched X‘03’= SNA LU Type 1
X3$INIT@	Initialization exit word	1 word containing the address or information exit word returned from the Initialization exit. This value is all X ‘00’ if no Initialization exit is used.

The Input exit can change four fields pointed to by the parameter list. They are as follows:

- X3$ACODE
- X3$WORKA
- X3$INPTL
- X3$INPTA

Input Exit Requirements

The following requirements apply to the Input exit. You must follow these rules for proper exit use.

- The purpose of the Input exit is to allow normal text message examination and modification. Do not attempt to use this exit to do your own I/O or line error recovery.
  - For SNA sites—The Input exit is called for standard text input and for the receipt of SNA Function Management Headers. The exit is not called for nontext input such as SNA commands or negative responses.
  - For BSC sites —The Input exit is called for standard text input and for the receipt of EOT or DLE/EOT (End of Transmission or Disconnect). The exit is not called for nontext read completions, such as RESETPL completions and temporary text delays.
- Use the following rules when changing the input data:
  - If you change the length of the input data you must set the new data length in the X3$INPTL parameter. The length must never exceed the value pointed to by X3$IMAXL.
  - Because the actual Connect:Enterprise input buffer is passed to you, use caution when modifying data that you do not overlay other areas and cause a system ABEND.
  - You must retain all SNA record control, such as SCB, IRS, and NL. Both online Connect:Enterprise and the offline utilities rely on the presence of proper control characters.

Note: Connect:Enterprise does not clear all old data out of the input buffer before issuing a Receive. There may be some data remaining in the input buffer from a previous transaction. Always use the input length parameter that is pointed to by X3$INPTL to determine the length of the current data.

- For abnormal read completions (input completion type = C‘3’), the length of the input data is set to binary zeros. Do not access the input buffer, since the data present is not reliable.
  - For BSC sites —If the input buffer begins with an EOT (X‘37’) or DLE/EOT (X‘1037’), then this is the last input block for a batch. When this situation occurs, the input buffer does not contain any text data. In this instance, action code X‘00’ should always be set. If the remote site sends BSC transparent data, remember that the input buffer begins with DLE/STX (X‘1002’) rather than the standard STX (X‘02’). An input completion type of C‘1’ is set only for the completion of a BTAM READ INITIAL.
- An action code of X‘04’ or X‘08’ is not allowed and is ignored in these instances:
  - Abnormal read completions

- Receipt of SNA Function Management Headers (SNA sites only)
- Last block of batch EOT or DLE/EOT (BSC sites only)

Sample Input User Exit STINPS (SNA Only)

The STINPS sample user exit program demonstrates how to do the following:

- Examine and change data in the input buffer before Connect:Enterprise examines the data
- Ignore blocks that contain a certain text header

STINPS Program Logic

The STINPS program performs the following processing:

- Scans the input data for the text SPECIAL HEADER= and ignores data records that contain this text. The STINPS program then sets an action code of X‘08’ and returns to Connect:Enterprise to ignore the input.
- If the special header is not found, the STINPS program scans the text for a special ID parameter within the first 40 bytes of the block. If an ID is found that begins with the characters ST, then it is a 10-byte ID which must be changed to an 8-byte ID before Connect:Enterprise examines the input. If ID=ST is found, STINPS overlays the ST with the 8 bytes that follow. Notes that the message length was not changed, so the X3$INPTL value need not be changed. The program sets an action code of X‘04’ if the data was changed and returns to Connect:Enterprise.

Implementing STINPS

To run STINPS with online Connect:Enterprise:

1. Insert the following information into your ODF:

```
*OPTIONS
   ...
   XINPUT=STINPS
```

2. Insert the following header data to test the removal of blocks. The special header record must be the first record in the block.

```
SPECIAL HEADER=SIGNON FROM BRANCH001
Data record 1
Data record 2
...
```

3. Test the use of 10-byte IDs. The $$ADD records must occur on a block boundary, such as:

```
$$ADD ID=STTESTEXIT BATCHID='TEST STINPS'
....
$$ADD ID=STXXXXXXXX BATCHID='TEST STINPS'
....
```

Sample Input User Exit STINP (BSC Only)

The STINP sample user exit program demonstrates how to do the following:

- Examine and change data in the input buffer before Connect:Enterprise examines the data
- Ignore blocks that contain a certain text header

STINP Program Logic

The STINP program performs the following processing:

- If the input area contains a final input (EOT or DLE/EOT), the STINP program returns to Connect:Enterprise with an action code of 00, which indicates to continue processing.
- Scans the input data for the text SPECIAL HEADER= and ignores data records that contain this text.
- If the special header is found and the input contains blocked records, the STINP program removes the single record that contains the special header and adjusts the input length (pointed to by X3$INPTL) for the record removed. The program then sets an action code of X'04' to tell Connect:Enterprise the data was changed.
- If the special header is found and the input contains only a single record, the STINP program ignores the record. The program then sets an action code of X'08' and returns to Connect:Enterprise to ignore the input.
- If the special header is not found or has been removed from blocked data, the STINP program scans the remaining text for a special ID parameter. If an ID is found that begins with the characters ST, then it is a 10-byte ID, and it must be changed to an 8-byte ID before Connect:Enterprise examines the input. If ID=ST is found, the program overlays the ST with the 8 bytes that follow it. Note that the message length was not changed, so the X3$INPTL value need not be changed. The program then sets an action code of X'04' if the data was changed and returns to Connect:Enterprise.

Implementing STINP

To run STINP with online Connect:Enterprise:

1. Place the following records into your ODF:

```
*OPTIONS
   ...
   XINPUT=STINP
```

2. Test the removal of blocks with a special header by using data, such as the following:

```
SPECIAL HEADER=THIS BLOCK SHOULD BE IGNORED
RECORD 1
RECORD 2
SPECIAL HEADER=THIS BLOCK SHOULD BE IGNORED
RECORD 3
```

3. Test the use of 10-byte IDs, such as the following:

```
$$ADD ID=STTESTEXIT BATCHID='TEST STINP'
....
$$ADD ID=STXXXXXXXX BATCHID='TEST STINP'
....
```

Using Session Security Exit

The session security exit enables users to extend or replace the standard Connect:Enterprise security interface functions. The session security exit is used for FTP sessions only.

The session security exit is optional. You must define, code, assemble, link, and test your own session security exit.

To have Connect:Enterprise call the exit, specify the name of the load module you created in the X_SECURE parameter in the ODF *OPTIONS record. A sample exit (STSECFTP) is included in the source library as a starting point for you.

Session Security Exit Parameters

The following table describes the parameters that are passed to the session security exit. This parameter list is documented by the M$SECMAP macro in the source library:

Field Name	Contains Address of	Data Field Length	Data Field Type	Data Field Contents
E1$TYPE	Exit Type Code	1	Character	1 = Session Security Exit (FTP server) 2 = Session Security Exit (FTP client)

Field Name	Contains Address of	Data Field Length	Data Field Type	Data Field Contents
E1$ACTCD	User Action Code (1)	1	Character	1 = Continue as normal 2 = Bypass all remaining security checks. 3 = Process as a security violation. 4 = Process as a security violation and terminate the session. 5 = Use reply number and text in E1$MSG as Reply substitution If the security interface uses the ODF *OPTIONS parameter MBXSECURE=, the security interface is called immediately after returning from the session security exit. The security interface is called after the exit returns control from processing the commands listed below: ◆ For Logon Security: PASS command. ◆ For Batch Function Security: DELE, LIST, NLST, RETR, STOR, and STOU commands. For complete information about using the security interface, refer to *Connect:Enterprise for z/OS User's Guide*.
E1$WORK	User Work Area	256	User-defined	Specified by user.
E1$MSG	User Message Area	256	Character	Message text sent if User Action Code is 3, 4, or 5.
E1$CMDTP	Command Type Code	1	Character	1 = The Logon function is set if the command is PASS. 2 = Session functions involving data transfer are set if the command is one of the following: LIST, NLST, RETR, STOR, or STOU. 3 = Session functions not involving data transfers are set if the command is DELE, APSV, or SITE.

Field Name	Contains Address of	Data Field Length	Data Field Type	Data Field Contents
E1$CMD	Command	4	Character	The 4-character FTP command code. The exit is called for the following commands: APSV, CWD, DELE, LIST, NLST, PASS, PASV, PORT, RETR, RNFR, RNTO, SITE, STOR, STOU, USER. **Note:** When E1$CMD = “CWD”, you can modify the E1$ID (Mailbox ID) value. When E1$CMD = “STOR” or “STOU”, you can modify both the E1$ID and E1$BID (BATCHID) values. When E1$CMD='LIST' or 'NLST', you can modify the E1$BID (Batch ID), E1$BIDL (Batch ID Length), and E1$LOPT (List Options) values.
E1$TEXTLN	Command String Text Length	2	Hex	Halfword value of the length, in hex, of the entire FTP command string received by the remote, including the trailing 1-byte x'15' new line character.
E1$TEXT	Command String	variable	Character	The FTP command string received by the remote including the trailing 1-byte x'15' new line character.
E1$RMTNM	Remote Name	8	Character	The remote name/user ID provided by the client, padded with trailing blanks.

Field Name	Contains Address of	Data Field Length	Data Field Type	Data Field Contents
E1$PASSWD	Password	8	Character	Password provided by the client, padded with trailing blanks. This field is set to binary zeros on a PASS command. When processing a PASS command (when E1$CMD=PASS), the password can be obtained from the command string value contained in E1$TEXT. The format of the PASS command string is: PASS *password* or PASS *password*/*newpass*/*newpass* When parsing the PASS command string, allow for multiple blanks between PASS and *password*. If the user exit makes Security Access Facility (SAF) calls directly, make sure to uppercase the USERID and PASSWORD values prior to issuing the RACROUTE macro. If upper casing is not performed by the user exit, the FTP client will be required to send these values as uppercase in the USER and PASS command strings (the security packages use uppercase values).
E1$ID	Mailbox ID (Current Working Directory)	8	Character	The mailbox ID specified by the client, padded with trailing blanks. **Note:** You can modify this value when E1$CMD =“STOR”, “STOU”, or “CWD”.
E1$BID	User Batch ID (File Name)	64	Character	User Batch ID specified by the client, padded with trailing blanks. **Note:** You can modify this value when E1$CMD = “STOR”, “STOU”, “LIST”, or “NLIST”.

Field Name	Contains Address of	Data Field Length	Data Field Type	Data Field Contents
E1$BID	User Batch ID length	2	Hex	Halfword value of the length, in hex, of the User Batch ID received in the LIST or NLST command. Length of User Batch ID in the LIST/NLST command text string. **Note:** You can modify this value when E1$CMD='LIST' or 'NLST'. If Batch ID is changed by the exit (pointed to E1$BID), this field should also be set by the exit, indicating the new length to be used for LIST and NLST processing.
E1$LOPT	List Options	1	Character	1 = Use first 24 bytes of User Batch ID 2 = Use Last 24 bytes of User Batch ID **Note:** You can modify this value when E1$CMD='LIST' or 'NLST'.
E1$LOGST	Logon status of the remote/user	1	Character	1 = User has not logged on yet. 2 = User has completed logon.
E1$SSLST	Secure Connection Status	1	Character	1 = This is not a secure connection. 2 = This is a secure connection using SSL.
E1$ASITE	Current SITE Values Table	4	Address	Pointer to a table that contains all the values set by previous SITE commands. The SITE Values Table is mapped by the M$SITEMP macro, located in the source library.

The User Work Area and User Message Area are the only parameters that the exit updates.

Session Security Exit Requirements

The session security exit requirements are:

- The exit load module must be placed in the Connect:Enterprise load library.
- The exit must be coded to be reentrant.
- The exit is called using standard z/OS call linkage.
- The exit is entered in the following environment:
  - Task mode
  - Problem state

- User key (storage key 8)
- AMODE 31
- Unlocked
- Non CMS (HASN=PASN=SASN)
- Primary ASC (non AR mode)

✦ Communications between Connect:Enterprise and an installation exit is done through the parameter lists.

When changing parameters passed to the exit, always use values that are valid for Connect:Enterprise use. Using invalid data can cause unpredictable results, including a system ABEND.

All character field parameters passed to and from the installation exit are left justified and padded to the right with blanks.

The parameter lists always contain valid addresses for all fields defined for each installation exit. Data fields pointed to by a parameter address may contain data or may be set to blanks.

✦ The exit must preserve the contents of general-purpose register (GPR) 14. It does not need to preserve the contents of other registers.

Configure the exit to use the standard 18-word (72-byte) register save area (RSA), whose address is in register 13.

Connect:Enterprise restores the contents of general-purpose registers 0 through 13 and 15 upon return from the exit.

✦ Connect:Enterprise provides one 256-byte work area for each FTP session in progress. Information you store in this work area is retained throughout the session. No information is shared between sessions. No information is retained after a session completes.

Connect:Enterprise initializes the 256-byte work area to binary zeros at session startup. Connect:Enterprise does not reference the 256-byte work area after initialization. You can reinitialize the area from your exit. If you require more than 256-bytes, allocate virtual storage and store the address and length of the allocated storage in the 256-byte work area.

The following table shows register contents on security exit entry and the expected register contents on exit from the security exit. The exit should not use access registers.

GPR	Contents on Entry	Expected Contents on Exit
0	Binary zero	Ignored
1	Address of parameter list	Ignored
2-12	Binary zero	Ignored
13	Address of standard register save area	Ignored
14	Return Address	Return Address passed on entry
15	Entry Address	Ignored

Sample Session Security Exit STSECFTP (FTP only)

The STSECFTP sample user exit program specifies one of the following actions:

- Security check passed. Continue with the security interface check (if active).
- Security check passed. Bypass the security interface check.
- Security check failed. Process as a security violation and continue the session. Connect:Enterprise stops processing the command.
- Security check failed. Process as a security violation and terminate the session. Connect:Enterprise stops processing the command and terminates the connection.

The session security exit can specify additional error message information when it returns a security violation. The error message information is formatted into a reply, and a reply code prefix is added before the reply is sent to the remote FTP. The following shows a sample security exit error message:

```
550-Improper authorization.
550 USER command failed.  Request denied by session security exit.
```

The STSECFTP sample user exit program can also use reply numbers and text stored in E1$MSG as reply substitution to RNFR and RNTO commands. See coding details in sample STSECFTP program.

STSECFTP Program Logic

The STSECFTP program performs the following processing:

- Examines the input command type flag (i.e. logon, data transfer, session control), and indicates to continue processing as normal.

Implementing STSECFTP

To run STSECFTP with online Connect:Enterprise, use an ODF that contains the following:

```
*OPTIONS
   ...
   X_SECURE=STSECFTP
```

Using Security Exit One

Security exit one can be used to do the following:

- Examine and override values used in all remote $$ commands or in a data collection without $$ADD.
- Provide security in addition to or instead of the Connect:Enterprise security.

Security exit one is called after Connect:Enterprise detects and reformats a $$ command ($$ADD, $$REQUEST, $$DELETE, $$LOGOFF, and $$DIRECTORY) from a remote site. It is also called for new data batches received without a $$ADD record and for attempted LOGONs to

Connect:Enterprise. The exit can examine and change values set by the $$ commands, as long as standard Connect:Enterprise values are used.

This exit can set an action code that requests Connect:Enterprise to continue normally, to bypass all ID validation, or to process the transaction as a security violation.

At SNA sites, this user exit can provide a substitute security violation message to replace the Connect:Enterprise default message.

Security Exit One Parameters

Parameters passed to Security exit one are addressed through the X1$DSECT in the M$XPARM macro. A list of the parameters and the contents of the data field pointed to by the parameters is shown in the following table.

DSECT Label	Contains Address of	Data Field Contents
X1$XTYPE	Exit Type Code	1-byte Exit Type Code C‘1’ = Security exit one
X1$ACODE	Area to return Action Code	1-byte exit action code X‘00’ = Continue as normal X‘04’ = Bypass MBXSECURE and SECURITY=BATCH ID Validation X‘08’ = Process as Security Violation
X1$WORKA	User Work Area	256-byte user work area
X1$LINID	Remote Name or BSC Line ID	8-byte Remote Name for which the transaction is in progress. If the Remote Name is not available for BSC, the Line ID is used.
X1$CTYPE	$$ Command Type	1-byte $$ Command Type C‘1’ = $$ADD C‘2’ = Missing $$ADD C‘3’ = $$REQUEST C‘4’ = $$DIRECTORY C‘5’ = $$DELETE C‘6’ = $$LOGOFF C‘7’ = SNA LOGON C‘8’ = BSC LOGON
X1$ID	Mailbox ID	8-byte Mailbox ID used
X1$BCHID	BATCHID	64-byte user BATCHID used
X1$PASSW	Password	8-byte Password entered
X1$BLOCK	Block Factor	1-byte Block Factor (hexadecimal) (Not used in SNA systems.)

DSECT Label	Contains Address of	Data Field Contents
X1$XMIT	XMIT Value	1-byte XMIT Value C‘Y’ = XMIT = Y C‘N’ = XMIT = N
X1$MULTX	MULTXMIT Value	1-byte MULTXMIT Value C‘Y’ = MULTXMIT = Y C‘N’ = MULTXMIT = N
X1$TRUNC	TRUNC Value	1-byte TRUNC Value (not used in SNA systems.) C‘Y’ = TRUNC = Y C‘N’ = TRUNC = N
X1$CMP	CMP Value	1-byte CMP Value (not used in SNA systems.) C‘Y’ = CMP = Y C‘N’ = CMP = N
X1$RCSEP	RECSEP Value	1-byte RECSEP used (not used in SNA systems.)
X1$MEDIA	MEDIA Value	1-byte MEDIA used (not used in BSC systems) X‘00’ = Console media X‘10’ = Transmission Exchange media X‘11’ = Basic Exchange Diskette X‘20’ = Card punch or card reader X‘30’ = Printer X‘FF’ = No explicit media specified
X1$APLID	APPLID Value	8-byte APPLID used (LUNAME when SPC Logon)
X1$RMTNM	BSC or SNA Remote Name	8-byte Remote Name. Set for BSC only if BSC Signon is used.
X1$UMSG@	User Message Address	Address Security violation message. This value is valid only if a return code of X‘08’ is returned in X1$ACODE.
X1$UMSGL	User Message Length	Halfword value of the length in hex of the Substitute Security violation message specified in X1$UMSG@
X1$LNTYP	Line Type	1-byte value indicating the line type: X‘01’ = BSC Switched X‘02’ = BSC Non-switched X‘03’ = SNA LU Type 1
X1$INIT@	Initialization exit word	Fullword containing the address or information returned from the Initialization exit. This value is all X‘00’ if no Initialization exit is used.

DSECT Label	Contains Address of	Data Field Contents
X1$ONEB	ONEBATCH Value	1-byte value indicating the use of ONEBATCH with $$REQUEST. C'Y' = ONEBATCH=YES C'N' = ONEBATCH=NO
X1$LPASS	Logon Password	8-byte Logon Password to be used by the security interface.
X1$NPASS	New Logon Password	8-byte New Logon Password to be used by the security interface.
X1$SFLAG	Logon Successfully Completed Flag	1 hex byte flag indicating if logon and batch/function security processing has been successfully completed. Could also be set on to skip normal logon or batch/function processing that is done by the security interface. ◆ XX$LGNOK X'01' Bypass MBXSECURE=LOGON Validation ◆ XX$BCHOK X'02' Bypass MBXSECURE=BATCH Validation
X1$RBUFA	Input Buffer Address (ADD only)	Fullword containing the address of the input buffer.
X1RBUFS	Input Buffer Size (ADD only)	2-byte hex pointing to length of the input buffer.
X1$VBQNO	VBQ # for ADD	2-byte hex number (1–20) of the VBQ to use.

Security exit one can change the following fields pointed to by the parameter list:

- X1$ACODE
- X1$APLID (SNA only)
- X1$BCHID
- X1$BLOCK (BSC only)
- X1$CMP (BSC only)
- X1$ID
- X1$SFLAG
- X1$LPASS
- X1$MEDIA (SNA only)
- X1$MULTX
- X1$NPASS
- X1$ONEB
- X1$PASSW
- X1$RCSEP (BSC only)

- X1$RMTNM (SNA or BSC if BSC Signon is used)
- X1$TRUNC (BSC only)
- X1$UMSG@
- X1$UMSGL
- X1$VBQNO
- X1$WORKA
- X1$XMIT

Security Exit One Requirements

The following requirements apply to Security exit one. You must follow these rules for proper exit use.

- Security exit one is called when Connect:Enterprise detects a $$ command in an input buffer from a remote site, when a new data collection is beginning for a remote site that does not supply a $$ADD at the front of the data, or when an SNA LOGON or BSC Signon is attempted.
- Security exit one is not called for $$ commands from the system console. It is also not called for host-initiated transmissions to a remote site during an Auto Connect, since these transmissions are initiated due to a request at the host site. The exit is called for data collections from the remote site during an Auto Connect.
- Some data fields pointed to by the parameter list are not set for all transactions. For example, $$REQUEST does not specify a MULTXMIT parameter, so do not use the MULTXMIT data field.
- If Security exit one is called for a data collection without a $$ADD ($$ Command Type = C‘2’), the data fields are set to the standard system defaults for a missing $$ADD. They are as follows:

```
ID=the Remote Name (if BSC without SIGNON, the Line ID)
 BATCHID='BATCH WITHOUT $$ADD'
 XMIT=N
```

- Use this exit to change these default values. You cannot change the ID to blanks. If you do, Connect:Enterprise uses the remote name as the Mailbox ID.
- If action code X‘08’ is set to force a security violation and you also supply a Security exit two, then Security exit two is later called by Connect:Enterprise. If action code X‘08’ is set and X1$UMSG@ is nonzero, it is assumed that a substitute security violation message is used as specified by X1$UMSG@ and X1$UMSGL.

Sample Security User Exit STSEC1

The STSEC1 sample user exit program demonstrates how to do the following:

- Access the user work area
- Bypass all Connect:Enterprise security checks
- Force security violations

- Change values set by Connect:Enterprise for a Mailbox ID, BATCHID, and so forth
- Provide a second level of security (such as an operator security code) for $$ADD and $$REQUEST transactions. When this exit is installed, a remote site must supply the proper 4-digit security code for its Mailbox ID in the first four characters of the 64-byte nongeneric BATCHID field.
- Provide your own security violation messages

STSEC1 Program Logic

The STSEC1 program performs the following processing:

- For the first entry only, the STSEC1 program builds a security table in the user work area from a //SECURITY DD file supplied in the JCL for Connect:Enterprise. Data in the //SECURITY DD file may resemble the following:

```
//SECURITY DD *
OPER01  0432
OPER02  7856
MARYJANE3887
```

- If the Mailbox ID=SECRET where SECRET is a special ID that is allowed unlimited system access, the STSEC1 program sets the action code to X'04' to bypass all security checks and returns to Connect:Enterprise.
- If the command is a $$DELETE, the command is not allowed. The STSEC1 program sets the action code to X'08' to process as a security violation and returns to Connect:Enterprise.
- If the command is a $$DIRECTORY, SNA LOGON or BSC SIGNON, the STSEC1 program uses standard Connect:Enterprise processing, leaves the action code set to the default value of X'00', and returns to Connect:Enterprise.
- If the transaction is a data collection that is missing a $$ADD record, the program changes the system default values to the following:
  - ID=NOADD
  - BATCHID='0000 NO $$ADD'
  - XMIT=Y
- For standard $$ADD and $$REQUEST transactions, the 4-digit security code supplied in the BATCHID parameter must now be validated. The remote site must specify the security code in the first four digits of the nongeneric BATCHID parameter.
- First looks up the Mailbox ID value in the security table in the user work area. If the ID is not in the table, overlays the security code with 9999, sets the action code to X'08' for a security violation, and returns to Connect:Enterprise.
- If the ID is valid, checks the first four digits of the nongeneric BATCHID against the security table in the user work area. If the security code is not in the table, overlays the security code with 9999, sets the action code to X'08' for a security violation, provides a substitute security violation message, and returns to Connect:Enterprise.
- If the security code is valid, overlays it with 0000. The security code is overlaid so it will not appear in any output, after which it would no longer be secure.

- If the transaction is a $$ADD, leaves the action code set to X'00' and returns to Connect:Enterprise.
- If the transaction is a $$REQUEST, provides for the capability to request all batches for a Mailbox ID (BATCHID set to blanks) or to request batches by specific batch number.
- If BATCHID='CCCC #NNNN' (4-digit security code followed by # and the 4-digit batch number), changes the BATCHID to X'00' followed by the batch number (the format expected by the Connect:Enterprise system).
- If BATCHID='CCCC ALL' (4-digit security code followed by the word ALL), changes the BATCHID to blanks. This format is expected by the Connect:Enterprise system.
- Leaves the action code set to X'00' and returns to Connect:Enterprise.

Implementing STSEC1

To run STSEC1 with online Connect:Enterprise, follow these instructions:

1. Use an ODF that contains the following:

```
*OPTIONS
   ...
   XSECUR1=STSEC1
```

2. Provide a security file in the online Connect:Enterprise execution JCL. The records should contain all valid 8-byte Mailbox IDs followed by the assigned 4-digit security code. For example:

```
//SECURITY DD *
OPER01  0432
OPER02  7856
MARYJANE3887
```

3. Test a variety of $$ commands, such as the following:

```
OK:    $$ADD ID=MARYJANE BATCHID='3887 PAYROLL DATA'
...
...
OK:    $$REQUEST ID=MARYJANE BATCHID='3887 ALL'
OK:    $$REQUEST ID=MARYJANE BATCHID='3887 PAYROLL DATA'
FAIL:  $$REQUEST ID=MARYJANE BATCHID='0432 ALL'
FAIL:  $$REQUEST ID=OPER01 BATCHID=#0052
FAIL:  $$REQUEST ID=OPER02
OK:    $$DIRECTORY ID=TEST
FAIL:  $$DELETE ID=OPER01 BATCHID=#0001
```

Using Security Exit Two

Security exit two is called after Connect:Enterprise detects a security violation during its optional security processing. The reason for the violation is passed to the exit program.

This exit can set an action code that requests Connect:Enterprise to continue as normally and process the violation, or to ignore the violation and allow the transaction.

Security Exit Two Parameters

Parameters passed to Security exit two are addressed by the X2$DSECT in the M$XPARM macro. A listing of the parameters and the contents of the data field pointed to by the parameters is in the following table:

DSECT Label	Contains Address of	Data Field Contents
X2$XTYPE	Exit Type Code	1-byte exit type code C‘2’ = Security exit two
X2$ACODE	Area to return Action Code	1-byte exit action code X‘00’ = Continue as normally X‘04’ = Ignore Security Violation
X2$WORKA	User Work Area	256-byte user work area
X2$LINID	Remote Name or BSC Line ID	8-byte Remote Name for which the transaction is in progress. If the Remote Name is not available for BSC, the Line ID is used.
X2$CTYPE	$$ Command Type	1-byte $$ Command Type C‘1’ = $$ADD C‘2’ = Missing $$ADD C‘3’ = $$REQUEST C‘4’ = $$DIRECTORY C‘5’ = $$DELETE C‘6’ = $$LOGOFF C‘7’ = SNA LOGON C‘8’ = BSC LOGON
X2$ID	Mailbox ID	8-byte Mailbox ID used
X2$BCHID	BATCHID	64-byte user BATCHID If BATCHID=‘xxx...xxx’, this field contains the user batch ID without the quotes. If BATCHID=#nnnnnnn, this field contains X‘00’ followed by the 7-digit packed decimal batch number. If BATCHID parameter was invalid, this field is set to X‘FF’ followed by blanks.
X2$PASSW	Password	8-byte Password entered

DSECT Label	Contains Address of	Data Field Contents
X2$SVCOD	Security Violation Codes	Four 1-byte value Security Violation Codes. (The proper 1-byte code is set to indicate which security violation occurred.) Byte 1 = C‘1’ ID validation failed Byte 2 = C‘1’ Password incorrect Byte 3 = C‘X’ SNA LOGON security error Byte 4 = C‘1’ Security exit one requested error
X2$LNTYP	Line Type	1-byte value indicating the line type: X‘01’ = BSC Switched X‘02’ = BSC Non-switched X‘03’ = SNA LU Type 1
X2$INIT@	Initialization exit word	Fullword containing the address or information returned from the Initialization exit. This value is all X‘00’ if no Initialization exit is used.

Security exit two can change five fields pointed to by the parameter list. They are as follows:

- X2$ACODE
- X2$WORKA
- X2$ID
- X2$BCHID
- X2$PASSW

Security Exit Two Requirements

The following requirements apply to Security exit two. You must follow these rules for proper exit use.

- Security exit two is called after Connect:Enterprise detects a security violation. Four security violation codes explain the reason for the failure. The address of the four codes is passed in the parameter list. The four codes occupy contiguous storage, and the proper code is set to C‘1’ to indicate the reason for the security violation.
  - If Byte 1 is set to C‘1’, Connect:Enterprise Batch Security failed. The ODF specified SECURITY=BATCH, but the ID supplied by the remote site was not defined as valid in the *SECURITY records of the ODF.
  - (For BSC Only) If Byte 1 is set to C‘2’, BTAM ID verification failed. The ID exchange during a connection attempt on a switched line failed, and BTAM has disconnected the line.
  - If Byte 2 is set to C‘1’, the PASSWORD supplied in a $$DIRECTORY command without an ID was incorrect.

- (For SNA Only) If Byte 3 is set to any value other than C‘0’, an SNA LOGON attempt is being rejected by Connect:Enterprise. You cannot attempt to override an SNA LOGON reject. (This byte should be ignored in BSC connections.)
- If Byte 4 is set to C‘1’, your Security exit one or the Connect:Enterprise security interface requested Connect:Enterprise to treat this transaction as a security violation by setting an action code = X‘08’. You can set an action code = X‘00’ to agree that the transaction is a security violation, or you can override Security exit one or the security interface by setting an action code = X‘04’ to allow the transaction.

✦ Some data fields pointed to by the parameter list are not set for all transactions. For example, $$REQUEST can be entered with only an ID parameter, so the BATCHID can be set to blanks.

Sample Security User Exit STSEC2

The STSEC2 sample user exit program demonstrates how to do the following:

✦ Override a security violation detected by Connect:Enterprise

✦ Produce a report of security violations

STSEC2 Program Logic

The STSEC2 program performs the following processing:

✦ For the first entry only, opens an output security report file and prints the security report headers.

✦ If the ID=SECRET, sets the action code to X’04’ to override the security violation and allow the transaction, and returns to Connect:Enterprise.

✦ For all other errors, formats a detail line on the security report that displays the reason for the failure, the $$ Command Type, ID, BATCHID, password, and line ID.

✦ Leaves the action code set to X’00’ and returns to Connect:Enterprise.

Implementing STSEC2

To run STSEC2 with online Connect:Enterprise, follow these instructions:

1. Use an ODF that contains the following:

```
*OPTIONS
   ...
   PASSWORD=BANANA
   ...
   XSECUR2=STSEC2
   XSECUR1=STSEC1 (optional)
   SECURITY=BATCH
   MBXSECURE=ALL (optional)
*SECURITY
   ID=OPER01 ID=OPER02 ID=MARYJANE ID=SECRET
```

2. Provide a DD statement for the output security report such as:

```
//SECREPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=133)
```

3. Test a variety of $$ commands that cause security errors. For example, use IDs that are not defined in your *SECURITY records, use invalid passwords on $$DIRECTORY, or test with the supplied STSEC1 user exit and cause some failures.

```
OK:    $$REQUEST ID=MARYJANE BATCHID='3887 ALL'
FAIL:  $$REQUEST ID=MARY BATCHID='3887 ALL'
OK:    $$DIRECTORY PASSWORD=BANANA
FAIL:  $$DIRECTORY PASSWORD=BANANAS
FAIL:  $$DIRECTORY ID=BADID
```

Using the Output Exit

The Output exit is called before every text write when transmitting data to a remote site. The Output exit can examine and change data in the output buffer, as long as the maximum buffer size is not exceeded. If data is changed, you must still retain the proper line and record control characters. The Output exit can set an action code that requests Connect:Enterprise to continue as normally, to use the data as changed by the exit, or to ignore the data and bypass the write to the remote site.

Output Exit Parameters

Parameters passed to the Output exit are addressed through the X4$DSECT in the M$XPARM macro. A listing of the parameters and the contents of the data field pointed to by the parameters is in the following table:

DSECT Label	Contains Address of	Data Field Contents
X4$XTYPE	Exit Type Code	1-byte Exit Type Code C'4' = Output exit
X4$ACODE	Area to return Action Code	1-byte exit action code X'00' = Continue as normal X'04' = Output Data Changed X'08' = Ignore Output Block
X4$WORKA	User Work Area	256-byte user work area
X4$LINID	Remote Name or BSC Line ID	8-byte Remote Name for which the transaction is in progress. If the Remote Name is not available, the Line ID is used.
X4$OFLAG	Output Data Flag Byte	1-byte data flags X'80' = SNA SCBs are present X'40' = SNA Data contains an FMH

DSECT Label	Contains Address of	Data Field Contents
X4$OMAXL	Maximum Length	2-byte Maximum Length of Output Area
X4$OUTPL	Length of Output Data	2-byte Length of Output Data, including control characters (hexadecimal)
X4$OUTPA	Output Area	Output Data to be Sent to Remote
X4$LNTYP	Line Type	1-byte value indicating the line type: X‘01’ = BSC Switched X‘02’ = BSC Non-switched X‘03’ = SNA LU Type 1
X4$CTYPE	Command Type	1-byte area indicating which command is currently being processed from the requested output type commands: C‘1’ = $$ADD C‘2’ = Missing $$ADD C‘3’ = $$REQUEST C‘4’ = $$DIRECTORY C‘5’ = $$DELETE C‘6’ = $$LOGOFF C‘7’ = SNA LOGON C‘8’ = BSC LOGON
X4$INIT@	Initialization exit word	Fullword containing the address or information returned from the Initialization exit. This value is all X‘00’ if no Initialization exit is used.

The Output exit can change four fields pointed to by the parameter list. They are as follows:

- X4$ACODE
- X4$WORKA
- X4$OUTPL
- X4$OUTPA

Output Exit Requirements

The following requirements apply to the Output exit. You must follow these rules for proper exit use.

- The purpose of the Output exit is to allow normal text message examination and modification. Do not attempt to use this exit to do your own I/O.
- The Output exit is called for standard text output only. The exit is not called for nontext output, such as WRITE TD (disconnect in BSC connections) or CLSDST (end session in SNA connections).

If you want to change the output data there are several rules which you must follow:

- If you change the length of the output data, you must set new data length in the X4$OUTPL parameter. The length must never exceed the value pointed to by X4$OMAXL.
- You are passed the actual Connect:Enterprise output buffer. Be careful when modifying data that you do not overlay other areas and cause a system ABEND.
- You must retain all SNA control characters, such as IRS, and SCBs.
- You must retain all BSC line control, such as STX, ETX, and IRS.
- You must not use return code X'08' to ignore critical writes, such as those that affect chaining or brackets or contain an ETX.

Sample Output User Exit STOUT

The STOUT sample user exit program demonstrates how to do the following:

- Examine and change data in the output buffer before Connect:Enterprise sends it to a remote site
- Ignore blocks that contain a certain text header.

STOUT Program Logic

The STOUT program performs the following processing:

- If the output buffer contains SNA compressed records, sets the action code to X'00' and sends the block unchanged.
- If the output buffer contains the word IGNORE starting anywhere in the first 10 bytes, sets the action code to X'08' to ignore the block and returns to Connect:Enterprise.
- If the output buffer contains the word NOCHG starting anywhere in the first 10 bytes, leaves the action code set to X'00' to send the block unchanged and returns to Connect:Enterprise.
- Otherwise, moves a 4-character standard field (the word OUT followed by a record separator) to the front of the output block, sets the action code to X'04' to send the changed block, and returns to Connect:Enterprise.

Implementing STOUT

To run STOUT with online Connect:Enterprise, follow these instructions:

1. Use an ODF that contains the following:

```
*OPTIONS
   ...
   XOUTPUT=STOUT
```

2. Create a batch for transmission containing the three types of special records and use the offline utilities to ADD it to the VSAM batch files. The batch could contain:

```
1 - THIS SHOULD BE PRECEDED BY AN OUT RECORD
2 - THIS SHOULD BE PRECEDED BY AN OUT RECORD
3 - IGNORE .  .  .  SHOULD NOT BE SENT
4 - NOCHG .  .  .  SHOULD NOT BE PRECEDED BY OUT
5 - IGNORE .  .  .  SHOULD NOT BE SENT
6 - THIS SHOULD BE PRECEDED BY AN OUT RECORD
```

3. $$REQUEST the added batch and examine the data sent to the remote site.

Using the End of Batch Exit

The End of Batch exit is called at the successful completion of an online batch collection. The exit can examine the Mailbox ID and BATCHID and submit an EXTRACT job or, if desired, begin an Auto Connect list. The exit can also invoke wake-up transaction processing if the CICS interface has been implemented. The End of Batch exit is entered only once per batch, that is, at the successful completion of the batch being added to Connect:Enterprise.

The purpose of the End of Batch exit is to give you control when a batch has been added successfully to Connect:Enterprise. The exit enables the user to extract or to automatically begin an Auto Connect or to do other processing as soon as a batch is received.

Note: Depending on your End of Batch processing requirements, you can use the End of Batch application agent as an alternative to coding and maintaining an exit program. Refer to *Chapter 1, Overview of Connect:Enterprise Application Agents,* for complete details.

Connect:Enterprise remains compatible with nonreentrant type programs, including any user-written exits and earlier versions of STEOBX and STEOBX2 assuming you have not implemented any of the FTP components. However, Connect:Enterprise version 1.x and later uses reentrant programming standards for those customers using FTP. The reentrant standards are enforced by the ODF parameter XEOBVER=2. Specific requirements are described in the following sections. The sample exits, starting with Connect:Enterprise version 1.x, conform to these standards, and any user-written exits must also conform if you are using FTP.

End of Batch Exit Parameters

Parameters passed to the End of Batch exits are addressed by Register 1. A listing of the parameters and contents of the data field pointed to by the parameters is in the following table:

R1 Points to Parameter List	Contains Address of	Data Field Contents
+0(4)	Exit type code	1-byte exit type code C’5’ = End of Batch exit
+4(4)	Area to return action code	1-byte exit action code X’00’ = Continue as normal C’W’ = Initiate CICS wake-up transaction X’08’ = Program failed initialization during Start processing (ABENDU0253)
+8(4)	Batch Control record	4-byte address of the Batch Control record (M$BCREC Macro)

R1 Points to Parameter List	Contains Address of	Data Field Contents
+C(4)	Initialization Word	4-byte address of the initialization exit word
+10(4)	Shared Storage Address	4-byte address of version 2 exit shared storage area address

The End of Batch exit can change the exit action code.

Nonreentrant End of Batch Exit Requirements

The following requirements apply to the End of Batch exits. You must follow these rules for proper exit use.

- Do not modify any of the fields in the BC$RECRD DSECT. They are intended as read only.
- To start an Auto Connect by modify commands, you must specify MODIFY=YES in Connect:Enterprise in the *OPTIONS section of the ODF and set the Auto Connect to start in the console.
- When using STEOBX2, you can write an End of Batch exit program to read a partitioned data set and select different members to be sent to the INTRDR based on the SUBTABLE. The SUBTABLE can also be a member of the PDS. This process requires you to add two data sets to the Connect:Enterprise JCL and enables you to make changes by updating only the PDS.
- Do not implement any FTP component.

Sample End of Batch User Exits STEOBX, STEOBX2, and STEOBX2V

Three sample End of Batch exits are provided in Connect:Enterprise:

- STEOBX
- STEOBX2
- STEOBX2V

All three sample user programs demonstrate how to do the following.

- Address the BC$RECRD DSECT and examine the fields. They contain information about the batch that was just received: Mailbox ID, batch number, BATCHID, number of blocks, and so on.
- Set up a table that determines if, when a batch is added, it is also extracted.
- Set up a physical sequential data set and a JES2 INTRDR data set.
- Write JCL or operator commands to the JES2 INTRDR.

STEOBX Program Logic

The program contains a user-tailored action table that is defined and assembled in the exit program source and is written as one of four sequential data sets to submit a user-written JCL stream for post-batch receive processing. Only four user JCL files (STJCL01, STJCL02, STJCL03, and STJCL04) are read for submission to the operating system to execute in the sample STEOBX.

The STEOBX program performs the following processing:

- Checks the SUBTABLE TYPE to see if the system compares ID only, BATCHID only, or ID and BATCHID.
- Performs the appropriate compare against the batch that has just completed to determine what action, if any, is to be done.
- If a JOB is to be submitted or an Auto Connect list is to be started, the program opens the proper JCL file, opens a JES2 INTRDR, and writes the JCL and/or MODIFY commands to JES2.

STEOBX2 Program Logic

The program reads the STPDS data set member SUBTABLE and creates the user-determined action table dynamically and holds it in memory for the duration of that instance of Connect:Enterprise. This capability permits you to update only the SUBTABLS PDS member between uses as your work load changes, rather than requiring you to reprogram the tables defined for the STEOBX source. Unlike STEOBX, which uses only four standard user JCL files, STEOBX2 retrieves the user-written JCL from members stored in the STPDS data set. You can define a JCL for a specific set of naming attributes (ID, ID+BATCHID, or BATCHID), limited only by the size of the data set.

STEOBX2V Program Logic

The program retrieves SUBTABLE-like look-up information from a VSAM data set and performs the user-determined action in a more dynamic manner than the STEOBX2 sample SOURCE exit program. When the VSAM data set is defined with SHAREOPTIONS(4), you can add or modify entries in the SUBTABLE dynamically; that is, they are recognized immediately without restarting the product. You can also use some third-party software packages to delete obsolete records from the VSAM data sets. After you update records or delete obsolete records in VSAM files, perform regular file maintenance to reclaim file space. See Connect:Enterprise for z/OS Administration Guide for more information on maintaining VSAM files.

The STEOBX2V sample exit conforms to all other STEOBX2 programming specifications and uses only the reentrant version of exit processing due to the popularity of the FTP protocol. You can define the VSAM KSDS key and record layout to meet your system requirements. However, the sample SOURCE program uses a 72-byte record key consisting of the 8-byte Mailbox ID and the 64-byte User BATCHID values followed by an 8-byte PDS member name. This record key is used to fetch the EXTRACT job or Auto Connect Modify command syntax to be retrieved from the STPDS data set and submitted to the operating system by the STINTRDR reader file.

Implementing Nonreentrant End of Batch Exits

To run End of Batch exits with online Connect:Enterprise:

1. Use an ODF that contains:

```
*OPTIONS

...
XEOBVER=1
XENDOFB=exit name
```

where XEOBVER=1 indicates nonreentrant standards are in place, and where *exit name* is STEOBX, STEOBX2, or the name of your user-written exit.

2. Successfully add batches online and verify the correct actions were taken.

Reentrant End of Batch Exit Requirements

In addition to the general exit requirements described previously, the reentrant (version 2) programming standard also requires you to adhere to the following requirements:

- In the *OPTIONS section of the ODF (DD=OPTDEF), set the XEOBVER= parameter to a value of 2 to indicate reentrant programming standards have been coded.
- The product initialization call that allocates and prepares the shared storage work area is mapped by the SOURCE data set macro members, S$EOBX, S$EOBX2, or S$EOBX2V. Use the mapping macro appropriate to the source program you are using, STEOBX, STEOBX2, or STEOBX2V.

  The pointer to this shared storage is passed as a parameter located at offset 16x'10' from entry Register 1. This parameter points to the address established by the initialization call to the EOB exit which contains any data control block (DCB) areas for files to be shared (STINTRDR, STPDS). It may also include the allocation for the in-storage SUBTABLE which directs the processing flow during the actual EOB event calls, and prevents repeat loading of this control table for each invocation of an exit.

  Note: Setting the return action code to x'08' during this call results in initialization termination indicating that the exit requirements were not satisfied by the current configuration.

- No batch control record address will be passed during this call because it occurs during initialization and is not a legitimate End of Batch event.
- To conform to reentrant standards, the programs cannot modify themselves during use. Implementing this restriction requires you to use the MF=L and MF=E type features on many standard operating system macros used by your exit program. Sample macros include OPEN/CLOSE, WTO, MGCR, and ENQ/DEQ.
- The program must serialize use of the shared storage areas to protect areas such as the DCB files. The sample SOURCE programs use a specific resource name combination where QNAME=CL8'*MAILBOX*' and the RNAME=CL8'*EOB EXIT*.' The scope of this resource is EXCLUSIVE, STEP to prevent bottlenecks resulting from users executing multiple copies of Connect:Enterprise. This is necessary because the program is coded to handle all EOB events concurrently, regardless of origin, and it must support calls from both the C:E STMAIN task and any number of concurrent FTP thread tasks.

Implementing Reentrant End of Batch Exits

To run an End of Batch exit with online Connect:Enterprise using the version 2 standards:

1. Use an ODF that contains:

```
*OPTIONS

...
XEOBVER=2
XENDOFB=exit name
```

where XEOBVER=2 indicates reentrant standards are in place, and where *exit name* is STEOBX, STEOBX2, STEOBX2V, or the name of your user-written exit.

2. Successfully add batches online and verify the correct actions were taken.

Using the Initialization Exit

The Initialization exit is called prior to Connect:Enterprise system initialization. The exit is passed a 1-word (4-byte) area in which it can store information. This area is subsequently passed to all other online exits to allow them to interrogate information that can be stored there.

Initialization Exit Parameter

The parameter passed to the Initialization exit is a fullword of storage (4 bytes aligned to a fullword) pointed to by register 1 on entry. This field is available to all online exits.

Sample Initialization User Exit STXINIT

The STXINIT sample user exit program demonstrates how to do the following:

- Perform global system initialization routines
- Pass appropriate information to other online exits

The STXINIT program performs the following processing:

- Dynamically allocates and opens the JES2 INTRDR
- Retains the DCB address in the Initialization exit word for subsequent use by the End Of Batch exit and the Termination exit

Using the Termination Exit

The Termination exit is called prior to Connect:Enterprise system shutdown. It is passed the same 1-word (4-byte) area returned by the Initialization exit.

Termination Exit Parameter

The parameter passed to the Termination exit is a fullword of storage (4 bytes aligned to a fullword) pointed to by register 1 on entry. This field is available to all other online exits, and is the same field given to the Initialization exit.

Sample Termination User Exit STTERM

The STTERM sample user exit program demonstrates how to do the following:

- Perform global system termination routines
- Perform system level cleanup

The STTERM program performs the following processing:

- Dynamically deallocates and closes the JES2 INTRDR
- Returns control to Connect:Enterprise for system termination

Using the Log Exit

The Log exit is called before each log record is written or updated on the VSAM log file. The exit is given control for both Auto Connect records (complete or queued) and remote connect records.

The Log exit can examine and change data in the log records, but cannot alter the log record key. The data must retain its standard format for the offline utilities REPORT function to process properly. The length of the log record cannot be changed.

Note: Depending on your log processing requirements, you can use the Logging application agent as an alternative to coding and maintaining an exit program. Refer to Chapter 1, *Overview of Connect:Enterprise Application Agents* for complete details.

Log Exit Parameters

The following macros are provided on the installation tape in the MAILBOX.SOURCE library:

- M$ACREC–Contains the DSECT for the Auto Connect record.
- M$DCREC–Contains the DSECT for the Auto Connect Queue record.
- M$LOGBQ–Contains the DSECT for the remote connect record.

The parameters passed to the Log exit are pointed to by register 1 and are described in the following table:

R1 Points to Parameter List	Parameter Contents	Data Field Contents
+0(4)	Address of Exit Type Field	1-byte exit type code C‘6’ = Log exit
+4(4)	Address of Return Code Byte	18-byte Return Code X‘00’ = Process as normal
+8(4)	Address of Log Function Type	1-byte Log Function Type C‘1’ = Put New Log record C‘2’ =Update Log record
+C(4)	Address of log record key	18-byte log record key 1st byte C‘A’ = Auto Connect 1st byte C‘B’ = remote connect 1st byte C‘D’ = Auto Connect queue
+10(4)	Address of log record data	Refer to the M$ACREC macro for the Auto Connect DSECT. Refer to the M$LOGB macro for the remote connect DSECT. Refer to the M$DCREC macro for the queued Auto Connect DSECT.
+14(4)	Address of Halfword Log Record Data Length	2-byte length of data portion of the log record
+18(4)	Address of Initialization exit word	A fullword containing the address of information returned from the Initialization exit. This value is binary zero if no Initialization exit is used.

The Log exit can change the Log record data.

Log Exit Requirements

The Log exit is called for Auto Connects and remote connects and for both summary and detail records. The Log exit is not called for the GET function, which is used only for positioning before a PUT UPDATE is done, or for accessing the Auto Connect master record.

The use of logging in Connect:Enterprise varies depending on the type of transaction being processed:

- Auto Connect
- Queued Auto Connect
- Remote Connect, and
- CICS API ADD and REQUEST functions.

Auto Connect Logging

Auto Connects write a single summary record for the entire Auto Connect, regardless of the number of remote sites contacted during the Auto Connect. A detail record is written for each batch sent or received, or for each error condition that occurs during the Auto Connect. When the Auto Connect ends, the summary record is updated with statistics for the completed Auto Connect.

For example, an Auto Connect that sends two batches to a remote site gives control to the Log exit as follows:

1. PUT NEW summary record (failure code 001 is set in case the Auto Connect does not complete).
2. PUT NEW detail record for batch number 1.
3. PUT NEW detail record for batch number 2.
4. PUT UPDATE summary record (failure code 001 is reset).

Queued Auto Connect Logging

When an Auto Connect is queued, a single log record is written. When the Auto Connect is reactivated, the log record is updated to reflect the reactivation status, date, and time. When a queued Auto Connect is deleted, the log record is updated with delete status, date and time.

If a reactivated Auto Connect still cannot establish a session with one remote, it is queued again. In this case, a new log record is created.

For example, an Auto Connect is queued and later reactivated, the log record receives control as follows:

1. PUT NEW Auto Connect queue record.
2. PUT UPDATE Auto Connect queue record.

Remote Connect Logging

Remote connects write a single summary record for a single connection from a remote site. A detail record is written, and sometimes updated, for each individual function used during the connection. When the connection ends, the summary record is updated with statistics for the connection.

Detail records are written for the short duration functions (CONNECT and DISCONNECT). Detail records are written and later updated for the long duration functions:

- ADD
- REQUEST
- DIRECTORY
- DELETE
- SIGNON

For example, a remote connect that adds one batch and requests one batch would give control to the Log exit as follows:

1. PUT NEW summary record.
2. PUT NEW detail record for CONNECT.

3. PUT NEW detail record for batch ADD.
4. PUT UPDATE detail record for batch ADD.
5. PUT NEW detail record for REQUEST.
6. PUT UPDATE detail record for REQUEST.
7. PUT NEW detail record for DISCONNECT.
8. PUT UPDATE summary record.

Connect:Enterprise CICS API ADD and REQUEST Logging

Logging for the CICS API transactions for ADD and REQUEST batches uses only detail records. Summary records are not required because each transaction is stand alone, rather than part of a connection process.

Each batch ADD and each batch REQUEST gives control to the Log exit for a PUT NEW detail record.

Sample Log User Exit STLOGX

The STLOGX sample user exit program demonstrates how to do the following:

- Differentiate between Auto Connect and Remote Connect log records
- Examine and change data in the log records before Connect:Enterprise writes the data to the VSAM log file
- How to display a console message for some log record failure codes.

STLOGX Program Logic

This program performs the following processing:

- If the log record key begins with C‘A’ for auto connect:
  - Checks for a failure code 011 (X ’0B’—no batches for transmission) and changes it to zero to suppress an error for this condition.
  - Checks for a failure code 001 (X’01’) on an Auto Connect summary record. This code is not an error, because this failure code is reset when the Auto Connect later ends.
  - For all other failure codes, displays a message on the system console that contains the failure code.
- If the log record key begins with C‘B’ for remote connect:
  - Checks for a failure code 011 (X’0B’—no batches for transmission) and changes it to zero to suppress an error for this condition.
  - Checks for a failure code 001 (X’01’) on any remote connect record. This code is not an error, because this failure code is reset when the function later ends.
  - For all other failure codes, displays a message on the system console which contains the failure code.

Implementing STLOGX

To run STLOGX with online Connect:Enterprise:

1. Use an ODF that contains the following:

```
*OPTIONS
  ...
  XLOG=STLOGX
```

2. Create some log records with failure codes to see the console message displayed. For a remote connect, try a $$DIRECTORY with an invalid password. For an Auto Connect, try to establish a connection with a remote site that is unavailable.

Using the APPC Security Exit

The APPC Security exit is called for every new transaction received from the CICS or ISPF interface. Transactions that are not new do not invoke this exit, such as an ADD or REQUEST that is in progress. There is no way for transactions in progress to circumvent security by switching functions midstream.

When called, the APPC Security exit is passed the address of the Interface Parameter Structure (IPS). The IPS includes a GDS header, an IPS header, and usually an IPS trailer. The IPS header contains the REQUEST CODE to identify the transaction type. A list of all valid request codes is in the table REQTAB in the sample security exit (STCSEC) supplied during installation.

The IPS trailer immediately follows the IPS header for most transactions. Only the VERIFY transaction, which provides the capability to verify the user ID and password, does not contain an IPS trailer.

Note: If you use the APPC security exit, you must modify the Interface Paramaeter Structure (IPS) macro that it calls. For more information, see *IPS Trailer Portion Data* on page 213 and the upgrading section in the *Connect:Enterprise for z/OS Release Notes*.

APPC Security Exit Parameters

Register 1 points to a parameter list upon entry to the user-supplied APPC Security exit program. The parameter list contains 7 fullwords as shown in the following table:

R1 Points to Parameter List	Parameter Contents	Data Field Contents
+0(4)	Address of exit type code	C’C’=Security exit

R1 Points to Parameter List	Parameter Contents	Data Field Contents
+4(4)	Address of area to return action code	X‘00’ ... Security Check OK X‘04’ ... Error, invalid password X‘08’ ... Error, user-defined error code is set in H00RTNCD
+8(4)	Address of IPS	IPS GDS Header (A$GDS DSECT)
+C(4)	Address of IPS header	IPS Header (C$H00 DSECT)
+10(4)	Address of IPS trailer	IPS Trailer (DSECT macro name matches the request code in H00REQCD, binary zeros if H00REQCD=VERIFY)
+14(4)	Address of Initialization exit word	A fullword containing the address of information returned from the Initialization exit. This value is binary zero if no Initialization exit is used.
+18(4)	Address of Logon/Batch/Function Security Flag	Points to a flag that indicates if logon and request processing has been successfully completed. This flag could also be set on to skip normal security interface processing.

APPC Security Exit Requirements

The following requirements apply to the APPC Security exit. You must follow these rules for proper exit use.

- Use the APPC Security exit to validate the user ID and password in the VERIFY transaction, or to examine or alter data in the IPS.
- Do not change the size of the IPS.
- The Connect:Enterprise APPC password is encrypted for security reasons. It is decrypted immediately before passing control to the security exit, and re-encrypted upon return.
- The security exit can set an action code to indicate the success or failure of the security checks.
- Modify *only* the password and new password fields in the IPS header.

Sample APPC Security User Exit STCSEC

The STCSEC sample user exit program demonstrates how to do the following:

- Validate a user and password from a CICS or ISPF interface transaction
- Validate that a user is authorized to perform any CICS or ISPF interface transaction
- Limit a batch browse to selected batches

STCSEC Program Logic

The STCSEC program performs the following processing:

- ✦ Builds a security table of user IDs and passwords from an input file.
- ✦ Refreshes the security table when a specific user ID/password is entered.
- ✦ Validates the user ID/password passed in the verify transaction, and sets return codes for user ID or password failure.
- ✦ Validates the request code in the IPS header.
- ✦ Checks that the user ID is authorized to use a request code. This example assumes an 8-byte user ID with the last byte set to a numeric authorization code.
- ✦ Checks the IPS trailer for the C$U22 request code (batch file data request) to determine if the batch can be browsed. This example only allows batches that begin with D to be browsed.

Implementing STCSEC

To run STCSEC with online Connect:Enterprise, follow these instructions:

1. Use an ODF that contains:

```
*OPTIONS
  ...
  XAPPCSEC=STCSEC
```

2. Perform various CICS interface transactions and verify the correct actions are being taken. For example, a batch that begins with an X cannot be browsed and a batch beginning with D can be browsed.
3. If you are using the provided sample exit, a CSECIN DD statement is needed for the security PDS.

Using the CICS Wake Up Initiate Exit

The Wake Up Initiate exit is called when a Connect:Enterprise End Of Batch exit sets a return code W to request that a wake up transaction be initiated and sent to a CICS API. Thus, when an online ADD completes, a CICS program can be notified of the event. The Wake Up Initiate exit is also driven each time a batch is successfully added by the APPC LU6.2 Interface.

Note: If you use the CICS Wake Up Initiate exit, you must modify the Interface Parameter Structure (IPS) macro that it calls. For more information, see *IPS Trailer Portion Data* on page 213 and the upgrading section in the *Connect:Enterprise for z/OS Release Notes*.

The Wake Up Initiate exit is passed an Interface Parameter Structure or IPS. The IPS includes a GDS header, an IPS header, and a trailer that is unique for the wake up transaction. The IPS header request code is set to C$W00, which is also the name of the macro that generates the DSECT for the wake up IPS trailer. The Wake Up Initiate exit can set an action code to send the IPS to the CICS API, to ignore the IPS, or it can set an error code.

Note: Depending on your Wake Up Initiate processing requirements, you can use the End Of Batch application agent with the WAKEUP instruction as an alternative to coding and maintaining an exit program. By invoking CICS wake up in this manner, you do not need either the End Of Batch exit or the APPC Wake Up Initiate exit to start a CICS wake up. Refer to Chapter 1, *Overview of Connect:Enterprise Application Agents* for complete details.

Wake Up Initiate Exit Parameters

Register 1 points to a parameter list upon entry to the user-supplied Wake Up Initiate exit program. The parameter list contains seven fullwords, as shown in the following table:

R1 Points to Parameter List	Parameter Contents	Data Field Contents
+0(4)	Address of exit type code	A=Wake Up Initiate exit
+4(4)	Address of area to return action code	X‘00’ ... Send Wake Up IPS to CICS X‘04’ ... Do not Wake Up CICS X‘08’ ... Unrecoverable error
+8(4)	Address of IPS	IPS GDS Header (A$GDS DSECT)
+C(4)	Address of IPS header	IPS Header (C$H00 DSECT)
+10(4)	Address of IPS trailer	IPS Trailer (C$W00 DSECT)
+14(4)	Address of Initialization exit word	A fullword containing the address of information returned from the Initialization exit. This value is binary zero if no Initialization exit is used.
+18(4)	Address of Wake Up reason code	C‘1’ Mailbox online ADD C‘2’ APPC online ADD

Wake Up Initiate Exit Requirements

Because the Wake Up Initiate exit is communicating with a user-supplied CICS API by sending it an IPS, some fields in the IPS header must be properly set to route the IPS. The following fields define the user transaction:

- H00CDEFN (CICS Resource Definitions)
- H00SYSID (CICS System ID)
- H00CRESC (CICS Resource Identifiers)

For complete details on the contents of these fields, refer to the IPS Header DSECT, generated by the macro C$H00.

The IPS trailer contains information for the batch that was added to Connect:Enterprise, such as the Mailbox ID, batch number, user batch ID, and number of blocks in the batch. This information can

be examined by the Wake Up Initiate exit and used to determine the processing requirements for the transaction. Refer to the IPS trailer DSECT, generated by the macro C$W00.

Sample Wake Up Initiate User Exit STCWI

The STCWI sample user exit program demonstrates how to do the following:

- Examine the Connect:Enterprise batch information to determine routing of the transaction
- Set installation-specific fields to send the wake up transaction to a CICS API. In this sample, the process to be executed is identified by user transaction ID, and is given control using a START with data.

STCWI Program Logic

The STCWI program performs the following processing:

- If Mailbox ID=NOWAKEUP, returns an action code of X'04', which does not wake up CICS.
- If Mailbox ID is not equal to NOWAKEUP and the batch is not added by a CICS API transaction, sets the action code to X'00' and wakes up CICS.

Implementing STCWI

To run STCWI with online Connect:Enterprise, follow these instructions:

1. Use an ODF that contains:

```
*OPTIONS
  ...
  XAPPCWI
```

2. Implement an End Of Batch exit which sets a return code of C'W'.
3. Define the correct CICS definitions within the program. Assemble and link the program.
   - H00CDEFN (CICS Resource Definitions)
   - H00SYSID (CICS System ID)
   - H00CRESC (CICS Resource Identifiers)
4. Add batches to Connect:Enterprise and verify that the CICS transaction is started for the correct batches.

Using the CICS Wake Up Terminate Exit

The Wake Up Terminate exit is called after the Wake Up Initiate transaction is delivered to the CICS API. Additionally, the exit is called with a return code set if the wake up transaction could not be delivered to the CICS API. The Wake Up Terminate exit is informed about the success or failure of the delivery of the message, but not about the results of the processing in the CICS API.

Note: If you use the CICS Wake Up Terminate exit, you must modify the Interface Parameter Structure (IPS) macro that it calls. For more information, see *IPS Trailer Portion Data* on page 213 and the upgrading section in the *Connect:Enterprise for z/OS Release Notes*.

- The Wake Up Terminate exit can provide logging or reporting on the results of the wake up delivery.
- The Wake Up Terminate exit is passed an IPS which includes a GDS header, an IPS header, and a trailer that is unique for the wake up transaction. The IPS header request code is set to C$W00, that is also the name of the macro which generates the DSECT for the wake up STIPS trailer.
- The Wake Up Terminate exit can set an action code to free the IPS, or it can set an error code.

Note: Depending on your Wake Up Terminate processing requirements, you can use the Wake Up Terminate application agent as an alternative to coding and maintaining an exit program. Refer to Chapter 1, *Overview of Connect:Enterprise Application Agents* for complete details.

Wake Up Terminate Exit Parameters

Register 1 points to a parameter list upon entry to the user-supplied Wake Up Terminate exit program. The parameter list contains six fullwords, as shown in the following table:

R1 Points to Parameter List	Parameter Contents	Data Field Contents
+0(4)	Address of exit type code	B=Wake Up Terminate exit
+4(4)	Address of area to return action code	X‘00’ ... Successful, free IPS X‘08’ ... Unrecoverable error
+8(4)	Address of IPS	IPS GDS Header (A$GDS DSECT)
+C(4)	Address of IPS header	IPS Header (C$H00 DSECT)
+10(4)	Address of IPS trailer	IPS Trailer Wake Up Transaction (C$W00 DSECT)
+14(4)	Address of Initialization exit word	A fullword containing the address of information returned from the Initialization exit. This value is binary zero if no Initialization exit is used.

Wake Up Terminate Exit Requirements

Because the Wake Up Terminate exit is communicating with a user-supplied CICS API, the setting and testing of return codes in the IPS is under their control. A return code can be set and tested in H00RTNCD in the IPS header, and values X’0400’ through X’0499’ are reserved for all online exits’ use (APPC Security, Wake Up Initiate, and Wake Up Terminate).

Sample Wake Up Terminate User Exit STCWT

The STCWT program demonstrates how to do the following:

- Examine fields in the STIPS header and trailer
- Format a report of the Mailbox ID, batch number, user batch ID, and return code for the wake up transaction

STCWT Program Logic

The STCWT program performs the following processing:

- Opens a report file to report on wake up transaction activity.
- Writes information to the report file on each wake up transaction completed.

Implementing STCWT

To run STCWT with online Connect:Enterprise, follow these instructions:

1. Use an ODF that contains the following:

```
*OPTIONS
  ...
  XAPPCWT=STCWT
```

2. Provide a DD statement in Connect:Enterprise JCL for the report file such as the following:

```
//CWTREPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
```

3. Test in conjunction with the Wake Up Initiate and End Of Batch exits.

Using the COBOL User Exit

A user exit is provided to illustrate how to link the online exits if you are writing them in COBOL.

Sample COBOL User Exit STCOBOL

The STCOBOL program demonstrates how to do the following:

- Specify proper linkage for COBOL online exits
- Use Security exit one to force the use of unique passwords for each Mailbox ID when using a full $$DIRECTORY request

STCOBOL Program Logic

The STCOBOL program performs the following processing:

- If the $$ command type is not a $$DIRECTORY, returns to Connect:Enterprise for normal processing.

- If the password is all blanks (was not supplied), returns to Connect:Enterprise for normal processing.
- Looks up the 8-byte Mailbox ID in a security table in the program. If the ID is not in the table, sets the action code to X'08' for a security violation and returns to Connect:Enterprise.
- If the Mailbox ID is found in the security table, compares the password to the password in the security table.
- If the passwords match, changes the transaction password to BANANA and changes the transaction Mailbox ID to blanks. Note that BANANA is the valid Connect:Enterprise password.
- If the passwords do not match, changes the transaction password to ZZZZZZZZ to force a password failure.
- Returns to Connect:Enterprise to process the $$DIRECTORY normally.
- When this exit is being used, the parameters supplied for a $$DIRECTORY from a remote site differ slightly from the standard Connect:Enterprise method. On a full directory request, an ID must be supplied. It is examined by the exit, then changed to blanks if it is correct.

  To request a full directory listing of all IDs:

```
$$DIRECTORY PASSWORD=xxxxxxxx ID=xxxxxxxx
```

```
$$DIRECTORY ID=xxxxxxxx
```

Implementing STCOBOL

To run STCOBOL with online Connect:Enterprise, follow these instructions:

1. Use an ODF that contains:

```
*OPTIONS
  VTAM=YES
  PASSWORD=BANANA
  ...
  XSECUR1=STCOBOL
```

2. The following ID/PASSWORD combinations are allowed:

```
ID                  PASSWORD
OPER01              APPLE
OPER02              ORANGE
OPER03              FIG
MARYJANE            PEAR
FRED                PEACH
```

3. Test a variety of $$ commands and a variety of $$DIRECTORY commands.

```
OK:    $$DIRECTORY ID=MARYJANE PASSWORD=PEAR
OK:    $$REQUEST ID=MARYJANE
OK:    $$ADD ID=MARYJANE BATCHID=‘PAYROLL DATA’
       ...
       ...
FAIL:  $$DIRECTORY PASSWORD=BANANA
FAIL:  $$DIRECTORY ID=OPER01 PASSWORD=FRUIT
OK:    $$DIRECTORY ID=TEST
OK:    $$DIRECTORY ID=OPER03
```

Chapter 5

Using Connect:Enterprise Offline Utility Exits

Connect:Enterprise provides several offline utility exits. Connect:Enterprise only provides the linkage to the offline utility exits. You must define, code, assemble, link, and test your own exits. Offline utility exit programs are optional and the system default is no exits.

The following offline utility user exits are available:

- Offline ADD Security exit
- Offline EXTRACT Security exit
- Offline STATFLG/DELETE/ERASE/MOVE/PURGE Security exit
- Offline Utility Startup exit

When the exits are in control, the Connect:Enterprise offline utility is not. *Define, assemble, link, and test your programs carefully* because, when an ABEND occurs (the system goes into a loop), the entire utility is affected.

This chapter describes the different types of offline exits and their uses.

Note: You can use the sample member, ASMLKXIT, to assemble and link your user exits.

How Connect:Enterprise Uses Offline Utility Exits

All Connect:Enterprise offline utility user exits are called using standard MVS CALL linkage. The executable user exit load modules must be present in your Connect:Enterprise utility load library.

An exit parameter list is passed to each of the user exits. The parameter list points to all of the data that can be examined and modified by the exit programs. The calling parameters and action codes for each of the user exit programs are documented with the descriptions for each type of user exit. For all the exits, the parameter list is a series of 4-byte addresses that point to the actual data field. The data field contents description uses letter codes to describe the field format. C is character data; X is hexadecimal data.

The M$OUXCB macro on the Connect:Enterprise release tape contain the parameter lists in the DSECT for the offline utility exits.

Assembly language programs must save and restore the calling program registers. At entry to the user exit, the parameter list address is in Register 1.

Coding Offline Utility Exits

The following requirements apply to all user exits. You must follow these rules for proper execution of all exits.

- When changing parameters passed to your exit programs, always use values that are valid for use by Connect:Enterprise. Failure to use valid data can cause system ABENDs or unpredictable results. All character fields are passed as left-justified, blank-filled, and should be returned in the same format.
- Ensure that the passed parameter list always contains valid addresses for all fields defined for each user exit.
- The default action code is X‘00’, which indicates that Connect:Enterprise should continue processing as normal. If an invalid action code is set, Connect:Enterprise forces an action code of X‘00’.

Testing Offline Utility Exits

Test the user exits before they are used in a production Connect:Enterprise system. An ABEND or loop in a user exit affects all of Connect:Enterprise, not just your exit load module.

Sample Offline Utility Exits

The Connect:Enterprise release tape contains several sample offline user exit programs.

Sample User Exit Programs	Definition
STSECA	Offline ADD Security exit
STSECE	Offline EXTRACT Security exit
STSECOU	Offline STATFLG/DELETE/ERASE/MOVE/PURGE Security exit
STUTAXIT	Offline Utility Startup exit

Note: When implementing an offline user exit, you *must* use the same name of the corresponding sample exit.

Using the Offline ADD Security Exit

The Offline ADD Security exit is called at the beginning of each offline ADD request. Use this exit to:

- Perform security verification
- Set an action code that indicates to either continue normally or to not perform the ADD request

ADD Security Exit Parameters

Parameters passed to the ADD Security exit are addressed by the XO$DSECT in the M$OUXCB macro. A listing of the parameters and the contents of the data field pointed to by the parameters is in the following table:

DSECT Label	Contains Address of	Data Field Contents
XO$ID	Mailbox ID	8-byte Mailbox ID
XO$BID	Batch Identification	64-byte batch ID
XO$ACODE	Area to Return Action Code	1-byte exit action code X‘00’ = Process as Normal X‘08’ = Terminate ADD Request X‘0C’ = Terminate entire offline utility job stream.
XO$VPF	VPF DSN	44-byte VPF DSN
XO$PGMID	Utility Name	8-byte Utility Name (ADD)
XO$MBXNM	Mailbox Name	8-byte name assigned to the Mailbox system.
XO$SFLAG	Security Flag	Flag that indicates if security processing is active. This flag could also be set on to skip normal security processing.
XO$VBQNO	VBQ ID (ADD only)	2-byte VBQ ID to which the batch is added.

ADD Security Exit Requirements

The following requirements apply to the Offline ADD Security exit. You must follow these rules for proper exit use.

The purpose of the ADD Security exit is to allow a security check to be performed.

- If the action code pointed to by XO$ACODE is set to X‘08’, the batch is not added.
- If you want the entire job stream to be terminated, set the return code in register 15 to 12 (X‘0C’).
- You can set or alter the VBQ ID for the VBQ to which the batch is added. The user exit is responsible for verifying the VBQ is defined and can be allocated.

Sample ADD Security User Exit STSECA

The STSECA program demonstrates how to do the following:

- Retrieve the userid of the submitted job

- ✦ Verify user authority to add batches with specific Mailbox ID and user batch ID values
- ✦ Verify the VPF DSN

You can use this exit to override the ID and BATCHID values regardless of where these values originated from, for example, in a SYSIN control record or an embedded $$ADD ID=xxxxxxxx BATCHID='xxx...xxx' record. See the sample STSECA source code for an example of performing a simple table lookup against the input ID/BATCHID and altering these values, based on a match.

STSECA Program Logic

The STSECA sample user exit program performs the following processing:

- ✦ Obtains the user ID for the job and then checks the user ID against the authorization table.
- ✦ If a match is found, another check is made against the table entry to see if the 8-byte Mailbox ID, 64-byte user batch ID, and VPF DSN are allowed for this user. If so, the action is allowed. If not, subsequent table items are searched. When the table is exhausted, the action is disallowed.

The following is an example of an authorization table:

Sample Authorization Table in STSECA	Description
DC CL4‘UID1’	First 4 bytes of user ID
DC CL4‘MID1’	Allowed Mailbox ID (first 4 bytes)
DC CL10‘USER BID1’	Allowed user batch ID (first 10 bytes)
DC CL44’VPF.TEST1’	44-byte VPF data set name
DC CL4‘UID2’	First 4 bytes of user ID
DC CL4‘MID2’	Allowed Mailbox ID (first 4 bytes)
DC CL10‘USER BID2’	Allowed user batch ID (first 10 bytes)
DC CL44’VPF.TEST2’	44-byte VPF data set name
DC CL4‘UID3’	First 4 bytes of user ID
DC CL4‘MID3’	Allowed Mailbox ID (first 4 bytes)
DC CL10‘USER BID3’	Allowed user batch ID (first 10 bytes)
DC CL44’VPF.TEST3’	44-byte VPF data set name

Implementing STSECA

To run STSECA with the offline utilities, follow these instructions:

1. Code your version of STSECA. If you do not code your own version of STSECA, use the load module shipped with the installation tape.
2. Assemble and link STSECA into your offline utility load library.

Using the Offline EXTRACT Security Exit

The Offline EXTRACT Security exit is called at the beginning of each offline EXTRACT request. Use this exit to:

- Perform security verification
- Set an action code that indicates to either continue normally or to not extract the batch

EXTRACT Security Exit Parameters

Parameters passed to the EXTRACT Security exit are addressed by the XO$DSECT in the M$OUXCB macro. A listing of the parameters and the contents of the data field pointed to by the parameters is in the following table:

DSECT Label	Contains Address of	Data Field Contents
XO$ID	Mailbox ID	8-byte Mailbox ID
XO$BID	Batch Identification	64-byte batch ID
XO$ACODE	Area to Return Action Code	1-byte exit action code X‘00’ = Process as normal X‘08’ = Terminate EXTRACT Request X’0C’ = Terminate entire offline utility job stream
XO$VPF	VPF DSN	44-byte VPF DSN
XO$PGMID	Utility Name	8-byte Utility Name (EXTRACT)
XO$MBXNM	Mailbox Name	8-byte name assigned to the Mailbox system.
XO$SFLAG	Security Flag	Flag that indicates if security processing is active. This flag could also be set on to skip normal security processing.
XO$VBQNO	VBQ ID (ADD only)	2-byte VBQ ID to which the batch is added.

EXTRACT Security Exit Requirements

The following requirements apply to the Offline EXTRACT Security exit. You must follow these rules for proper exit use. The purpose of the EXTRACT Security exit is to allow a security check to be performed.

- If the action code pointed to by XO$ACODE is set to X‘08’, the batch is not extracted.
- If you want the entire job stream to be terminated, set the return code in register 15 to 12 (X‘0C’).

Sample EXTRACT Security User Exit STSECE

The STSECE program demonstrates how to do the following:

- Retrieve the user ID of the submitted job
- Verify user authority to extract batches with specific Mailbox ID, User Batch ID, and VPF DSN

STSECE Program Logic

The STSECE sample user exit program performs the following processing:

- Obtains the user ID for the job, and then checks the user ID against the authorization table.
- If a match is found, makes another check against the table entry to see if the 8-byte Mailbox ID, 64-byte user batch ID, and VPF DSN is allowed for this user. If so, the action is allowed. If not, subsequent table items are searched. When the table is exhausted, the action is disallowed.

The following is an example of an authorization table:

Sample Authorization Table in STSECE	Description
DC CL4‘UID1’	First 4 bytes of user ID
DC CL4‘MID1’	Allowed Mailbox ID (first 4 bytes)
DC CL10‘USER BID1’	Allowed user batch ID (first 10 bytes)
DC CL44’VPF.TEST1’	44-byte VPF data set name
DC CL4‘UID2’	First 4 bytes of user ID
DC CL4‘MID2’	Allowed Mailbox ID (first 4 bytes)
DC CL10‘USER BID2’	Allowed user batch ID (first 10 bytes)
DC CL44’VPF.TEST2’	44-byte VPF data set name
DC CL4‘UID3’	First 4 bytes of user ID
DC CL4‘MID3’	Allowed Mailbox ID (first 4 bytes)
DC CL10‘USER BID3’	Allowed user batch ID (first 10 bytes)
DC CL44’VPF.TEST3’	44-byte VPF data set name

Implementing STSECE

To run STSECE with the offline utilities, complete the following steps:

1. Code your version of STSECE. If you do not code your own version of STSECE, use the load module shipped with the installation tape.
2. Assemble and link STSECE into your offline utility load library.

Using the Offline STATFLG/DELETE/ERASE/MOVE/PURGE Security Exit

This user exit is called at the beginning of each offline STATFLG, DELETE, ERASE, MOVE, and PURGE request. Use this exit to:

- Perform security verification
- Set an action code that indicates to either continue normally or to not perform the function

STATFLG/DELETE/ERASE/MOVE/PURGE Security Exit Parameters

Parameters passed to the STATFLG/DELETE/ERASE/MOVE/PURGE Security exit are addressed by the XO$DSECT in the M$OUXCB macro. A listing of the parameters and the contents of the data field pointed to by the parameters is in the following table:

DSECT Label	Contains Address of	Data Field Contents
XO$ID	Mailbox ID	8-byte Mailbox ID
XO$BID	Batch Identification	64-byte batch ID
XO$ACODE	Area to Return Action Code	1-byte exit action code X‘00’ = Process as normal X‘08’ = Terminate STATFLG/DELETE/ ERASE/MOVE/ PURGE X’0C’ = Terminate entire offline utility job stream
XO$VPF	VPF DSN	44-byte VPF DSN
XO$PGMID	Program ID	8-byte STATFLAG, DELETE, ERASE, MOVE, or PURGE (left justified, blank-filled)
XO$MBXNM	Mailbox Name	8-byte name assigned to the Mailbox system.
XO$SFLAG	Security Flag	Flag that indicates if security processing is active. This flag could also be set on to skip normal security processing.
XO$VBQNO	VBQ ID (ADD only)	2-byte VBQ ID to which the batch is added.

Note: Since the PURGE Utility is not batch specific, the Mailbox ID and user batch ID fields are blank.

STATFLG/DELETE/ERASE/MOVE/PURGE Security Exit Requirements

The purpose of the STATFLG/DELETE/ERASE/MOVE/PURGE Security exit is to allow a security check to be performed.

The following requirements apply to the Offline STATFLG/DELETE/ERASE/MOVE/PURGE Security exit. You must follow these rules for proper exit use.

- If the action code pointed to by XO$ACODE is set to X‘08’, the function is not performed.
- If you want the entire job stream to be terminated, set the return code in register 15 to 12 (X‘0C’).

Sample Security User Exit STSECOU

The STSECOU sample user exit program demonstrates how to do the following:

- Retrieve the user ID of the submitted job
- Verify user authority to alter statflags, delete, erase, or move batches with specific Mailbox ID and user batch ID values
- Verify the VPF DSN
- Verify Program ID and use separate tables

STSECOU Program Logic

The STSECOU sample user exit program performs the following logic:

- Obtains the user ID for the job and the checks the user ID against the authorization table.
- If a match is found, another check is made against the table entry to see if the 8-byte Mailbox ID, 64-byte user batch ID and VPF DSN is allowed for this user. If so, the action is allowed. If not, subsequent table items are searched. When the table is exhausted, the action is disallowed.

The following is an example of an authorization table:

Sample Authorization Table in STSECOU	Description
DC CL4‘UID1’	First 4 bytes of user ID
DC CL4‘MID1’	Allowed Mailbox ID (first 4 bytes)
DC CL10‘USER BID1’	Allowed user batch ID (first 10 bytes)
DC CL44’VPF.TEST1’	44-byte VPF data set name
DC CL4‘UID2’	First 4 bytes of user ID
DC CL4‘MID2’	Allowed Mailbox ID (first 4 bytes)
DC CL10‘USER BID2’	Allowed user batch ID (first 10 bytes)
DC CL44’VPF.TEST2’	44-byte VPF data set name
DC CL4‘UID3’	First 4 bytes of user ID
DC CL4‘MID3’	Allowed Mailbox ID (first 4 bytes)
DC CL10‘USER BID3’	Allowed user batch ID (first 10 bytes)
DC CL44’VPF.TEST3’	44-byte VPF data set name

Implementing STSECOU

To run STSECOU with the offline utilities, complete the following steps:

1. Code your version of STSECOU. If you do not code your own version of STSECOU, use the load module shipped with the installation tape.
2. Assemble and link STSECOU into your offline utility load library.

Using the Offline Utility Startup Exit

The Startup exit is called only once at the beginning of each STOUTL execution. The exit can be used for conversion aid and security.

Use this exit to set the default VSAM File Server name and the VPF data set name which are required when executing any offline utilities. This exit enables you to use existing JCL members without having to update each one with the two parameters mentioned above; however, other JCL changes must still be made to some DD statements.

This exit can also be used for offline security authorization. For example, the exit could retrieve the user ID of the submitted job and check it against a table of authorized users. If the authorization check fails, the exit can set the return code to 12 (X‘0C’) in register 15 which terminates execution of the job. For examples of retrieving and checking the submitted job’s user ID, see any of the other offline utility security exits (STSECA, STSECE, or STSECOU).

This exit can also perform security by ensuring the VSAM File Server name is valid. If the VSAM File Server name is not valid, the job can be terminated with a return code 12 (X‘0C’) in register 15.

Startup Exit Parameters

Parameters are passed to the Startup exit using the standard convention. Register 1 points to a parameter list containing the addresses of the data. A listing of the parameters and the contents of the data field pointed to by the parameters is shown in the following table:

Parameter List Offset	Contains Address of	Data Field Contents
0 (4)	Area to return Action Code	1-byte action code X‘00’ = No returned Server Name/VPF DSN X‘01’ = Use returned Server/VPF DSN
4 (4)	VSAM File Server Name	4-byte VSAM File Server Name (returned if action code = X‘01’). Upon entry, this parameter is that set by the user or “SRV1".
8 (4)	VPF Cluster Name	44-byte DSN of VPF (returned if action code = X‘01’)

Startup Exit Requirements

The purpose of the Startup exit is either to enable setting of the default VSAM File Server name and the VPF DSN or allow performing of security authorization.

- If the user exit sets the action code to X'01', the offline utility (STOUTL) expects the 4-byte VSAM File Server name and the 44-byte VPF cluster name to be filled in by the exit. STOUTL then uses these values passed back on return from the exit.
- Verify that the return code is set to 0 in register 15. If you use the exit for security purposes and want the entire job stream terminated, set the return code in register 15 to 12 (X'0C').

Sample Startup User Exit STUTAXIT

The STUTAXIT program demonstrates how to do the following:

- Set the VSAM File Server name used by the utilities
- Set the VPF by the utilities

STUTAXIT Program Logic

The STUTAXIT sample user exit program performs the following processing:

- A flag field (USENAMES) activates or deactivates this program.
- If USENAMES is X'00' (the default), the VSAM File Server name and VPF cluster name cannot be set by the program, in which case they must be provided in the JCL.
- If USENAMES is X'01', the VSAM File Server name and VPF cluster name is set by the program and used for STOUTL execution. The Server name used in the EXIT overrides the JCL, but the VPF cluster name in the EXIT is only used if VPF= is not coded in the SYSIN.

Implementing STUTAXIT

To run STUTAXIT with the offline utilities, complete the following steps:

1. Code your version of STUTAXIT.

Note: If you do not code your own version of STUTAXIT, use the load module shipped with the installation tape.

2. Assemble and link STUTAXIT into your offline utility load library.

Chapter 6

Using the Connect:Enterprise CSCU Startup Exit

The Cross System Client Utility (CSCU) user exit is called once, at the beginning of each STCSC000 execution. Use this exit to:

- Override the SYSIN2 processing parameters specified in the JCL.
- Perform a security check and set an action code that indicates to either continue normally or to end the job.

Caution: Test the user exit before it is used in a production environment. An ABEND or loop in a user exit can affect the operation of the entire Connect:Enterprise system, not just your exit load module. You can use the sample member, ASMLKXIT, to assemble and link your user exits.

Specifying Preprocessing Parameters with the STCSCUSR User Exit

You can use the STCSCUSR user exit to assign or override preprocessing parameters. This user exit is called during CSCU startup. The parameters are passed in a standard parameter list. Register 1 (R1) points to a parameter list containing a list of addresses. Upon return from the user exit, the CSCU uses the specified parameters.

The following table shows the STCSCUSR user exit parameters.

R1 Points to Parameter List	Contains Address of	Description
+0(4)	SYSIN DD name	8-byte DD name for the SYSIN file.
+4(4)	PRINT DD name	8-byte DD name for the PRINT file.
+8(4)	INPUT DD name	8-byte DD name for the INPUT file.

R1 Points to Parameter List	Contains Address of	Description
+12(4)	OUTPUT DD name	8-byte DD name for the OUTPUT file.
+16(4)	SYSPRT DD name	8-byte DD name for the SYSPRINT file.
+20(4)	LOGFILE DD name	8-byte DD name for the LOGFILE file.
+24(4)	SNAPOUT DD name	8-byte DD name for the SNAPOUT file.
+28(4)	Local VTAM APPL	8-byte name of the local (CSCU) VTAM APPL.
+32(4)	Remote VTAM APPL	8-byte name of the remote (Connect:Enterprise) APPL.
+36(4)	Logmode Table Entry Name	8-byte name of the Logmode table defining the LU6.2 session characteristics.
+40(4)	Enterprise User ID	8-byte Connect:Enterprise user ID used for security authorization.
+44(4)	Enterprise User Password	8-byte Connect:Enterprise user password used for security authorization.
+48(4)	Enterprise Name	8-byte (Currently not used).
+52(4)	Jobname	8-byte job name.
+56(4)	Stepname	8-byte step name.
+60(4)	User Id	8-byte user ID of the person who submitted the job.
+64(4)	Return code	1-byte exit action code. The values are: x'00' = Continue as normally x'08' = End job with user return code 08 x'nn' = End job with user return code nn

Sample Cross System Client Startup Exit STCSCUSR

The STCSCUSR sample user exit program demonstrates how to do the following:

- Print the parameter list data values passed from the caller (STCSC000).
- Verify user authority to execute the CSCU based on job name, job submitter, and destination Connect:Enterprise name.
- Validate the password of the specified User ID.

STCSCUSR Program Logic

The STCSCUSR program demonstrates how to do the following:

- Print the parameter list data values passed from the caller (STCSC000).
- If a match is found against authorization table 1 (job name table), another check is made to see if the job submitter has authority to execute the CSCU against the targeted Connect:Enterprise system.

- If a successful match is made against the job name authorization table, a second check is made against authorization table 2 (user/password table), to see if the specified Connect:Enterprise user ID and password match.
- If either of the above checks fail, the return code is set to x'08' to end the job with an RC=08.

The following is a sample job name table.

Authorization Table 1	Description
DC CL8'JOBNAME1'	8-byte job name
DC CL*'JOBUSER1'	8-byte user ID of the person who submitted the job
DC CL8'ENTPRS1'	8-byte Connect:Enterprise System Name
DC CL8'JOBNAME2'	8-byte job name
DC CL8'JOBUSER2'	8-byte user ID of the person who submitted the job
DC CL8'ENTPRS2'	8-byte Connect:Enterprise System Name
DC CL8'JOBNAME3'	8-byte job name
DC CL8'JOBUSER3'	8-byte user ID of the person who submitted the job
DC CL8'ENTPRS3'	8-byte Connect:Enterprise System Name

The following is a sample user/password authorization table.

Authorization Table 2	Description
DC CL8'USERID1'	8-byte name of the Connect:Enterprise user ID
DC CL8'USERPW1'	8-byte password for the Connect:Enterprise user ID
DC CL8'USERID2'	8-byte name of the Connect:Enterprise user ID
DC CL8'USERPW2'	8-byte password for the Connect:Enterprise user ID
DC CL8'USERID3'	8-byte name of the Connect:Enterprise user ID
DC CL8'USERPW3'	8-byte password for the Connect:Enterprise user ID

Implementing STCSCUSR

Perform these steps to run STCSCUSR with the Cross System Client:

1. Code your version of STCSCUSR.

> **Note:** If you do not code your own version of STCSUSR, use the load module shipped with the installation tape.

2. Assemble and link STCSCUSR into your Cross System Client load library.

Chapter 7

Using the VSAM File Server Exit

An optional VSAM File Server user exit is provided to call the VSAM File Server each time a request is made to allocate and open a VSAM data set that is not currently open. This exit can then examine the cluster name requested and determine if the open should occur.

Connect:Enterprise only provides the linkage to this exit. You must define, code, assemble, link, and test your own exit. *The VSAM File Server exit program is optional and the system default is NO exit.*

The VSAM File Server exit uses standard CALL linkage. Therefore, when the exit is in control, the VSAM File Server is not in control. *Define, assemble, link, and test your programs carefully* because, when an ABEND occurs, the entire VSAM File Server system is affected.

No testing or debugging aids are built into the VSAM File Server exit. Use standard Assembler language debugging and testing techniques.

This chapter describes the exit available for the VSAM File Server.

Note: You can use the sample member, ASMLKXIT, to assemble and link your user exits.

How Connect:Enterprise Uses the VSAM File Server Exit

The VSAM File Server Open exit is called by the VSAM File Server each time a specific request is made by a client task, either by Connect:Enterprise or offline utilities. Each exit then determines if the request should be processed. If the exit determines that processing should not continue, a negative response is returned to the client.

The identity of the client task is not always known by the VSAM File Server exits when invoked. Your user exit programs must take this into consideration.

To activate the VSAM File Server exit, replace the dummy exit version that was provided with your version. To deactivate the VSAM File Server exit, replace your version with the dummy version.

Because of the internal relationship between the C language and the VSAM File Server, the exit and entry macros that are provided in the sample VSAM File Server exit should be used all the time, even if you modify the exit to suit your needs.

Coding the VSAM File Server Exit

The following requirements apply to the VSAM File Server exit. You must follow these rules for the proper execution of the exit.

- Parameters defined to exist for the exit are always passed to the exit.
- Alter only the parameters that are recommended for alteration; unpredictable results an occur if other parameters are altered.
- Code exits to assume that they are always being entered for the first time.

Note: While your VSAM File Server exit is executing, the VSAM File Server is waiting for it to return from the CALL and is not processing other transactions. For this reason, avoid writing exits that excessively slow down the system.

Testing the VSAM File Server Exit

Fully test VSAM File Server exits before implementing them on a production Connect:Enterprise system. An ABEND or loop in an exit could cause the VSAM File Server and Connect:Enterprise to terminate completely.

No trace or snap features are provided with the VSAM File Server exits. Use standard Assembler language debugging and testing techniques.

Sample VSAM File Server Exit

The Connect:Enterprise release tape contains the BTVSMOSX VSAM File Server Open sample user exit program in the source library.

Using the VSAM File Server Open User Exit

The VSAM File Server Open exit is called by the VSAM File Server each time a request is made to allocate and open a VSAM data set that is not currently open. This exit can then examine the cluster name requested and determine if the open should occur.

VSAM File Server Open User Exit Parameters

A single parameter is passed to the Open User exit. This parameter is the address of the data set name requested. The data set name is left justified and padded to 44 characters with hex zeros. This data set name cannot be changed.

Upon return from the Open User exit, the return code value passed in register 15 is examined. If the return code value is zero, processing continues and the allocation/open attempted. If the return code value is non-zero, the allocation/open request is denied and a return code=8, reason=98 (authorization failed) is passed back to the requester.

VSAM File Server Open User Exit Requirements

The following requirements apply to the VSAM File Server Open exit. You must follow these rules for proper exit use.

- Link this exit as RENT and AMODE=31.
- Link this exit as a separate module that is loaded by the VSAM File Server each time it is needed. The module name must be BTVSMOSX.
- If the VSAM File Server is executing when you link this module, you must stop the server and restart it.
- Because this exit resides in the VSAM File Server, no trace facility is available.
- You do not need to change any startup parameters to activate this exit. The existence of the module in the load library is all that is required.
- To deactivate the exit, replace it with the dummy BTVSMOSX module supplied with the product.
- A sample of this exit is found in member BTVSMOSX in the source library.

Sample Open User Exit BTVSMOSX

The BTVSMOSX program demonstrates how to do the following:

- Examine the data set name passed as input
- Compare the data set name against a static list of data set names, either specific names or generic names

BTVSMOSX Program Logic

The BTVSMOSX sample user exit program performs the following processing:

- Obtains the input data set name
- Sets the default return code to a nonzero value that fails the open request
- Loops through the table of data set names that are allowed by this VSAM File Server
- When a match is found, resets the return code to zero and exits the loop
- Returns with the set return code.

Implementing BTVSMOSX

To run BTVSMOSX with the VSAM File Server, complete the following steps:

1. Modify the data set name table (CLUSTTBL) to include data sets that should be used by the VSAM File Server. Assemble and link the program into the library used by the VSAM File Server.

2. If the VSAM file server is already active, stop and restart it.
3. Test the allocation/open by starting an online Connect:Enterprise system that uses data sets listed in the table.
4. Test the allocation/open by starting an online Connect:Enterprise system that uses data sets not listed in the table.

Chapter 8

Using ISPF Interface User Exits

You can supply two user exit programs that are called by the ISPF interface. The exits available are as follows:

- Function Initiate Security exit
- Function Request Security exit

The Function Initiate Security exit is invoked when each ISPF function or function menu is selected. The Function Request Security exit is invoked when a request is sent to the Connect:Enterprise system. The return code returned by each exit determines if processing continues or is not allowed.

This chapter describes the two available ISPF interface user exits and their use.

Note: You can use the sample member, ASMLKXIT, to assemble and link your user exits.

Coding ISPF interface User Exits

Write each exit as a single executable load module. Each load module is loaded once during ISPF interface startup and called (branch entry) each time processing is required. A dummy version of each exit is supplied with the ISPF interface.

Testing ISPF Interface User Exits

When the exits are in control, the ISPF interface is not. Define, assemble, link, and test your programs carefully because, if an ABEND occurs, the entire ISPF interface is affected.

Sample ISPF Interface User Exits

The Connect:Enterprise release tape contains several sample ISPF user exit programs. Use the sample programs as guidelines in coding your own ISPF interface exits:

Sample User Exit Programs	Description
MZMCPFIX	Function initiate security exit
MZAPCFRX	Function request security exit

Using the Function Initiate Security Exit

This exit is called each time an ISPF user attempts to invoke an ISPF interface function or selects a lower level menu panel. Within the exit, the ISPF Administrator can control which functions (or groups of functions) can be used by individual users.

Function Initiate Security Exit Parameters

Parameters are passed to the Function Initiate Security exit using the standard convention. Register 1 points to a parameter list containing the address of the data. The data pointed to is as follows:

Parameter List Offset	Contains Address of	Data Field Contents
0(4) Function String (left justified and blank filled)		20-byte Function Control String: Administration All functions under main panel option 10
		GLOBALDEFAULTS Option 10.1
		CONNECTIONDEFINITION Option 10.2
		INTERFACEDEFINITIONS Option 10.3
		DISPLAYDEFINITIONS Option 10.4
		REINITIALIZE Option 10.5
		SYSTEMTRACES Option 10.6
		USERFUNCTIONS All functions under main panel option 20

Parameter List Offset	Contains Address of	Data Field Contents
		BATCHFILEREPORTING All functions under main panel option 21
		BATCHQUEUEFUNCTIONS All functions under main panel option 22
		AUTOCONNECTMODEL All functions under main panel option 23 and option 20.8
0(4) Function String (left justified and blank filled)		BATCHUTILITYFUNCTION All functions under main panel option 24 and option 20.9
		AUTOCONNECTSUMMARY Option 20.1 and 21.1
		AUTOCONNECTDETAIL Option 20.2 and 21.2
		REMOTECONNECTSUMMARY Option 20.3 and 21.3
		REMOTECONNECTDETAIL Option 20.4 and 21.4
		QUEUEDAUTOCONNECT Option 20.5 and 21.5
		BATCHQUEUELIST Option 20.6 and 22.1
		BATCHUTILIZATIONSTAT Option 20.7 and 22.2
		BATCHUTILITYMODEL Option 20.9.91
		BATCHUTILITYSUBMIT Option 20.9.92
		ADDUTILITYMODEL Option 24.1
		EXTRACTUTILITYMODEL Option 24.2
		ADDUTILITYSUBMIT Option 24.3
		EXTRACTUTILITYSUBMIT Option 24.4
		LISTUTILITYSUBMIT Option 24.5
		STATFLGUTILITYSUBMIT Option 24.6
		DELETEUTILITYSUBMIT Option 24.7
		ERASEUTILITYSUBMIT Option 24.8
		PURGEUTILITYSUBMIT Option 24.9
		ACSUMMARYRPTSUBMIT Option 24.10
		ACDETAILRPTSUBMIT Option 24.11
		RCSUMMARYRPTSUBMIT Option 24.12
		RCDETAILRPTSUBMIT Option 24.13
		ACQRPTSUBMIT Option 24.14

Parameter List Offset	Contains Address of	Data Field Contents
		OFFLINERPTSUBMIT Option 24.15
		MOVEUTILITYSUBMIT Option 24.16
		OPERATORTASKS All functions under main panel option 30
		ISSUECOMMANDS All functions under main panel option 31
		MONITORACTIVITY All functions under main panel option 32
		ONLINEODFUPDATES All functions under main panel option 33
0(4) Function String (left justified and blank filled)		$$CONNECT Option 30.1 and 31.1
		$$DUMP Option 30.2 and 31.2
		$$LIST Option 30.3 and 31.3
		$$SHUTDOWN Option 30.4 and 31.4
		$$START Option 30.5 and 31.5
		$$STOP Option 30.6 and 31.6
		$$TRACE Option 30.7 and 31.7
		$$LISTFILES Option 30.8 and 31.8
		$$SPACE Option 30.9 and 31.9
		$$ALLOC Option 30.10 and 31.10
		$$DALLOC Option 30.11 and 31.11
		$$REFRESH Option 30.12 and 31.12
		$$INVOKE Option 30.13 and 31.13
		$$DIALOG Option 20.14 and 21.14
		ACTIVESESSIONS Option 30.21 and 32.1
		ACTIVEAUTOCONNECT Option 30.22 and 32.2
		ONLINEOPTIONSUPDATE Option 30.30.1 and 33.1
		ONLINESECURITYUPDATE Option 30.30.2 and 33.2
		ONLINECONNECTUPDATE Option 30.30.3 and 33.3
		ONLINEREMOTESUPDATE Option 30.30.4 and 33.4
		ONLINECALENDARUPDATE Option 30.30.7 and 33.7

Parameter List Offset	Contains Address of	Data Field Contents
		MESSAGELIBRARY All functions under main panel option 40
		SECURITY All functions under main panel option 50
		USERIDLIST All functions under main panel option 60 Option 30.3.5
		QUPDATE Option 31.3.4
		ACQ
4(4)	TSO Userid (left justified and blank filled)	8-byte User ID
8(4)	Result Field	Halfword Result Field H'0'—continue processing else —disallow processing

Function Initiate Security Exit Requirements

The load module name must be MZMCPFIX. A dummy module is supplied in the ISPF interface ISPLLIB library. To activate this exit, simply replace the dummy module with your user-written exit module.

Upon entry, the exit parameters indicate what function (or group of functions) is being accessed (parameter 1) and who is attempting the function (parameter 2). After determining if the user is allowed to use the function, set the result field (parameter 3) to the appropriate value. Any nonzero value in the result field denies access to the function (or group of functions). An appropriate message is displayed indicating that access is denied to the function. The user is allowed to continue using the other functions of the ISPF interface.

Because the Function Initiate Security exit load module is loaded once and called by branch entry, it is possible for the program to load in a security table during its first invocation and use it again later during subsequent calls.

Sample Function Initiate Security User Exit MZMCPFIX

The MZMCPFIX sample user exit program demonstrates how to do the following:

- ✦ Obtain the input parameters passed to the exit
- ✦ Validate a request using a security table
- ✦ Return a positive or negative response from the exit

MZMCPFIX Program Logic

The MZMCPFIX program performs the following processing:

- Stores all input parameters for use by the exit
- Begins searching the security table for a match to the 20 byte function string
- If a match is found, the 8-byte TSO user ID is checked for a match
- If a match is found the request is disallowed
- If no match is found, the next entry in the security table is checked
- The result field is set and the exit returns to the calling program

Using the Function Request Security Exit

The Function Request Security exit is called each time an ISPF user attempts to send a request to the Connect:Enterprise system. Within the exit, the ISPF Administrator can control which requests can be sent to certain Connect:Enterprise systems by which users.

Function Request Security Exit Parameters

Parameters are passed to the Function Request Security exit using the standard convention. Register 1 points to a parameter list containing the address of the data. The data pointed to is as follows:

Parameter List Offset	Contains Address of	Data Field Contents
0(4)	Addr of IPS Header	The IPS header is mapped by the C$H00 macro in the Connect:Enterprise source library. The IPS header identifies the user ID and encrypted password that is passed to the Connect:Enterprise system, the Connect:Enterprise system (and release level) the request is for. It also identifies the request type that is being sent to the Connect:Enterprise system. IPS request types (trailers) are individually documented and mapped in the Connect:Enterprise source library (all macros that begin with C$) and Appendix A, *IPS Trailers*. This data is read-only and should not be modified.
4(4)	Addr of IPS Trailer	The IPS trailer identifies the actual request being sent to the Connect:Enterprise system. This field is zero if a Security Function (option 50) is being sent. Each IPS trailer identifies the unique parameters inherent to that function. IPS trailers are individually documented and mapped in the Connect:Enterprise source library (all macros that begin with C$) and Appendix A, *IPS Trailers*. This data is read-only and should not be modified.
8(4)	TSO Userid (left justified and blank filled)	8-byte user ID

Parameter List Offset	Contains Address of	Data Field Contents
12(4)	Addr of MID Table Entry	This field is zero if a Security Function (option 50) is being sent. For all other requests, this information contained in this table entry identifies the Connect:Enterprise system the request is sent to. It shows the friendly name assigned to the Connect:Enterprise system, the operating system type, and the release level.

Function Request Security Exit Requirements

The load module name must be MZAPCFRX. A dummy module is supplied in the ISPF interface ISPLLIB library. To activate this exit, simply replace the dummy module with your user-written exit module.

Upon entry, the exit parameters indicate the following information:

- Function request information, through the IPS header
- Request specific information, through the IPS trailer
- The requester, through the TSO user ID
- Identify the Connect:Enterprise system, through the MID table entry

After determining if the user is allowed to send the function request, the exit sets register 15 to an appropriate value. Any nonzero value in register 15 denies access to the function request. An appropriate message is displayed indicating that the requested function was denied. The user can continue making requests to the same and other Connect:Enterprise systems.

Because the Function Request Security Exit load module is loaded once and called by branch entry, it is possible for the program to load in a security table during its first invocation and use it again later during subsequent calls.

Sample Function Request Security User Exit MZAPCFRX

The MZAPCFRX sample user exit program demonstrates how to do the following:

- Obtain the input parameters passed to the exit
- Obtain information from each control block pointed to by the input parameters
- Validate a request using a security table
- Return a positive or negative response from the exit

MZAPCFRX Program Logic

The MZAPCFRX program performs the following processing:

- Stores all input parameters for use by the exit.
- Obtains information from the request Header.
- Obtains information from the request Trailer, if one was available.

- Obtains information for the MID table entry, if one was available.
- Begins searching the security table for a match to the 8-byte Mailbox name.
- If a match is found, the 8-byte Trailer ID (function type) is checked for a match.
- If a trailer ID match is found, the mailbox user ID from the Header is checked against the security table. If the table entry is blank or the name matches, the request is disallowed.
- If a Trailer ID match is found, the TSO user ID is checked against the security table. It the table entry is blank or the name matches, the request is disallowed.
- If a Trailer ID match is found, the Enterprise system type (MVS or VSE) from the MID table entry is checked against the security table. If the table entry is blank or the type matches, the request is disallowed.
- If no mailbox name and trailer ID match is found, the next entry in the security table is checked.
- The result field is set and the exit returns to the calling program.

Chapter 9

Using CICS Interface User Exits

The user-written exit feature is available in the CICS interface only during interactive operation. These exits are not available to a user-written Application Programming Interface (API) program. Any exit (or security) processing required by the API process is the responsibility of the API designer and programmer.

The optional user exits can be invoked to examine or record information that is in process. No modification of data values is possible from the user-written exits in the CICS interface. A return code parameter is provided to the user-written exit program to allow the exit program to communicate an accept/reject condition back to the CICS interface. If the return code is set to accept, the activity in process is completed. Otherwise, the process is terminated with an error message displayed to the terminal user.

Initialization and termination exit points are provided to allow special global level operations to be performed prior to CICS interface initialization or termination or both. An exit point is provided both before and after initial contact with the Connect:Enterprise system. An exit is also invoked before any data modification occurs. You must define, code, assemble, link, and test your own exit processing.

You can supply up to four user-written exit programs which are invoked by the CICS interface. The exits available are defined as follows:

- Initialization
- Security (invoked before and after the initial contact with Connect:Enterprise)
- Data Modification
- Termination

The exits are invoked at appropriate times during CICS interface processing to perform user defined functions. The return code value set by the exit can alter the normal CICS interface processing.

This chapter discusses the CICS interface User exits and their uses.

Note: You can use the sample member, ASMLKXIT, to assemble and link your user exits.

How Connect:Enterprise Uses CICS User Interface Exits

These exits are invoked by the CICS interface using a CICS LINK command. Therefore, when these exits are in control, the CICS interface is suspended waiting for the exit code to issue a CICS RETURN. In case of an ABEND, HANDLE processing in the CICS interface terminates the transaction.

Due to the real-time nature of the CICS interface and its relationship to Connect:Enterprise, the before image for a data modification exit is the data known to the transaction when it was last retrieved for processing. No enqueue or other form of protection is utilized to ensure that this data remains unmodified except when processing ODF updates. The modification (if allowed by the user-written exit program) can overwrite data that no longer matches the before image if another transaction has performed a similar modification request.

Implementing CICS Interface User Exits

To implement a user-written exit program, you must code the appropriate processing for the type of exit to be invoked. You can develop a single program to handle all four types of exit processing or you can develop an individual program for each of the four types of exit processing or some other suitable combination. You can activate as many of the exit types as you desire. None are required for operation of the CICS interface.

These programs must then be assembled (compiled) and link edited into a library defined to CICS. The CICS definitions (CEDA PROGRAM definition or DFHPPT macro definition) must be completed and finally, the programs must be identified to the CICS interface via the Interface System Exit Definition Update (Panel 1.4). The program name must be entered in each exit category for which it is to be invoked. If a single user-written exit program services all exit types, Panel 1.4 must specify the same program name for all exit types.

Coding CICS Interface User Exits

This section and the sample program supplied with the CICS interface aid in the design and testing of user-written exits. No application data can be modified by your exit program. The express purpose of user-written exits in the CICS interface system is to interrogate application data related to current processing at specific exit points and to accept (allow) or reject (disallow) the logical conclusion of the process. For example, if the user-written initialization exit determines the terminal user initiating the transaction should not be granted access to the application (based upon the User ID, Terminal ID, or other information available) and sets the return code as REJECT, the transaction is terminated before the terminal user sees the Primary Panel for the CICS interface.

The CICS interface sets a REJECT return code value in the COMMAREA prior to LINKing a user-written exit program. A zero return code value set by a user-written exit program indicates acceptance or acknowledgment of the in process activity and normal CICS interface processing

continues. A return code value other than zero set by the user-written exit program indicates that the in process activity should be terminated with an error message to the terminal user. The return code values allowed for each exit are documented with the description for each exit.

Because user-written exit programs are linked, the CICS interface is suspended expecting a return from your program. An ABEND HANDLE is provided in the CICS interface to free associated resources and terminate the transaction. If a user-written exit program has established a local ABEND HANDLE, it must conclude the local ABEND processing activities by issuing an EXEC CICS ABEND using the original ABEND code or an ABEND code of your designation. The CICS interface then regains control in its ABEND HANDLE logic and completes transaction termination.

Note: Prior release user-written exits MUST be reassembled using the new source members C$VSAM, EXITS and the related macros supplied in the current version installation source library.

Testing CICS Interface User Exits

User-written exit programs should be fully tested before they are used in a production environment. An ABEND temporarily terminates a terminal user's access to the system. Extreme mistakes in logic can result in the entire CICS interface being disabled. It can be beneficial to assemble the supplied test program EXITSAMP and review the execution and output from this program prior to installing your own version of an exit program.

Linking CICS Interface User Exits

Exit parameter data is passed to the exit program as a COMMAREA. All application data that can be examined by a specific exit program is included in this COMMAREA. In your installation source library is a member named EXITS that describes in detail each field of information provided to each exit. You should study this DSECT for complete understanding of the information available at each exit invocation. Typically, information available to the user-written exit program from CICS, such as EIB data and EXEC CICS ASSIGN data, is not supplied in the exit parameter data.

Several different DSECTs are provided to define the specific data type records in conjunction with the DSECT defined by the member EXITS. The STATFLG before and after data in the COMMAREA are defined by the DSECT supplied in macro C$U28. The Auto Connect Queue before and after data in the COMMAREA are defined by the DSECT supplied in macro C$064. The CICS interface model records and Help text records before and after data in the COMMAREA are defined by the DSECT supplied in macro C$VSAM. The Options Definition File (ODF) before and after data in the COMMAREA are defined by a series of DSECTs supplied in macro format. The macros involved are defined by name in comments following the label ODFTYPE in the EXITS member. They are also coded at the end of the sample program, member EXITSAMP in your installation source library.

The parameters passed to each user exit program are documented with the description for each type of exit.

Sample CICS Interface User Exits

The Connect:Enterprise release tape contains a sample program. The member named EXITSAMP in your CICS interface installation source library is an Assembler Language CICS Command Level program that demonstrates the use of each exit type. A review of this sample program reveals:

- How a user-written exit program receives the COMMAREA when the program is LINKed by the CICS interface
- How the data in the COMMAREA is examined for each exit type
- How, during a data modification exit, each data type is interrogated
- How the return code value is set to indicate acceptance of the CICS interface in process activity
- How all types of data modification can be rejected for a specific user ID

This program should be used as a guideline in coding your own user exits.

Using the Initialization Exit

This exit is invoked before an interactive user gains access to the CICS interface. The Connect:Enterprise Interface Primary Menu (Panel 0.0) is not displayed until after invocation of the user-written exit program generates an ACCEPT return code value. If an ACCEPT return code value is not set by the user-written exit program, the CICS task is terminated. The terminal user briefly sees a message stating: Access denied by initialization exit. This message is followed by a blank screen.

Initialization Exit Parameters

Parameters passed to the Initialization exit are addressed by the CMCIXITS DSECT in member EXITS in your CICS interface installation source library. A listing of the parameters and the content of the data fields passed to the user written exit program is shown in the following table.

DSECT label	Description of data	Data Field content
EXITLEN	Halfword (2-byte) length of the parameter data	Assembled value of DSECT label INITLEN
EXITID	An 8-byte character format identifier stamp	The character string CMCIXITS
EXITPROG	An 8-byte character format name of LINKed program	Data from Initialization field on Panel 1.4
EXITTYPE	A 1-byte character format type of exit indicator	Assembled value of DSECT label XINIT
EXITRC	A 1-byte hexadecimal return code value	Assembled value of DSECT label XRCABORT

Understanding Initialization Exit Usage

The purpose of the Initialization exit is to provide notification prior to a terminal user gaining access to the CICS interface. You can validate the identifying information (CICS User ID if specified, or the originating Terminal ID) as authorized to use the CICS interface. For charge-back or auditing use, you can record the identifying information, such as date, time, user, and terminal of the user beginning to use the CICS interface.

To allow the terminal user access to the CICS interface, the EXITRC field in the exit parameter data (COMMAREA) must be set to the assembled value of DSECT label XRCOKAY before executing a CICS RETURN in the user-written exit program.

Using the Security (Before) Exit

This exit is invoked after the User ID and Password for a specific Connect:Enterprise are entered on Panel 5.0 Connect:Enterprise Security Update, but before the connection to the specified Connect:Enterprise for verification of this information is initiated. The LU6.2 connection is not attempted until after invocation of the user-written exit program generates an ACCEPT return code value. If an ACCEPT return code value is not set by the user-written exit program, Panel 5.0 is re-displayed with the message: Access denied by security exit. The CICS interface functions that do not require contact with Connect:Enterprise can continue to be performed by the terminal user.

Security (Before) Exit Parameters

Parameters passed to the Security (Before) exit are addressed by the CMCIXITS DSECT in member EXITS in your CICS interface installation source library. A listing of the parameters and the content of the data fields passed to the user-written exit program is shown in the following table:

DSECT label	Description of data	Data Field content
EXITLEN	Halfword (2-byte) length of the parameter data	Assembled value of DSECT label SECLEN
EXITID	An 8-byte character format identifier stamp	The character string CMCIXITS
EXITPROG	An 8-byte character format name of LINKed program	Data from Security field on Panel 1.4
EXITTYPE	A 1-byte character format type of exit indicator	Assembled value of DSECT label XSEC1
EXITRC	A 1-byte hexadecimal return code value	Assembled value of DSECT label XRCABORT
SECSID	An 8-byte character format Connect:Enterprise Symbolic name (Not APPLID)	Connect:Enterprise Name entered on Panel 5.0 by the terminal user
SECSUSER	An 8-byte character format User ID for validation by Connect:Enterprise	Connect:Enterprise user ID entered on Panel 5.0 by the terminal user

DSECT label	Description of data	Data Field content
SECPSWD	An 8-byte character format Password for validation by Connect:Enterprise	Connect:Enterprise Password entered on Panel 5.0 by the terminal user
SECRC	A 2-byte hexadecimal process completion return code value	Not used when EXITTYPE = XSEC1

Understanding Security (Before) Exit Usage

The purpose of the Security (Before) exit is to provide notification prior to a terminal user gaining access to the specified Connect:Enterprise. You can validate the identifying information (CICS User ID if specified, or the originating Terminal ID) and any of the DSECT data as authorized to connect to the specified Connect:Enterprise. For charge-back or auditing use, the identifying information, such as date, time, user, and terminal of the user can be recorded.

To allow the terminal user to attempt initial connection with Connect:Enterprise, the EXITRC field in the exit parameter data (COMMAREA) must be set to the assembled value of DSECT label XRCOKAY before executing a CICS RETURN in the user-written exit program.

Using the Security (After) Exit

This exit is invoked after the connection processing has completed. It is solely a reporting function by the CICS interface to the user-written exit program. It in no way can be used to alter the processing path of the CICS interface.

Security (After) Exit Parameters

Parameters passed to the Security (After) exit are addressed by the CMCIXITS DSECT in member EXITS in your CICS interface installation source library. A listing of the parameters and the content of the data fields passed to the user-written exit program is shown in the following table:

DSECT label	Description of data	Data Field content
EXITLEN	Halfword (2-byte) length of the parameter data	Assembled value of DSECT label SECLEN
EXITID	An 8-byte character format identifier stamp	The character string CMCIXITS
EXITPROG	An 8-byte character format name of linked program	Data from Security field on Panel 1.4
EXITTYPE	A 1-byte character format type of exit indicator	Assembled value of DSECT label XSEC2

DSECT label	Description of data	Data Field content
EXITRC	A 1-byte hexadecimal return code value	Assembled value of DSECT label XRCABORT
SECSID	An 8-byte character format Connect:Enterprise Symbolic name (Not APPLID)	Connect:Enterprise Name entered on Panel 5.0 by the terminal user
SECSUSER	An 8-byte character format User ID for validation by Connect:Enterprise	Connect:Enterprise user ID entered on Panel 5.0 by the terminal user
SECPSWD	An 8-byte character format Password for validation by Connect:Enterprise	Connect:Enterprise Password entered on Panel 5.0 by the terminal user
SECRC	A 2-byte hexadecimal process completion return code value	Return code from the IPS Header field (H00RTNCD)

Understanding Security (After) Exit Usage

The purpose of the Security (After) exit is to report to the user-written exit program the result of the attempt to contact Connect:Enterprise. The SECRC return code value can be from Connect:Enterprise regarding the verification of the User ID and Password. A return code value in the range X‘0000’ through X‘7FFF’ means the connection was successful and the return code was generated by Connect:Enterprise or an associated exit program. Interpret these return codes based upon documented return codes related to Connect:Enterprise. A return code in the range X‘8000’ through X‘FFFF’ means the connection attempt failed and the return code was generated by the CICS interface. Interpret these return codes based upon documented return codes related to the CICS interface for Connect:Enterprise.

A record of failed attempts and the reason for these failures can be a useful diagnostic tool. Identifying information, such as date, time, user, and terminal, can be recorded for charge-back or auditing purposes.

Although the return code value from the user-written exit program has no affect on the subsequent CICS interface processing following the Security (After) exit, as a standard, set the EXITRC field in the exit parameter data (COMMAREA) to the assembled value of the DSECT label XRCOKAY before executing a CICS RETURN in the user-written exit program.

Using the Data Modification Exit

This exit is invoked before an interactive user request to modify data is processed. The types of data that are observed by this exit include:

- STATFLG updates on the Connect:Enterprise Batch Queues
- Connect:Enterprise Auto Connect Queue updates
- Connect:Enterprise Options Definitions updates

- ✦ CICS interface Model Library record updates
- ✦ CICS interface Help text record updates

The data modification is not performed until after invocation off the user-written exit program generates an ACCEPT return code value. If an ACCEPT return code value is not set by the user-written exit program the CICS interface does not complete the modification activity and the terminal user receives the following message:

```
Action disallowed by modification exit.
```

Data Modification Exit Parameters

Parameters passed to the Data Modification exit are addressed by the CMCIXITS DSECT in member EXITS in your CICS interface installation source library. A listing of the parameters and the content of the data fields passed to the user-written exit program is shown in the following table:

DSECT label	Description of data	Data Field content
EXITLEN	Halfword (2-byte) length of the parameter data	Assembled value of DSECT label: DMEHLEN—Help text DMEMALEN—ADD Model DMEMCLEN—CONNECT Model DMEMELEN—EXTRACT Model DMEMJLEN—User JCL Model DMESULEN—STATFLG update DMEQULEN—A/C Queue update DMEODLEN—ODF Update
EXITID	An 8-byte character format identifier stamp	The character string CMCIXITS
EXITPROG	An 8-byte character format name of linked program	Data from Modification field on Panel 1.4
EXITTYPE	A 1-byte character format type of exit indicator	Assembled value of DSECT label XMDFY
EXITRC	A 1-byte hexadecimal return code value	Assembled value of DSECT label XRCABORT
DMEDTYPE	A 1-byte character format type of data indicator	Assembled value of DSECT label: AMODEL—ADD Model CMODEL—CONNECT Model EMODEL—EXTRACT Model JMODEL—User JCL MODEL HELP—Help text ODF—ODF Update STATF—STATFLG update ACQUEUE—A/C Queue update

The remainder of the exit parameter data must be interpreted using the DMEDTYPE (type of data indicator) described. If DMEDTYPE = AMODEL, the remaining data in the DSECT is as follows:

DSECT label	Description of data	Data Field content
AFMABEF	A fixed length data area containing a single ADD model. See macro C$VSAM for field definitions of the multiple record (3 records) model entry.	An image of the complete ADD model prior to any modification
AFMAAFT	A fixed length data area containing a single ADD model. See macro C$VSAM for field definitions of the multiple record (3 records) model entry.	An image of the complete ADD model after the modification

If DMEDTYPE = CMODEL, then the remaining data in the DSECT is as follows:

DSECT label	Description of data	Data Field content
AFMCBEF	A fixed length data area containing a single CONNECT model. See macro C$VSAM for field definitions of the multiple record (3 records) model entry.	An image of the complete CONNECT model prior to any modification
AFMCAFT	A fixed length data area containing a single CONNECT model. See macro C$VSAM for field definitions of the multiple record (3 records) model entry.	An image of the complete CONNECT model after the modification

If DMEDTYPE = EMODEL, then the remaining data in the DSECT is as follows:

DSECT label	Description of data	Data Field content
AFMEBEF	A fixed length data area containing a single EXTRACT model. See macro C$VSAM for field definitions of the multiple record (3 records) model entry.	An image of the complete EXTRACT model prior to any modification
AFMEAFT	A fixed length data area containing a single EXTRACT model. See macro C$VSAM for field definitions of the multiple record (3 records) model entry.	An image of the complete EXTRACT model after the modification

If DMEDTYPE = JMODEL, then the remaining data in the DSECT is as follows:

DSECT label	Description of data	Data Field content
AFMJBEF	A fixed length data area containing a single USER JCL model. See macro C$VSAM for field definitions of the multiple record (3 records) model entry.	An image of the complete USER JCL model prior to any modification
AFMJAFT	A fixed length data area containing a single USER JCL model. See macro C$VSAM for field definitions of the multiple record (3 records) model entry.	An image of the complete USER JCL model after the modification

If DMEDTYPE = HELP, then the remaining data in the DSECT is as follows:

DSECT label	Description of data	Data Field content
AFMHBEF	A fixed length data area containing a single HELP model. See macro C$VSAM for field definitions of the multiple record (3 records) model entry.	An image of the complete HELP model prior to any modification
AFMHAFT	A fixed length data area containing a single HELP model. See macro C$VSAM for field definitions of the multiple record (3 records) model entry.	An image of the complete HELP model after the modification

If DMEDTYPE = ODF, then the remaining data in the DSECT is as follows:

DSECT label	Description data	Data Field content
ODFTYPE	A 1-byte character format type of data indicator	Assembled value of DSECT label: ODFUSEC—Security ODFUOPT—Options ODFUREM—Remotes ODFUSIGN—Signon ODFUCLST—Connect List Add/Modify ODFUCLDL—Connect List Delete ODFUCRMT—Connect List Remote ODFUCLIN—Connect List ODFUCTIM—Connect List ODFUCMOD—Connect Copy update exist ODFUCFIN—Connect Finish ODFUCCPY—Connect Copy ODFUCCAN—Connect Cancel ODFUPOOL—Pools Delete ODFUPLUN—Pools/LUName Add/Update ODFUCAL—Calendar Delete ODFUDATE—Days/Date Add/Update
ODFBEF	A fixed length data area which contains variable length data mapped by a C$Onn DSECT. See the ODFTYPE comments for the DSECT identifiers. See macro C$Onn for field definitions of the entire record layout.	An image of the specific ODF data prior to any modification
ODFAFT	A fixed length data area which contains variable length data mapped by a C$Onn DSECT. See the ODFTYPE comments for the DSECT identifiers. See macro C$Onn for field definitions of the entire record layout.	An image of the specific ODF data modification activity, (delete, add, modify) to be performed and the related (add, modify) data

If DMEDTYPE = STATF, then the remaining data in the DSECT is as follows:

DSECT label	Description of data	Data Field content
SUBEF	A fixed length data area containing a variable number of Status Flags data. See macro C$U28 for field definitions of the entire record.	An image of the Status Flags for one or more batches prior to any modification

DSECT label	Description of data	Data Field content
SUAFT	A fixed length data area containing a variable number of Status Flags data. See macro C$U28 for field definitions of the entire record.	An image of the Status Flags for one or more batches after modification

If DMEDTYPE = ACQUEUE, then the remaining data in the DSECT is as follows:

DSECT label	Description of data	Data Field content
ACQBEF	A fixed length data area containing a variable number of A/C Queue entries. See macro C$064 for field definitions of the entire record.	An image of the A/C Queue entry for one or more Auto Connects prior to any modification
ACQAFT	A fixed length data area containing a variable number of Status Flags data. See macro C$064 for field definitions of the entire record.	An image of the A/C Queue data modification action, (delete, modify) to be performed and the related (modify) data for one or more A/C Queue entries

Understanding Data Modification Exit Usage

The purpose of the Data Modification exit is to provide notification prior to a terminal user modifying any system data. You can validate the identifying information (CICS User ID if specified, or the originating terminal ID) and any of the DSECT data as authorized to be modified. You can set limits on what can be modified or determine that modifications are appropriate. Identifying information, such as date, time, user, and terminal, as well as actual data that will be modified can be recorded for charge-back or auditing purposes.

To allow the terminal user to modify system data, the EXITRC field in the exit parameter data (COMMAREA) must be set to the assembled value of DSECT label XRCOKAY before executing a CICS RETURN in the user-written exit program.

Using the Termination Exit

This exit is invoked before use of the CICS interface is terminated completely by an interactive user. It is solely a reporting function by the CICS interface to the user-written exit program. It in no way can be used to alter the processing path of the CICS interface.

Termination Exit Parameters

Parameters passed to the Termination exit are addressed by the CMCIXITS DSECT in member EXITS in your CICS interface installation source library. A listing of the parameters and the content of the data fields passed to the user-written exit program is shown in the following table:

DSECT label	Description of data	Data Field content
EXITLEN	Halfword (2-byte) length of the parameter data	Assembled value of DSECT label TERMLEN
EXITID	An 8-byte character format identifier stamp	The character string CMCIXITS
EXITPROG	An 8-byte character format name of linked program	Data from Termination field on Panel 1.4
EXITTYPE	A 1-byte character format type of exit indicator	Assembled value of DSECT label XTERM
EXITRC	A 1-byte hexadecimal return code value	Assembled value of DSECT label XRCABORT

Understanding Termination Exit Usage

The purpose of the Termination exit is to provide notification of a terminal user terminating use of the CICS interface. As the terminal user discontinues use of the CICS interface, information, such as date, time, user, and terminal, for charge-back or auditing purposes can be recorded.

The return code value from the user-written exit program has no affect on the subsequent CICS interface processing that follows the Termination exit. It is recommended that the EXITRC field in the exit parameter data, COMMAREA, be set to the assembled value of DSECT label XRCOKAY before executing a CICS RETURN in the user-written exit program.

Chapter 10

CICS User API

The User Application Programming Interface (UAPI) facilitates data exchange between a user-developed CICS transaction and the CICS interface. The data to exchange is defined in an Interface Parameter Structure (IPS).

An IPS is a collection of data for a specific task or function. The IPS describes either a function and the data required for its execution, or the results of an executed function. The IPS has two parts:

- A fixed-length header is always present. All data exchanges use this portion of the IPS.
- A variable-length trailer that is included only if an in-progress activity requires it. A field in the header portion of the IPS specifies the header length plus any present trailer portion.

This chapter describes the available UAPI functions in the CICS interface, including:

- **Wake Up Transaction**—This feature initiates a transaction or program that is specified by a user-written exit or by an Connect:Enterprise End of Batch application agent. This initiation occurs as the result of an End of Batch exit invoking a Wake Up exit or an End of Batch application agent using the WAKEUP instruction.
- **Online Batch ADD**—This user-written transaction requests that a specified Connect:Enterprise perform an ADD function. The user-written transaction delivers batch file data to the Connect:Enterprise through the CICS interface.
- **Online Batch REQUEST**—This user-written transaction contains a REQUEST for file data from a specified Connect:Enterprise. Connect:Enterprise delivers batch file data to the user-written transaction through the CICS interface.
- **Issue Connect:Enterprise Commands**—This user-written transaction requests execution of Connect:Enterprise commands and receives acknowledgments.
- **Request Directory Listing**—This user-written transaction requests information about Connect:Enterprise Batch Queue content.

Activating Interface Parameter Structure

Your coded CICS transactions and programs contain an IPS definition. This definition can be a Data Definition in COBOL, a DSECT in Assembler Language, or a Structure declaration in C.

Based on the request to be issued by the user program, you must supply specific fields in the IPS with data values. After all fields in the IPS are initialized, the IPS must be passed to the CICS interface. Either of two methods can be used to accomplish this task:

- Using a CICS Temporary Storage Queue (TSQ)
- Passing the IPS as a COMMAREA when the CICS interface is linked by your transaction.

The two methods are described below. In either method, the CICS interface determines which method was selected by the user transaction for transferring the IPS data. Return (response) IPS data is returned to the user transaction using the same method used to deliver it originally.

A user-developed UAPI transaction invokes the CICS interface by issuing an EXEC CICS LINK command from the user transaction. This LINK command specifies the program xx62002. The transaction prefix, xx, is established for all Connect:Enterprise transactions. The default is CM, which is combined with a COMMAREA. The COMMAREA specified in the EXEC CICS LINK command can be either the actual IPS or locator data (described below) used for retrieving the IPS.

Using a CICS TSQ to Pass the IPS

In certain situations, you may want to store the IPS in a CICS TSQ, and then provide data to the CICS interface for locating and retrieving the stored IPS. The IPS data to be sent to the CICS interface can be written into a TSQ of your choice.

The user-written transaction can format a control COMMAREA and pass this COMMAREA when linking to the CICS interface program. Format the locator data as described below:

- An 8-character identifier string containing the value CMCIS$CA
- A half-word reserved for the return code.
- A half-word of reserved space. Initialize to low values. Do not further modify.
- Name of the TSQ that contains the IPS to be processed. This name must be 8 characters, left-justified, padded with blanks on the right.
- A full word containing the TSQ item number that contains the IPS to be processed.

  When a response is returned to the user transaction, this field indicates the TSQ item number that contains the processed IPS.
- A full word containing the length of the IPS stored in the TSQ entry.
- 8 bytes reserved for EIBFN and EIBRCODE data in case of an unexpected error condition.
- 20 bytes of reserved space. This space must be included. Initialize to low values. Do not further modify.

When the CICS interface program returns the response IPS, it is written into the specified TSQ as the next available item number. The assigned item number is returned in the item number field of your control COMMAREA.

There is a macro named C$CTLCA in your CICS interface installation source library that describes each field required in a control COMMAREA.

Passing the IPS as a COMMAREA

In other situations, you may want to specify the actual IPS address and length in the EXEC CICS LINK command parameters. The COMMAREA passed by the user-written transaction program to the CICS interface program serves as both the send and receive buffer for the LU6.2 conversation. As a result, it must be large enough to accommodate the anticipated response to your request. The IPS data (header and trailer) is formatted as the COMMAREA and the header field H00TSLNG describes the data bytes actually sent to Connect:Enterprise.

When control returns to the user-written transaction program following the CICS LINK, your COMMAREA contains the response IPS.

Interface Parameter Structure Format

The CICS interface source library created during CICS interface installation contains three sample programs that demonstrate the functionality of a user-written API. The supplied sample programs are identified by CSECT name.

For these samples to execute correctly, you must change the following lines of code to your site-specific definitions:

1. Locate every occurrence of the label H00SNAME and change it to specify the symbolic name for your Connect:Enterprise system. Check Panel 1.3 or 1.5 if you are unsure of the symbolic name.
2. Modify each reference to H00SUSER and H00SPSWD to specify the user ID and password required by your Connect:Enterprise system. If you do not have security activated in the Connect:Enterprise system, these fields do not require changes.
3. After you modify the sample programs, assemble and link edit them into your CICS load library
4. Complete the PPT and PCT definitions. You can define the first three characters of the PCT names. However, the fourth character of the PCT names are hard coded into the samples and must be numbered from 1 to 3, as shown below:

   xxx1 CSECT APISAMP1
   xxx2 CSECT APISAMP2
   xxx3 CSECT APISAMP3

   Also, the Wake Up exit in Connect:Enterprise must initiate transaction xxx1 to start the entire sample exercise.

The sample programs simulate the following situation:

1. A CICS transaction or program is initiated and receives standard Connect:Enterprise End of Batch information.

   Transaction xxx1 is initiated in the sample exercise.
2. Using the End of Batch information, batch data is requested from Connect:Enterprise and stored by the CICS transaction or program for later processing.

Transaction xxx1 writes batch data records to a TSQ. Each batch of data is terminated by an End of Batch record. When all batch data has been received, an End Of File record is written to Temporary Storage to simulate EOF processing. The TSQ name, first and last Item Number, and other batch-related identifiers are then passed to transaction xxx2 using a START command (with a 10-second delay).

3. After related processing, another CICS transaction or program is initiated to add new batch data to Connect:Enterprise Batch Queues.

   The 10-second delay, specified by transaction xxx1, simulates related processing (that presumably has modified the batch data). Transaction xxx2 is then initiated. This transaction reads the batch data from the TSQ and adds it to the Connect:Enterprise Batch Queues. At End of Batch, Connect:Enterprise returns the new Batch Number that has been assigned.

Note: Ensure that your End of Batch exit or Wake Up exit or application agent rules in Connect:Enterprise can differentiate the batch that was initially added (to start the sample program demonstration) and the batch added by the ADD API sample program (Transaction xxx2). Failure to do so results in a continuous execution of the sample program demonstration set.

---

4. The Batch Number of the newly added batch is used in a $$CONNECT command requested by the CICS transaction or program.

   Transaction xxx2 continues execution and formats a $$CONNECT command specifying the Batch Number and identifies Transaction xxx3 as the transaction to be invoked when the Auto Connect completes.

5. When the Auto Connect completes, a CICS transaction or program is initiated and receives standard Connect:Enterprise completion messages.

6. Transaction xxx3 is invoked and simulates processing of Auto Connect completion messages. Once all messages are disposed of, the transaction terminates.

Interface Parameter Structure Content

Every IPS is designed to contain header and trailer data. These separate data groups must form a physically contiguous storage area record as shown in the following example:

Header	Trailer

Both the header and trailer portions contain required data, referred to as the fixed data. Additional data that might be present (conditionally required) in either the header or trailer is referred to as variable data.

The Source Library delivered with the CICS interface contains DSECTs that describe the IPS discussed below. Each description includes the name of the applicable Source Library member.

Review the identified DSECT while reading the following descriptions to increase your understanding of UAPI functions and IPS requirements.

IPS Header Portion Data

The header portion (source member C$H00) contains both fixed and variable data. A full header contains both fixed and variable data. A mini header contains only fixed data. The full header is required in all cases except during a batch data file transfer to Connect:Enterprise (an ADD) or from Connect:Enterprise (a REQUEST).

The following summarizes the rules applying to headers:

- Every IPS must contain a header portion.
- The first IPS in every function requires a full header.
  - Each request to ADD a batch of data.
  - Each request to retrieve (REQUEST) batch data. Multiple batches can be returned by a single REQUEST.
  - Each request to issue commands.
- All subsequent IPSs require a mini header.
  - Second through the remaining IPSs required to complete addition (ADD) of a single batch.
  - Second through the remaining IPSs required to complete receipt of all batch data resulting from a single REQUEST.

IPS Fixed Header Data

The following table describes the fixed header data in the sequence that it appears in the DSECT. The data is identified by the DSECT field label. All fields are required.

Field Label	Description
H00HLNG	An aligned half-word format field containing the length of the header portion of the IPS. The length of a full header is defined by the equated DSECT value (H00HLEN). The length of a mini header is defined by the equated DSECT value (H00MHLEN).
H00HDRID	An 8-character identifier stamp that must always contain C$H00 followed by three blanks. The stamp is the same in both types of headers.
H00HTYPE	Defines the type of header data that is present. A full header contains both fixed data and variable data. A mini header contains only fixed data.

Field Label	Description
H00REQCD	An 8-character name identifying the function requested by the initiator of the process. Generally, this request identifier is the trailer portion identifier stamp. For example, when a batch data file transfer to Connect:Enterprise (an ADD) is requested, both this field and the trailer data are identified by C$A20 followed by three blanks. Similarly, when a batch data file transfer from Connect:Enterprise (a REQUEST) is requested, both this field and the trailer data are identified by C$R20 followed by three blanks. A complete list of request codes is included later in this section.
H00RTNCD	A half-word aligned 2-byte field containing the return code from the requested function. When this field contains low values the requested function has been completed (or at least initiated) without error. If an error condition exists during execution of a requested function, the return code field contains unique values that identify both the source of the return code and the error condition. A hexadecimal value in the range X'0100-X'05FF' indicates the code was returned by Connect:Enterprise. The hexadecimal value in the range X'0400'-X'04FF' indicates the error condition was detected by a user-written exit executing in Connect:Enterprise. Character values 0x–9x (hexadecimal values F0xx–F9xx) indicate the code was returned by the CICS interface. A complete list of Connect:Enterprise or the CICS interface supplied return codes is included in *Connect:Enterprise for z/OS Messages and Codes Guide*.
H00DATA	Defines the type of data carried in the IPS. Equated values are provided for defining or interrogating the type of data field. Type of data depends on the activity being performed. A user-written API to ADD batch data to Connect:Enterprise or to REQUEST batch data from Connect:Enterprise uses the equated value H00UAPI to indicate the correct data type. A user-written API that requests issuance of a console command by Connect:Enterprise uses the equated value H00REQ to indicate the correct data type. An acknowledgment from Connect:Enterprise following any request is identified as response data (equated value H00RSP). The IPS from Connect:Enterprise does not indicate response type data when Connect:Enterprise initiates a Wake Up transaction. In this case, the data type is identified as wake-up (equated value H00WAKE) data. This type of data is explained later in this section.

Field Label	Description
H00CFCTL	Defines both the conversation flow required by the user application and the conversation condition that exists in the system. The field must be set correctly by the user each time a request function is passed to the CICS interface. The returned values must be interrogated each time control is returned by the CICS interface. If so indicated, alter subsequent processing logic to accommodate system conditions. The sample programs include examples of how this field is set and tested. The sample programs also include code to demonstrate how to process exceptional conditions. In the request IPS, indicate the processing sequence that will be executed by the CICS interface. Choices include SEND data, RECEIVE data, request CONFIRMATION of completed activity, or TERMINATE the process. Most combinations of processing sequences are allowed except CONFIRMATION which can only follow and be requested simultaneously with SEND. At this time, a user-written transaction has no requirement to issue a CONFIRMATION request and should never do so. In the response IPS, the CICS interface indicates the condition of the conversation that exists following execution of the requested flow control. Possibilities include: ◆ NODATA received (your IPS buffer is unchanged from what was passed to the CICS interface). ◆ Data received is INCOMPLETE (your IPS buffer is too small for the data to be received and a subsequent RECEIVE request must be executed to get the remaining data). ◆ END RECEIVE (you have received all possible data). ◆ CONVERSATION TERMINATED. These conditions can be present in logical combinations. The user-written API must test these conditions in every response IPS to determine if the conversation requires more attention or if it has satisfactorily completed. The condition CONVERSATION TERMINATED must be attained before the user-written API can permanently suspend CICS interface use. Failure to do so results in conversations (LU6.2 connections) to Connect:Enterprise being in an active, but unusable, condition.

Field Label	Description
	The DSECT provides a complete list of equated values that can be used to define the required or desired flow control and to interrogate the condition of the conversation. Examples of usual flow control combinations are: ◆ A user-written API to receive batch data from Connect:Enterprise normally requests combined flow control to SEND (send the request to Connect:Enterprise) then RECEIVE (receive the batch data from Connect:Enterprise). As batch data is received, a flag in the trailer data indicates first, middle, or end segment of batch data. At the end of the batch, the user-written transaction executes the end of file processing as required by the application. Following each end of file process, the user-written transaction must format an IPS containing a zero return code indicating that the batch data has been accepted or a nonzero return code indicating the batch data has been rejected. This IPS must also indicate flow control RECEIVE if the conversation condition END RECEIVE is not indicated (Connect:Enterprise has more batch data to be sent) or QUIT (eliminate the LU6.2 connection to Connect:Enterprise) if the condition END RECEIVE is indicated. The CICS interface uses this IPS and the return code value to fulfill an outstanding confirmation request from Connect:Enterprise before the requested flow control is executed. ◆ A user-written API to send batch data to Connect:Enterprise normally requests flow control to SEND (send the request and the initial batch data to Connect:Enterprise). Subsequent blocks or segments of data are sent to Connect:Enterprise by another request to SEND. All SEND requests after the initial send for each batch of data are done through a mini header. When the last data for this batch is placed into the IPS, the flow control specifies a combination of SEND (send the final data to Connect:Enterprise) then RECEIVE (receive the batch number that has been assigned by Connect:Enterprise) then QUIT (eliminate the LU6.2 connection to Connect:Enterprise). The RECEIVE (for the batch number from Connect:Enterprise) forms an implied confirmation because when a Batch Number is available, Connect:Enterprise has accepted the data and recorded it to the Batch Queues. If the batch was not accepted, a nonzero return code IPS (with a zero Batch Number) is returned by the RECEIVE. ◆ A user-written API to request that an Connect:Enterprise console command be issued normally requests a combination flow control to SEND (send the issue command request to Connect:Enterprise) then RECEIVE (receive notification that the command was either issued or that the request to issue was rejected or receive the response data) then QUIT (eliminate the LU6.2 connection to Connect:Enterprise). The flow control and conversation condition flags place the authority for the success or failure of the attempted processes under the control of the user-written API. The first request in a processing sequence must be SEND. RECEIVE must initially follow SEND or a previous RECEIVE if the condition END RECEIVE was not indicated. QUIT must be the last request made in a processing sequence.
H00IDENT	An internal control field for the CICS interface. The user-written API must initialize it to low values when the IPS is formatted prior to the first SEND request. Do not modify this field until after the QUIT request has executed.

Field Label	Description
H00TSLNG	Defines the length of the header portion plus any trailer portion that is present. The length must correspond to the length supplied in the trailer data (if any) plus the value stored in the H00HLNG field described previously. The length defines the data to be sent to Connect:Enterprise. This field does not relate to the length of the data received from Connect:Enterprise (or the maximum buffer size to be used). The receive data length maximum is defined by the length of the actual IPS data area supplied to the CICS interface as a COMMAREA (supplied through the CICS LINK) or the maximum TSQ record length.

IPS Variable Header Data

The following table describes variable header data in the sequence that it appears in the DSECT. It is identified by the DSECT field label. These fields are defined in a full header but can be used conditionally, depending on the processing activity.

Field Label	Description
H00FDBK	These 12 bytes are a collection of system-dependent error codes that can be included in the IPS to further describe a nonzero return code in the IPS. If the return code was supplied by Connect:Enterprise, feedback information indicates whether the error was VTAM or VSAM related, the severity of the VTAM errors, the identity of the failing VSAM function and the detailed error identification. If the return code was supplied by the CICS interface, feedback information contains the last CICS command executed (EIBFN), the CICS response code (EIBRCODE) and, if the error was related to LU6.2 communication activity, the CICS LU6.2 conversation error indicator (EIBERRCD). If the return code was supplied by a user-written exit executing in Connect:Enterprise, feedback information is defined or utilized by the application designer. The system does not modify this field except when setting nonzero return codes.
H00CDEFN, H00SYSID, H00CPROG or H00CTRAN, H00CTERM and H00CUSER	These fields are required when a user-written API requests that an Auto Connect be initiated by Connect:Enterprise. The initiation of the specified Auto Connect is immediately reported to the requesting user-written API. The completion of the Auto Connect is reported later (possibly hours later) to the transaction or program defined and identified by these fields. These same definition fields are used by the Connect:Enterprise Wake Up process to identify the transaction or program that is executed as the result of a Wake Up call.
H00CDEFN	Indicates if a PROGRAM is executed through an XCTL command or if a TRANSACTION is initiated through a START command. Also, this field indicates if a terminal identifier and an associated user identifier are present in related fields. If present, it further indicates whether or not the START command should specify a terminal. Generally, starting a transaction as terminal attached is not recommended. However, the capability to do so is provided to the user-written API.
H00SYSID	Indicates the system identification of the CICS system identified in the SYSID parameter of the START transaction command. When the equated value of H00TRNID is present in H00CDEFN this field is required.

Field Label	Description
H00CRESC	Generically defines the area containing the target program name and terminal identifier (if specified) or the transaction identifier and terminal identifier (if specified). This area is redefined as indicated below.
H00CPROG	If the equated value H00PGNME is present in H00CDEFN, this field (the first eight bytes of H00CRESC defined above) must contain the program (PPT) name to be executed through an XCTL command. If this option is chosen, the program to be executed must reside in the same CICS system as the CICS interface as no facility is provided to specify a CICS SYSID in the XCTL command.
H00CTRAN	If the equated value H00TRNID is present in H00CDEFN, this field (the first eight bytes of H00CRESC defined above) must contain the transaction (PCT) identifier to be initiated through a START command. Transaction ID must be left justified in this field definition. If this option is chosen, the H00SYSID specification is included in the START command.
H00CTERM	If the equated value H00TRMID or H00TRMUS is present in H00CDEFN, this field (the second eight bytes of H00CRESC defined above) must contain the terminal (TCT) identifier. Terminal ID must be left justified in this field definition. If the equated values found in H00CDEFN are H00TRNID and H00TRMID, the specified terminal identifier is included in the START transaction command. If the equated values found in H00CDEFN are H00TRNID and H00TRMUS, the START transaction command does not specify a terminal but the start data (the IPS) contains the Terminal ID and the User ID as part of the header data. If the equated value found in H00CDEFN is H00PGNME, the COMMAREA (the IPS) passed during the XCTL contains the Terminal ID (and possibly the User ID) as part of the header data.
H00CUSER	If the equated value H00TRMUS is present in H00CDEFN, this field contains the User ID to be associated with the executed resource. This field is manipulated in an application-specific manner. This field data does not affect either the XCTL or the START command. The field data is simply passed to the program or transaction as part of the COMMAREA or Start Data respectively. If this field is indicated as present (H00TRMUS present in H00CDEFN) it must not contain blanks or low values.
H00CINTV	Describes, in minutes, the maximum amount of time that can elapse between CICS LINKs to the CICS interface by the user-written API. The time interval is used to determine whether LU6.2 resources are occupied by tasks that appear to be terminated. A normal conversation occurrence (SEND or RECEIVE a buffer of data) would normally be measured in terms of seconds, not minutes. The user-written API logically pursues application-specific processing between conversation occurrences. This processing time must be estimated in minutes and indicated in this field. If the user-written API has not returned to the CICS interface in the estimated time interval, the task is considered dead and any CICS resources it is holding are released or cleaned up. This field is not fully implemented in the current version of this system.
H00SNAME	Identifies the Connect:Enterprise APPC component that is to be contacted on behalf of the user-written API. The name is a symbolic name that has been defined by the CICS interface administrator. It is not necessarily the APPLID of the Connect:Enterprise APPC component, although the administrator can assign symbolic names that match the APPLID. This symbolic name is required in every IPS formatted with a full header.

Field Label	Description
H00SUSER	A user ID defined to an external security package or a security exit executed from in Connect:Enterprise. This user ID (and the associated password) is used to validate or authorize the request defined by this IPS. This user ID is required in every IPS formatted with a full header.
H00SPSWD	Contains the user's password defined to an external security package or a security exit executed from Connect:Enterprise. This password (and the associated user ID) is used to validate or authorize the request defined by this IPS. This password is required in every IPS formatted with a full header.
H00SVRM	Contains the version, release and modification level of the target Connect:Enterprise and is included in every full header IPS sent to the user by Connect:Enterprise. This information is provided by the system for user control or validation functions. You are not required to supply this information in any IPS sent to Connect:Enterprise. The data in this 3-byte binary field, when returned to character representation, is VVRRMM where: ◆ VV = version (2 digits; 02–99) ◆ RR = release (2 digits; 00–99) ◆ MM = modification (2 digits; 00–99)
H00NPSW	Contains a new password that is assigned to the user after confirmation of the current password.

IPS Trailer Portion Data

Every request directed to Connect:Enterprise requires a trailer portion to complete the IPS. The trailer portion must be physically adjacent to the last byte of the header portion. The format of every trailer is divided into fixed and variable data portions.

Fixed trailer data varies from one DSECT to another; however every trailer DSECT begins with a half-word aligned half-word format length field followed by an 8-character identifier stamp. This stamp contains the trailer identifier C$xxx followed by blanks. The xxx characters in the trailer identifier define the purpose or source of the IPS trailer data.

Beginning with Connect:Enterprise for z/OS Version 1.4, certain IPS trailer layouts were increased in size to accommodate the expanded User Batch ID (BID64). Each of these macro definitions can be generated with the following layouts (FORMAT=1 and FORMAT=2).

- C$xxx FORMAT=1 which generates the IPS trailer used when communicating with a pre-1.4 Connect:Enterprise for z/OS system
- C$xxx FORMAT=2 which generates the IPS trailer used when communicating with a 1.4 or later version of Connect:Enterprise for z/OS

Your program must build the correct trailer format to properly communicate with the corresponding version of Connect:Enterprise for z/OS.

Following is a list of the C$xxx macro definitions, which must be specified with either FORMAT=1 or 2, depending on which release level of Connect:Enterprise for z/OS your application is communicating with. All of these macro definitions default to FORMAT=1.

- C$O03—AUTO CONNECT INITIATION REQUEST
- C$O29—ACTIVE SESSION DETAIL DISPLAY OF REMOTE AC/RC

- C$O34—ACTIVE A/C REMOTE SUMMARY DISPLAY
- C$O43— REMOTES LIST DISPLAY
- C$O44—REMOTES UPDATE
- C$U06—AUTO CONNECT DETAIL REQUEST
- C$U07—AUTO CONNECT DETAIL SELECTION LIST DATA
- C$U071—QUEUED AUTO CONNECT DISPLAY
- C$U12—REMOTE CONNECT DETAIL REQUEST
- C$U13—REMOTE CONNECT DETAIL DISPLAY
- C$U15—BATCH QUEUE DIRECTORY REQUEST
- C$U16—BATCH QUEUE DIRECTORY LIST
- C$U28— BATCH STATUS FLAGS REQUEST
- C$W00—C:E ISSUES "WAKE-UP" CALL AT THE COMPLETION OF A SPECIFIED ACTIVITY

The following table describes several common trailer identifiers. For a complete listing of IPS trailers, see Appendix A, *IPS Trailers*.

Identifier	Description
C$W00	Identifies the trailer, built by Connect:Enterprise, to be passed to a user-written CICS program or transaction that is initiated by the CICS interface, triggered by an End of Batch exit requesting that a Wake Up exit be executed in the Connect:Enterprise APPC component. This process is known as the Wake Up transaction and is discussed in greater detail later in this section. **Note:** The C$W00 and C$U15 macros must be invoked with FORMAT=2 when communicating with a Connect:Enterprise for z/OS Version 1.4 or later release, and with FORMAT=1 when communicating with a release prior to Version 1.4. When FORMAT=2 is specified, the W00FMTID and U15FMTID fields must contain c'F2'. For more upgrading information, see the *Connect:Enterprise for z/OS Release Notes*.
C$A20	Identifies the trailer, built by the user-written API, to request batch data be ADDed to Connect:Enterprise and to supply the batch data records. Additionally, if encryption of the batch data records is desirable, the encryption controls (such as method and key) are included as the first data record in the ADD request trailer. The CICS interface uses this encryption information to process all subsequent batch data during the ADD. The encryption control information is not sent to Connect:Enterprise. When data is encrypted, it must be decrypted using the same control information to be usable. Connect:Enterprise uses the same trailer format (with different variable data) to return the batch number when the ADD is successfully completed.

Identifier	Description
C$R20	Identifies the trailer, built by the user-written API, to REQUEST batch data from Connect:Enterprise. Additionally, if batch data record decryption is required, the decryption controls (such as method and key) are included in the REQUEST request trailer. The CICS interface uses this decryption information to process all batch data received during the REQUEST. The decryption control information is not sent to Connect:Enterprise. When data is decrypted, it must have been encrypted using the same control information to be usable. The batch data is returned by Connect:Enterprise using the same trailer format with different variable data.
C$O03	Identifies the trailer, built by the user-written API, to request Connect:Enterprise to initiate an Auto Connect. The acknowledgment (response) from Connect:Enterprise indicating that the Auto Connect was initiated is returned to the user-written API as only a return code value in the header portion of the IPS. No trailer data is present in this response IPS.
C$O02	Identifies the trailer, built by Connect:Enterprise, to be passed to a user-written CICS program or transaction that is initiated by the CICS interface, triggered by the completion of an Auto Connect that was initiated earlier through an IPS with the C$O03 trailer discussed above.
C$O66	Identifies the trailer, built by the user-written API, to request Connect:Enterprise to issue a $$LIST FILES command and return a list of the files that meet the specified criteria. C$O67 identifies the trailer that contains the response (file status listing) information.
C$U15	Identifies the trailer, built by the user-written API, to request Connect:Enterprise to read through the Batch Queues and return a listing of all batches that meet the specified criteria. C$U16 identifies the trailer, built by Connect:Enterprise, that contains the response information. **Note:** The C$W00 and C$U15 macros must be invoked with FORMAT=2 when communicating with a Connect:Enterprise for z/OS Version 1.4 or later release, and with FORMAT=1 when communicating with a release prior to Version 1.4. When FORMAT=2 is specified, the W00FMTID and U15FMTID fields must contain c'F2'. For more upgrading information, see the *Connect:Enterprise for z/OS Release Notes.*

A list of all IPS Trailers is in Appendix A, *IPS Trailers*.

Using the Wake Up Transaction

Wake Up transactions can be generated in Connect:Enterprise to contact a CICS system in two ways: user-written exits or an application agent.

A user-written exit, such as the Wake Up exit, in Connect:Enterprise can contact a CICS system whenever an End of Batch has occurred. The normal Connect:Enterprise End of Batch exit must be invoked to request that the Wake Up exit also be invoked. See Chapter 4, *Using Connect:Enterprise Online Exits*. The user-written Wake Up exit defines which CICS system to contact and whether a program or transaction is to be initiated. This information is carried in the header portion (see

H00CDEFN and related fields) of the IPS that is transmitted from Connect:Enterprise to the CICS interface. The header on this IPS is a full header. (See *Interface Parameter Structure Content* on page 206 for a description of header data.)

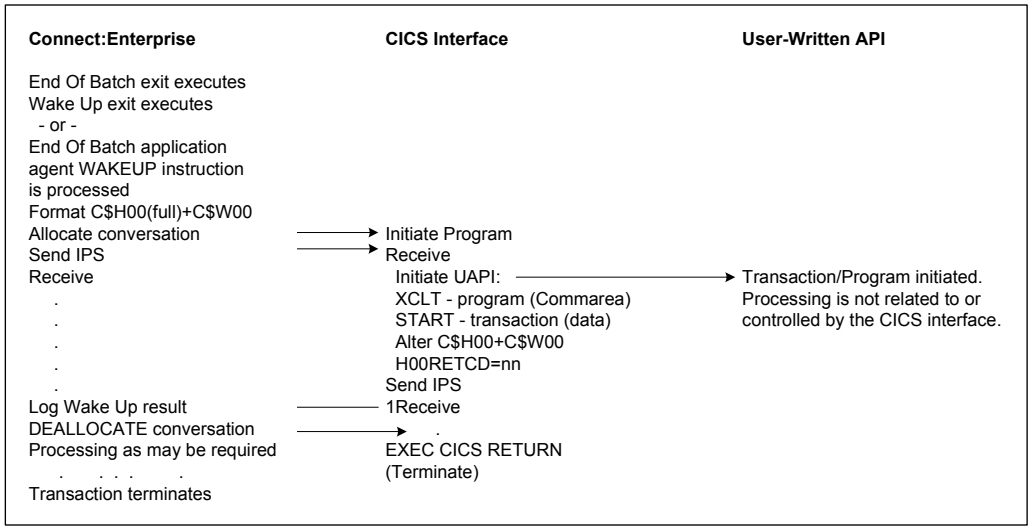
Usually the End of Batch application agent WAKEUP can be used to provide an easier method of implementing a CICS wake up. See Chapter 1, *Overview of Connect:Enterprise Application Agents* for implementation details. The application agent builds the IPS that is transmitted from Connect:Enterprise to the CICS, using parameters you supply in the End of Batch application agent rules.

Transaction Execution

The IPS is formatted, either by the user-written Wake Up exit and Connect:Enterprise or by the user-supplied End of Batch application agent rules, and transmitted to the CICS interface. The user defined program or transaction in the specified CICS system is then initiated by the CICS interface. If a program is initiated, the complete IPS (both header and trailer) is passed as a COMMAREA specified in the XCTL command. If a transaction is initiated, the complete IPS is passed as DATA in the START command.

The CICS interface then notifies Connect:Enterprise regarding the success or failure of the task initiation and terminates. The user-written program or transaction is not related to or controlled by the CICS interface after it is initiated.

The following diagram depicts the events that occur when a Wake Up transaction is requested through the End of Batch exit. Understanding the steps that the user completes as well as the order in which these steps are completed will help you understand the Wake Up transaction processing flow.



Trailer Data

The following table describes the trailer (source member C$W00) data in the order that it appears in the DSECT. It is identified by the DSECT field label.

Field	Description
W00FMTID	An identifier that must contain c'F2'. This field is only present when the source macro is generated using C$W00 FORMAT=2.
W00TLNG	A half-word aligned half-word format field containing the length of the trailer portion of the IPS. The length of this trailer is constant as it contains only fixed data.
W00TRLID	An 8-character identifier stamp that must always contain C$W00 followed by three blanks.
W00KEYID, W00BNO, W00UBID, W00BLKCT, W00ERCL, W00EOB@, W00FLAG1, W00FLAG2, W00CTIME, W00CDATE, W00OPRID, W00RTYPE and W00FILL	These fields are standard Connect:Enterprise End of Batch exit information fields. See the *Using the End of Batch Exit* on page 144 for a description and use of this data.

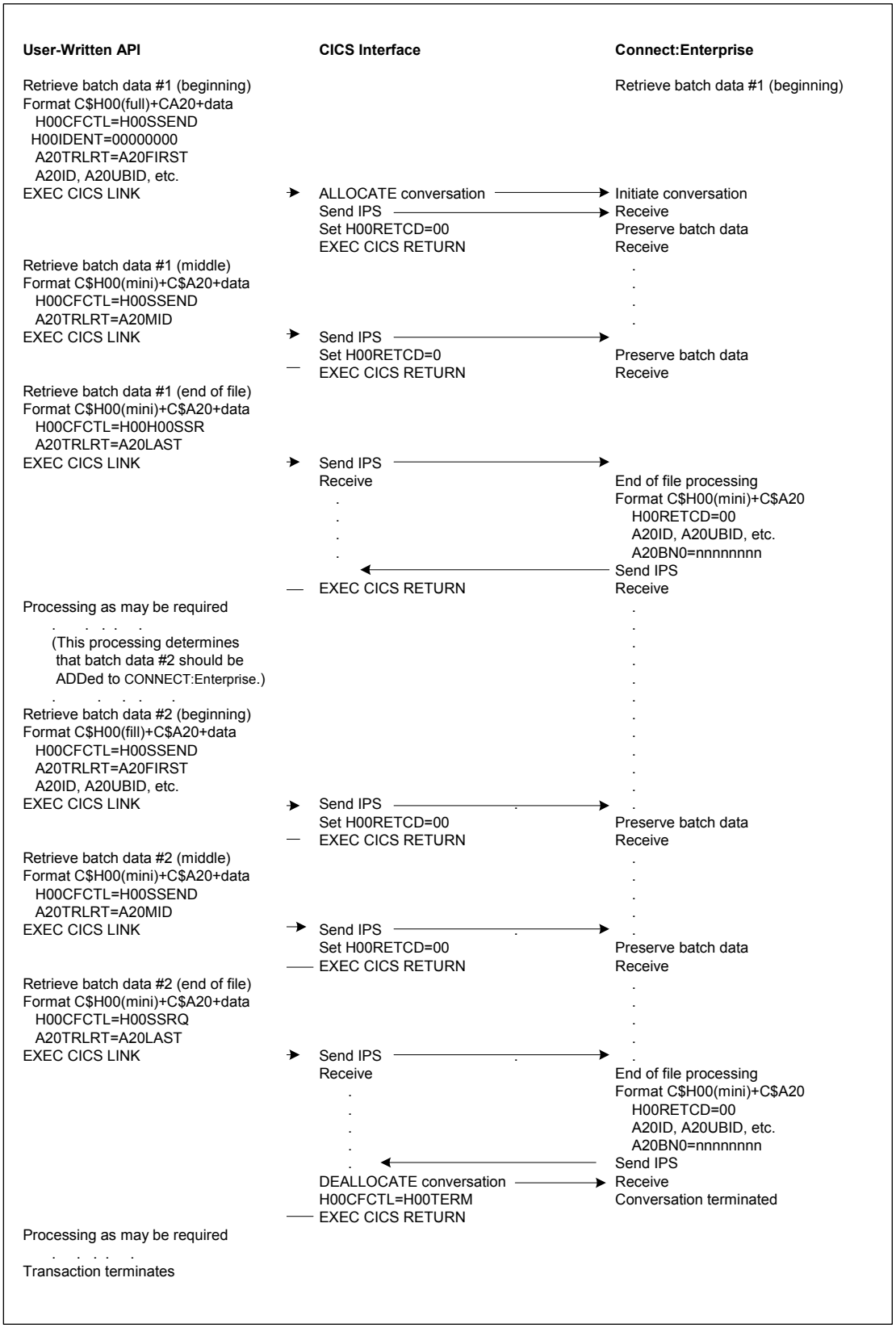
Using the ADD Transaction

This facility enables a user-written transaction to add batch data to the Connect:Enterprise Batch Queues. The header on the initial IPS for each batch is a full header. If the volume of batch data requires more than one IPS, each subsequent IPS must be formatted with a mini header. See *Interface Parameter Structure Content* on page 206 for a description of header data fields and content.

Transaction Execution

All IPSs are formatted by the user-written API and passed to the CICS interface through a CICS LINK command. The IPS is then transmitted (using LU6.2 facilities) to Connect:Enterprise. The transmission flow control specified in the header is executed by the CICS interface. If the flow control flag indicates only SEND (indicating more user batch data remains to be sent to Connect:Enterprise), the CICS interface transmits the IPS to Connect:Enterprise and informs the user-written API regarding the success or failure of the transmission. When the flow control flag indicates SEND then RECEIVE (indicating the final batch data has been formatted into the IPS) the CICS interface transmits the IPS to Connect:Enterprise and waits for the response. The Connect:Enterprise response containing the batch number assigned when the batch data was recorded on the Batch Queues is returned to the user-written API. Additional processing can be requested or the connection to Connect:Enterprise can be terminated.

The following diagram shows the events that occur when a user-written API initiates an ADD transaction to add batch data to the Connect:Enterprise batch Queues. Understanding the steps that the user must complete as well as the order in which these steps are completed will help you understand the ADD transaction processing flow. Two batches of data are shown being added to Connect:Enterprise. Notice that the flow control flag (H00CFCTL) is specified each time the CICS interface is LINKed.


User-Written API
CICS Interface
Connect:Enterprise
Retrieve batch data #1 (beginning)
Format C$H00(full)+CA20+data
H00CFCTL=H00SSEND
H00IDENT=00000000
A20TRLRT=A20FIRST
A20ID, A20UBID, etc.
EXEC CICS LINK
ALLOCATE conversation
Initiate conversation
Send IPS
Receive
Set H00RETCD=00
Preserve batch data
EXEC CICS RETURN
Receive
Retrieve batch data #1 (middle)
Format C$H00(mini)+C$A20+data
H00CFCTL=H00SSEND
A20TRLRT=A20MID
EXEC CICS LINK
Send IPS
Set H00RETCD=0
Preserve batch data
EXEC CICS RETURN
Receive
Retrieve batch data #1 (end of file)
Format C$H00(mini)+C$A20+data
H00CFCTL=H00H00SSR
A20TRLRT=A20LAST
EXEC CICS LINK
Send IPS
Receive
End of file processing
Format C$H00(mini)+C$A20
H00RETCD=00
A20ID, A20UBID, etc.
A20BN0=nnnnnnnn
Send IPS
EXEC CICS RETURN
Receive
Processing as may be required
(This processing determines
that batch data #2 should be
ADDed to CONNECT:Enterprise.)
Retrieve batch data #2 (beginning)
Format C$H00(fill)+C$A20+data
H00CFCTL=H00SSEND
A20TRLRT=A20FIRST
A20ID, A20UBID, etc.
EXEC CICS LINK
Send IPS
Set H00RETCD=00
Preserve batch data
EXEC CICS RETURN
Receive
Retrieve batch data #2 (middle)
Format C$H00(mini)+C$A20+data
H00CFCTL=H00SSEND
A20TRLRT=A20MID
EXEC CICS LINK
Send IPS
Set H00RETCD=00
Preserve batch data
EXEC CICS RETURN
Receive
Retrieve batch data #2 (end of file)
Format C$H00(mini)+C$A20+data
H00CFCTL=H00SSRQ
A20TRLRT=A20LAST
EXEC CICS LINK
Send IPS
Receive
End of file processing
Format C$H00(mini)+C$A20
H00RETCD=00
A20ID, A20UBID, etc.
A20BN0=nnnnnnnn
Send IPS
DEALLOCATE conversation
Receive
H00CFCTL=H00TERM
Conversation terminated
EXEC CICS RETURN
Processing as may be required
Transaction terminates

Fixed Trailer Data

The following table describes trailer (source member C$A20) data in the sequence that it appears in the DSECT. The data is identified by the DSECT field label.

Field	Description
A20TLNG	A half-word aligned half-word format field containing the length of the trailer portion of the IPS. The length of this trailer is variable as it contains both fixed data and variable data (user-defined records).
A20TRLID	An 8-character identifier stamp that must always contain C$A20 followed by three blanks.
A20TRLRT	A flag byte that defines the content of the trailer in terms of its relationship to the entire batch data file. In the initial, IPS (for each batch) this flag must be set to the equated value A20FIRST to indicate the first trailer of file data. If the entire batch data file is contained in the initial IPS the flag must be set to A20ONLY. As the transfer of batch data proceeds, this flag byte must be set to A20MID for each intermediate IPS and when the final batch data is formatted into the IPS, this flag byte must be set to A20LAST.
A20DDSP	A half-word aligned half-word format field containing the displacement (number of bytes) from the first byte of the C$A20 trailer (label A20TLNG) to the first byte of the first user data length (label A20LEN) field. Use of this displacement field allows the trailer format to change without requiring reassembly of user programs that choose not to use the new trailer fields. When the initial C$A20 trailer is built, users are encouraged to provide the amount of reserved space as indicated at the end of the DSECT labeled A20DSECT.

Initial Variable Trailer Data

Variable trailer data is described below in the sequence it appears in the DSECT. It is identified by the field label from the DSECT.

The following table describes the variable fields in the initial (or only) trailer to pass control information to Connect:Enterprise. These fields are also present in the response IPS (which contains the Batch Number) returned by Connect:Enterprise when the batch data is added successfully to the Batch Queue.

Field	Description
A20ID	The Mailbox ID assigned to your site by host site operations.
A20UBID	The free-form user batch ID. The maximum length is 24 bytes.
A20XMIT	The flag byte that controls whether the batch data is available only for host site operations or whether it is available for transmission to other remote sites.
A20MULTX	The flag byte that controls whether the batch data is to be marked T once it is transmitted to a remote site, making it unavailable for additional transmission unless requested by Batch Number.
A20TONCE	The flag byte that allows transmission of batch data to occur only once.

Field	Description
A20EONCE	The flag byte that allows batch data to be extracted only once.
A20BNO	The Batch Number, assigned by Connect:Enterprise, when the batch data file addition has been successfully completed.
A20UBID64	The free-form user batch ID. The maximum length is 64 bytes.
A20VBQ#	The target VSAM Batch Queue Number. You can specify the VBQ to which the batch is added or you can allow the batch to be added to the Current Collection Queue. To indicate the Current Collection Queue, set this field to low values (hex zeros). To indicate a specific VBQ, the VBQ must be defined and currently allocated to Connect:Enterprise. Place the VBQ identifier number (as a binary value, displayed in hex below) in this field. Identifier numbers are: **VBQ Specify** 01 x‘01’ 02 x‘02’ 03x‘03’ 04 x‘04’ 05 x‘05’ 06 x‘06’ 07 x‘07’ 08 x‘08’ 09 x‘09’ 10 x‘0A’ 11 x‘0B’ 12 x‘0C’ 13 x‘0D’ 14 x‘0E’ 15 x‘0F’ 16 x‘10’ 17 x‘11’ 18 x‘12’ 19 x‘13’ 20 x‘14’

Using the Reserved Area

The reserved area is subject to change. For best results, allocate the specified amount of reserved space. Use the field A20DDSP to locate the first entry of user data.

Variable Trailer Data

User data is present in every IPS that is transmitted to Connect:Enterprise. In an initial, or only, trailer the user batch data immediately follows the initial variable trailer data fields just described above. In all other trailers the user batch data immediately follows the fixed data fields.

The following table describes the variable trailer data:

Field	Description
A20LEN	A half-word aligned half-word format field containing the length of the user data that follows plus the length of this length field. For example, if a user data record of 80 characters follows, the value in this field must be 82. This field allows you to format multiple records (or record segments) in an IPS, completely under the control of the user-written API.
A20DATA	DSECT label that locates the first byte of user data for this record (or record segment). Multiple records can be blocked into an IPS. The complete length of the trailer data (A20TLNG described above) must include the length of every data record (A20DATA) plus two bytes for every data record length (A20LEN) plus the length of the fixed and any variable data.

Batch Data Encryption

When batch data is encrypted before transmission to Connect:Enterprise, the encryption request and control information must be passed to the CICS interface as the first user data in the first C$A20 trailer. The A20LEN field must be set to include the length of the encryption control information plus two bytes for the length itself, just as is done for any other user data entry. The following table shows how to format the user data (A20DATA) entry containing the encryption control information:

Field	Description
A20EID1	A 5-character identifier stamp that must always contain C:EAE (Connect:Enterprise ADD encryption) to indicate that data encryption is required.
A20EID2	The sixth character of the identifier stamp field and must always contain the value assigned to the label A20FIRST to further validate the request for encryption.
A20EID3	The last two bytes of the identifier stamp field. This field must always contain the same value as A20DDSP. This displacement value locates the user data length (A20LEN) field for this user data entry.
A20EACT	A flag byte that concludes the positive identification that this user data is a request for encryption of the batch data that follows. This field must always contain the value assigned to the label A20ENCR.
A20ETYP	A single byte method code that describes the method of encryption that is invoked. Comments in the A20DSECT adjacent to this field define the supported methods. This field must always contain a valid encryption method code.
A20EKYL	A half-word aligned half-word format field containing the length of the encryption key data. Unlike other length fields, this field contains the length of the actual data only, not length of the data plus two bytes. Encryption key data can be from 1 to 16,383 bytes in length. If the batch data is decrypted using the Connect:Enterprise system standard utility program, the encryption key length must be eight bytes.

Field	Description
A20EKEY	This DSECT label locates the first byte of the user supplied encryption key data. The length of this data is defined by the value in A20EKYL. If the batch data is decrypted using the Connect:Enterprise standard utility program, the encryption key data must be eight bytes in length, left justified, padded on the right with blanks. No encryption control information is transmitted to Connect:Enterprise. The encryption key is stored within the CICS interface in an encrypted format to protect the integrity of the encryption process.

Using the REQUEST Transaction

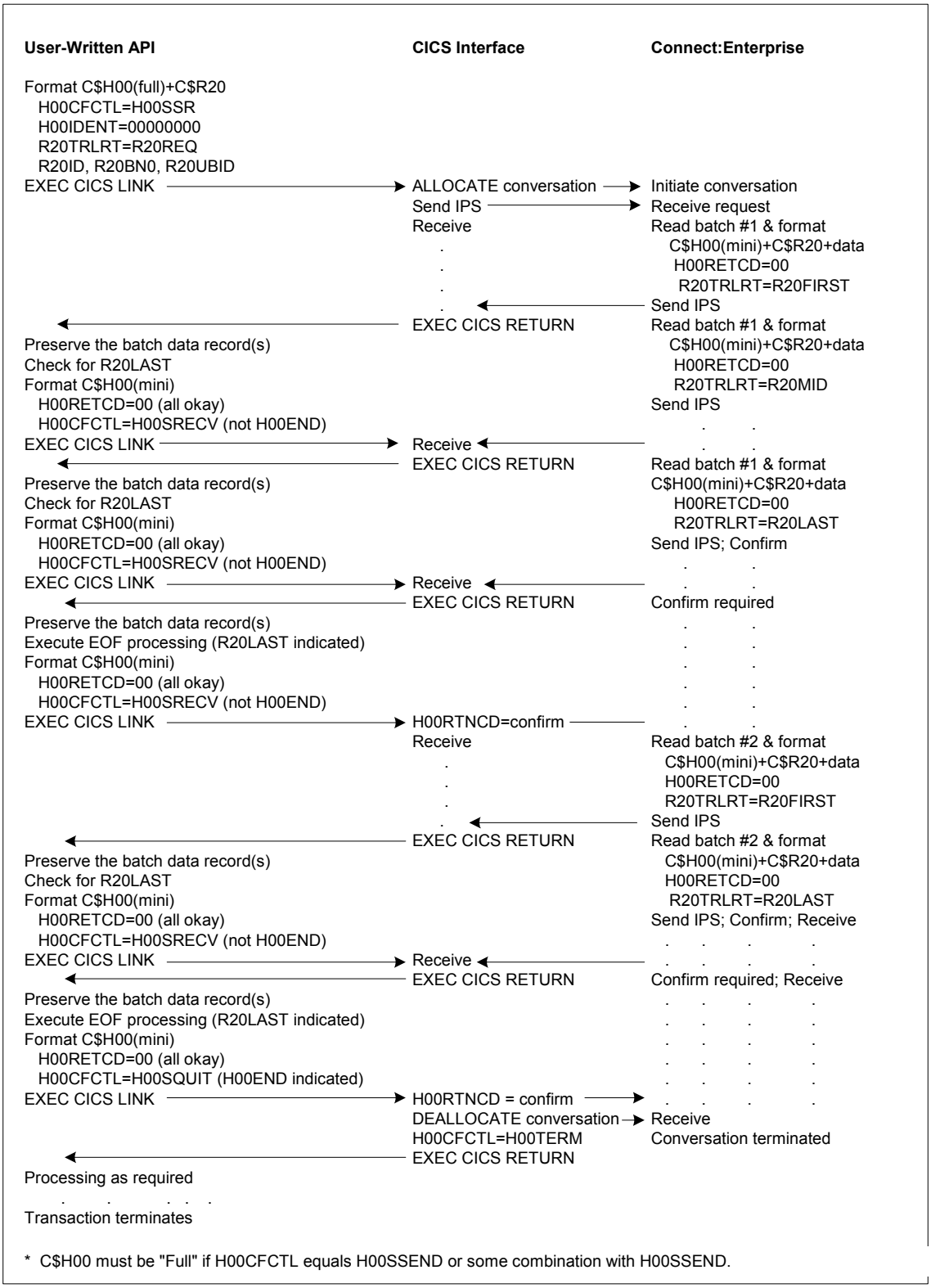
This facility allows a user-written transaction to retrieve batch data from the Connect:Enterprise Batch Queues. Header data is omitted from the following discussion. The header on the initial IPS is a full header. The initial IPS on each REQUEST contains a flow control flag that indicates SEND (H00SSEND) or SEND in combination with another flow control command, generally RECEIVE (H00SSR). All IPSs returned with batch data by Connect:Enterprise are formatted with a mini header. See the previous discussion for clarification of header data fields and content.

All IPSs are formatted by the user-written API and passed to the CICS interface through a CICS LINK command. The request (initial) IPS is then transmitted (using LU6.2 facilities) to Connect:Enterprise. The flow control flag in the request IPS must be SEND then RECEIVE. This flow control flag setting indicates that the request is sent to Connect:Enterprise and a response (batch data) is returned by Connect:Enterprise. The CICS interface transmits the IPS to Connect:Enterprise and waits for the response. The response from Connect:Enterprise contains batch data records from the requested batch.

After processing the batch data record(s), the user-written API must set H00RETCD to indicate the acceptance (RC= zero) or rejection (RC= nonzero) of the batch data just processed. This return code value is used by the CICS interface to determine whether processing continues and to provide confirmation to Connect:Enterprise for each end of file that is encountered during batch data extraction.

After setting the return code, the user-written API must check the flow control flag for H00END. If the flow control flag does not indicate H00END, the new flow control flag must indicate RECEIVE (H00SRECV) to receive the remaining batch data. If the flow control flag indicates H00END, the new flow control flag must indicate SEND (H00SSEND or in some combination) to initiate additional processing or QUIT (H00SQUIT) to cause the connection to Connect:Enterprise to terminate. The completed IPS is again given to the CICS interface through a CICS LINK command. The CICS interface executes the flow control flag explicitly to receive additional data or provide confirmation at end of file (using the H00RTNCD value) and then initiate additional processing or terminate the connection to Connect:Enterprise.

The following diagram shows the events that occur when a user-written API initiates a REQUEST transaction to retrieve batch data from the Connect:Enterprise Batch Queues. Understanding the steps that the user must complete as well as the order in which the steps are completed will help you understand the REQUEST transaction processing flow. In the following diagram, two batches of data are returned by Connect:Enterprise. Notice that the flow control flag (H00CFCTL) is specified each time the CICS interface is LINKed.


User-Written API
CICS Interface
Connect:Enterprise
Format C$H00(full)+C$R20
H00CFCTL=H00SSR
H00IDENT=00000000
R20TRLRT=R20REQ
R20ID, R20BN0, R20UBID
EXEC CICS LINK
ALLOCATE conversation
Initiate conversation
Send IPS
Receive request
Receive
Read batch #1 & format
C$H00(mini)+C$R20+data
H00RETCD=00
R20TRLRT=R20FIRST
Send IPS
EXEC CICS RETURN
Read batch #1 & format
C$H00(mini)+C$R20+data
H00RETCD=00
R20TRLRT=R20MID
Send IPS
Preserve the batch data record(s)
Check for R20LAST
Format C$H00(mini)
H00RETCD=00 (all okay)
H00CFCTL=H00SRECV (not H00END)
EXEC CICS LINK
Receive
EXEC CICS RETURN
Read batch #1 & format
C$H00(mini)+C$R20+data
H00RETCD=00
R20TRLRT=R20LAST
Send IPS; Confirm
Preserve the batch data record(s)
Check for R20LAST
Format C$H00(mini)
H00RETCD=00 (all okay)
H00CFCTL=H00SRECV (not H00END)
EXEC CICS LINK
Receive
EXEC CICS RETURN
Confirm required
Preserve the batch data record(s)
Execute EOF processing (R20LAST indicated)
Format C$H00(mini)
H00RETCD=00 (all okay)
H00CFCTL=H00SRECV (not H00END)
EXEC CICS LINK
H00RTNCD=confirm
Receive
Read batch #2 & format
C$H00(mini)+C$R20+data
H00RETCD=00
R20TRLRT=R20FIRST
Send IPS
EXEC CICS RETURN
Read batch #2 & format
C$H00(mini)+C$R20+data
H00RETCD=00
R20TRLRT=R20LAST
Send IPS; Confirm; Receive
Preserve the batch data record(s)
Check for R20LAST
Format C$H00(mini)
H00RETCD=00 (all okay)
H00CFCTL=H00SRECV (not H00END)
EXEC CICS LINK
Receive
EXEC CICS RETURN
Confirm required; Receive
Preserve the batch data record(s)
Execute EOF processing (R20LAST indicated)
Format C$H00(mini)
H00RETCD=00 (all okay)
H00CFCTL=H00SQUIT (H00END indicated)
EXEC CICS LINK
H00RTNCD = confirm
DEALLOCATE conversation
Receive
H00CFCTL=H00TERM
Conversation terminated
EXEC CICS RETURN
Processing as required
Transaction terminates
* C$H00 must be "Full" if H00CFCTL equals H00SSEND or some combination with H00SSEND.

Fixed Trailer Data

The following table describes trailer (source member C$R20) data in the sequence it appears in the DSECT. The data is identified by the DSECT field label:

Field	Description
R20TLNG	A half-word aligned half-word format field containing the length of the trailer portion of the IPS. The length of this trailer is variable as it contains both fixed data and variable data (user-defined records).
R20TRLID	An 8-character identifier stamp that must always contain C$R20 followed by three blanks.
R20TRLRT	A flag byte that defines the content of the trailer as it relates to the entire batch data file. In the initial IPS (for each batch) this flag must be set to the equated value R20REQ indicating that this trailer carries the identifiers of the batch file for Connect:Enterprise to return. When Connect:Enterprise formats the subsequent IPS, this flag byte is set to R20ONLY if the entire batch data file is contained in the initial response IPS. If the volume of batch data exceeds the capacity of the initial IPS this flag is set to R20FIRST. As the transfer of batch data proceeds, this flag byte is set to R20MID for each intermediate IPS and when the final batch data is formatted into the IPS, this flag byte is set to R20LAST. When this flag indicates R20LAST and the final batch data in this IPS has been processed, the user-written API performs end of file processing for the output file. The end of file indication does not necessarily mean that Connect:Enterprise has completed sending batch data. Test the flow control flag for H00END to determine whether more batch data is being sent by Connect:Enterprise.
R20DDSP	A half-word aligned half-word format field containing the displacement (number of bytes) from the first byte of the C$R20 trailer (label R20TLNG) to the first byte of the first user data length (label R20LEN) field. This displacement field allows the trailer format to change without requiring reassembling user programs that did not use the new trailer fields. When the initial C$R20 trailer is built that includes decryption control information, users are encouraged to provide the amount of reserved space as indicated at the end of the DSECT labeled R20DSECT. When the initial C$R20 trailer is built that does not include decryption control information, this field is set to low values.

Initial Variable Trailer Data

The following table describes variable trailer data is in the sequence that it appears in the DSECT. The data is identified by the DSECT field label. The variable fields are defined in the initial (or request) trailer to supply selection criteria to Connect:Enterprise.

Field	Description
R20ID	The Mailbox ID assigned to your site by host site operations.
R20BNO	The Batch Number that was assigned by Connect:Enterprise when the batch data was originally added to the Batch Queues.

Field	Description
R20UBID	The free-form user batch ID; maximum length is 24 bytes When supplying the selection criteria described above, the following are true: ◆ Mailbox ID (R20ID) must always be specified. ◆ When only Mailbox ID (R20ID) is specified, all batches for the ID are returned. Specify R20ONEB=Y to limit the batch data returned to only the first batch found. ◆ If both Mailbox ID and user batch ID (R20UBID) are specified, all batches for the ID that match the specified user batch ID are returned. Specify R20ONEB=Y to limit the batch data returned to only the first batch found. ◆ If both Mailbox ID and Batch Number (R20BNO) are specified, a single batch, if found, is returned. ◆ If both user batch ID and Batch Number are specified, an error is returned.
R20UBID64	The free-form user batch ID. The maximum length is 64 bytes.
R20XRLEN	A half-word aligned half-word format field containing the record length to use when deblocking transparent BSC batch data. This field is valid only when deblocking is requested, the data is indicated as transparent and the batch is BSC data. All output records returned to the user application are the length specified by this field.
R20ONEB	A flag byte that can be used to limit the number of batches that are returned if more than one batch exists for the specified selection criteria. If R20ONEB=Y is not specified and multiple batches exist, all batches that match the selection criteria are returned to the user application.
R20DEBLK	A flag byte that can be used to request deblocking of batch data by Connect:Enterprise prior to transmitting it to the CICS interface. Deblocking is allowed for BSC and SNA batches. Batch data that has been encrypted (by the system standard utility program or another UAPI) cannot be deblocked.
R20PCC	A flag byte that specifies the method of handling BSC Print Carriage Control ESC sequences that can be in batches from remote sites. These characters can be present in some print files created for printing at the remote site. You can remove, keep or convert these special characters. See the PCC (Print Carriage Control) parameter in the appendix listing the offline utility parameters in *Connect:Enterprise for z/OS User's Guide* for a description of PCC processing options and results.

Using the Reserved Area

The reserved area is subject to change. When including decryption data in the C$R20 trailer, allocate the specified amount of reserved space. Use the field R20DDSP to locate the user data.

Variable Trailer Data

User data is present in every IPS that is returned from Connect:Enterprise and immediately follows the fixed data fields. The following table describes the data:

Field	Description
R20LEN	A half-word aligned half-word format field containing the length of the user data that follows plus the length of this length field. For example, if a user data record of 80 characters follows, the value in this field must be 82. This field allows you to format multiple records (or record segments) into an IPS, depending on the original format of the user data.
R20DATA	This DSECT label locates the first byte of user data for this record (or record segment). Multiple records can be blocked in an IPS. The complete length of the trailer data (R20TLNG described above) includes the length of every data record (R20DATA) plus two bytes for every data record length (R20LEN) plus the length of the fixed data in the trailer.

Batch Data Decryption

When batch data must be decrypted after receipt from Connect:Enterprise, the decryption request and control information must be passed to the CICS interface as user data in the initial C$R20 trailer built by the user-written API. The R20DDSP field must be set to locate the user data as explained previously. The R20LEN field must be set to include the length of the decryption control information plus two bytes for the length itself.

The following table describes how to format the user data (R20DATA) entry containing the decryption control information:

Field	Description
R20DACT	A flag byte that must always contain the value assigned to the label R20DECR.
R20DTYP	A single byte method code that describes the method of decryption that is invoked. Comments in the R20DSECT adjacent to this field define the supported methods. This field must always contain a valid decryption method code.
R20DKYL	A half-word aligned half-word format field containing the length of the decryption key data. Unlike other length fields, this field contains the length of the actual data only, not length of the data plus two bytes. Decryption key data can be from 1–16,383 bytes in length. If the batch data was encrypted using the Connect:Enterprise system standard utility program, the decryption key length must be eight bytes.
R20DKEY	This DSECT label locates the first byte of the user supplied decryption key data. The length of this data is defined by the value in R20DKYL. If the batch data was encrypted using the Connect:Enterprise system standard utility program, the decryption key data must be eight bytes in length, left justified, padded on the right with blanks.

No decryption control information is transmitted to Connect:Enterprise. The decryption key is stored within the CICS interface in an encrypted format to protect the integrity of the decryption process.

Connect:Enterprise Command Transactions

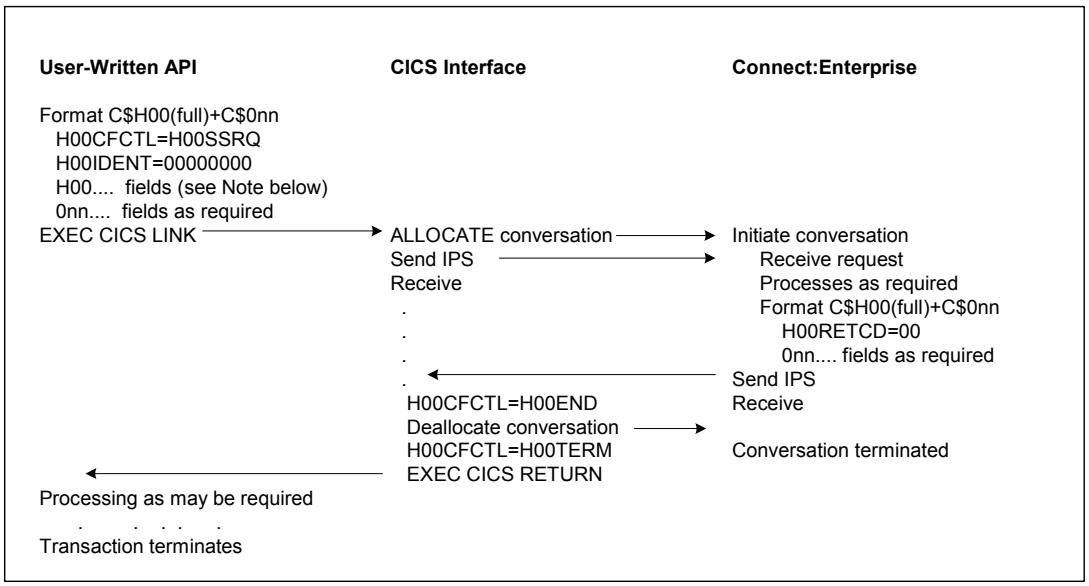
This facility allows a user-written API to issue a subset of the console commands that are available in Connect:Enterprise. The commands that are supported and the trailer DSECT source member identifiers for each are:

- Initiate an Auto Connect (source member C$O03)
  - Auto Connect completion messages (source member C$O02)
- Request a SNAP dump (source member C$O05)
- Request List status of Traces, BSC lines, SNA sessions, Auto Connect Queue, or a combination list of all except the A/C Queue (source member C$O07)
  - Traces Status Display (source member C$O08)
  - BSC Lines Status Display (source member C$O09)
  - SNA Sessions Status Display (source member C$O13)
  - Traces/BSC Lines/SNA Sessions Status Display (source member C$O15)
  - Auto Connect Queue Status Display (source member C$O17)
- Request Connect:Enterprise shutdown (source member C$O16)
- Restart a closed BSC line (source member C$O19)
- Stop an Auto Connect or Remote Connect (source member C$O21)
- Start/Stop Traces (source member C$O23)
- Request an Connect:Enterprise System Files listing (source member C$O66)
  - Connect:Enterprise System Files listing (source member C$O67)
- Request an Connect:Enterprise system file allocation (source member C$O68)
- Request an Connect:Enterprise system file deallocation (source member C$O69)
- Request an Connect:Enterprise system File Space Allocation Display (source member C$O70)
  - Connect:Enterprise system File Space Allocation Display (source member C$O71)

The DSECTs provided are specific to each of these functions. The structure of the DSECTs is similar to those previously discussed. Header data is omitted from the following discussion except where defining specific header fields clarifies the documentation. Only full headers are used for the request IPS. Every request IPS issued by a user-written API receives a response (or acknowledgment) IPS from Connect:Enterprise.

The following diagram depicts the events that occur when a user-written API initiates an ISSUE COMMAND transaction to simulate entry of a console command in Connect:Enterprise.

Understanding the steps, as well as their chronological order, helps you understand the overall processing flow that is required in the execution of an ISSUE COMMAND transaction.



Note: The header field H00CDEFN and other related fields are utilized in the Initiate an Auto Connect command.

Fixed Trailer Data

The following trailer data is present in every DSECT and is described in the sequence it appears in the DSECT and is identified by the DSECT field label, where nn represents the last two characters in the specific DSECT.

Field	Description
OnnTLNG	A half-word aligned half-word format field containing the length of the trailer portion of the IPS. The length of this trailer can be variable if it contains both fixed data and variable data. DSECTS that include variable data are C$O02, C$O09, C$O13, C$O15, C$O17, C$O67 and C$O71.
OnnTRLID	An 8-character identifier stamp that must always contain C$Onn followed by three blanks.

Initiating an Auto Connect Command

Initiating an Auto Connect is unique because both an immediate response (command issued acknowledgment) is received and a delayed response (the Auto Connect completion messages) are forthcoming. Disposition of the delayed response is defined in the original request through the header fields H00CDEFN, H00SYSID, H00CPROG or H00CTRAN and optionally H00CTERM and H00CUSER. When present, these fields are edited for validity (nonblank, nonnulls, and so on). If the user-written API does not wish to receive delayed response messages these fields should be set to low values.

The delayed response DSECT (source member C$O02) and the communication flow diagram is presented at the end of the Issue Commands narrative.

Field	Description
O03CMD	A character format command line field that must contain the $$CONNECT command and appropriate parameters to define the Auto Connect for Connect:Enterprise to initiate. The format of the command line data is identical to the syntax used for actual console entry.

Requesting a $$DUMP Command

Requesting a SNAP dump requires the following information.

Field	Description
O05SCOPE	A single character format byte that defines the type of the SNAP dump to generate: 1 = Connect:Enterprise list of all Auto Connects. 2 = Control blocks related to the specified Line ID 3 = Complete Connect:Enterprise region 4 = Connect:Enterprise system (anchor) control block 5 = Processor Router system control block
O05LINID	An 8-character format field that must contain the BSC Line ID when the O05SCOPE field specifies 2.

The data specified in this DSECT is used to issue an equivalent $$DUMP command to Connect:Enterprise.

Requesting a $$LIST Command

Requesting a List status of Traces, BSC lines, SNA sessions, Auto Connect Queue, or a combination list of all except the Auto Connect Queue requires the following information.

Field	Description	
O07OPTN	A single-character format byte that defines the type of list to generate.	
	1	Traces
	2	BSC Lines
	3	SNA Sessions
	4	All of the above (A/C Queue is not included)
	5	Auto Connect Queue

Field	Description
O07MXENT	A half-word aligned half-word format field that must contain the maximum number of entries the user-written API is prepared to handle at a specific instance. This format relates directly to the size of the response buffer (length of the linked COMMAREA or the maximum TSQ record length) that is provided to the CICS interface. A reasonable value for this number and the length of the response buffer by response trailer DSECT identifier follows:

Response DSECT ID	O07MXENT Value	Response Buffer Length Calculation Formula
C$O08	1	H00HLEN+O08LNGTH
C$O09	90	H00HLEN+O09LNGTH+(O09SEGLN*90)
C$O13	100	H00HLEN+O13LNGTH+(O13SEGLN*100)
C$O15	190	H00HLEN+O15LNGTH+(O15SEGLN*190)
C$O17	150	H00HLEN+O17LNGTH+(O17SEGLN*150)

The data specified in this DSECT is used to issue an equivalent $$LIST command to Connect:Enterprise. The $$LIST result is returned to the user-written API in one of five DSECTs.

If the $$LIST request specified only traces, the Traces Status Display is returned in DSECT C$O08.

Field	Description
O08TRCID	An 8-character field containing the Trace ID if one has been specified previously.
O08ALLTP	An 8-character field containing either ACTIVE or INACTIVE to indicate the status of tracing ALLTP activity.
O08SNA	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing SNA activity.
O08VSAM	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing VSAM activity.
O08EXITS	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing information exchanged with user exits.
O08RPEOB	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing End of Batch application agent rules processing activity.
O08ACONN	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing the initiation and completion of Auto Connect activity.
O08PR	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing the processor routing (entry/exit) activity.
O08CP	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing the TP activity associated with certain command processors.

Field	Description
O08RPLOG	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing Logging application agent rules processing activity.
O08APO	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing (SNAP dumping) all APPC activity. This trace can generate massive volumes of output data and should not be used unless directed to do so by Sterling Commerce Customer Services personnel.
O08APQ	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing the activity between the Process Router and the APPC function. This trace provides a before and after view of all APPC traffic. This trace can generate massive volumes of output data and should not be used unless directed to do so by Sterling Commerce Customer Services personnel.
O08RPWKT	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing Wake Up Terminate application agent rules processing activity.
O08TCPC	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing the TCP Scheduler.

Connect:Enterprise traces can be required for debugging. Sterling Commerce Customer Services may ask you to turn on some traces. Several of these trace facilities can be resource intensive and can cause system performance degradation. You should not start traces (or allow traces to remain active) without a specific purpose.

If the $$LIST request specified only BSC Lines, the BSC Lines Status Display is returned in DSECT C$O09.

Field	Description
O09#ENT	A half-word aligned half-word format field that contains the number of BSC Line entries that are included in the response IPS.
O09LNID	An 8-character field containing the BSC Line ID.
O09COND	A 6-character field containing either OPEN or CLOSED indicating the condition of the line.
O09LNST	An 8-character field containing either ACTIVE or INACTIVE indicating the status of the line.
O09AC	A 1-character field containing either Y or N indicating whether this BSC Line activity was initiated through an Auto Connect.
O09BID	An 8-character field containing the Mailbox ID.
O09ACLST	An 8-character field containing either the Auto Connect list name or the Remote name.

If the $$LIST request specified only SNA Sessions, the SNA Sessions Status Display is returned in DSECT C$O13.

Field	Description
O13#ENT	A half-word aligned half-word format field that contains the number of SNA Session entries that are included in the response IPS.
O13REMID	An 8-character field containing the Mailbox ID.
O13SESST	An 8-character field containing either ACTIVE or INACTIVE indicating the status of the session.
O13FRAC	A 1-character field containing either Y or N indicating whether this SNA Session activity was initiated through an Auto Connect.
O13BID	An 8-character field containing the Mailbox ID.

If the $$LIST request specified Traces, BSC Lines and SNA Sessions, the SNA Sessions/BSC Lines/Traces Status Display is returned in DSECT C$O15.

Field	Description
O15#ENT	A half-word aligned half-word format field that contains the number of SNA Session/BSC Line entries that are included in the response IPS.
O15TRCID	An 8-character field containing the Trace ID if one has been specified previously.
O15ALLTP	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing ALLTP activity.
O15SNA	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing SNA activity.
O15VSAM	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing VSAM activity.
O15EXITS	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing information exchanged with user exits.
O15ACONN	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing the initiation and completion of Auto Connect activity.
O15PR	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing the processor routing (entry/exit) activity.
O15CP	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing the TP activity associated with certain command processors.
O15APO	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing (SNAP dumping) all APPC activity. This trace can generate massive volumes of output data and should not be used unless directed to do so by Sterling Commerce Customer Services personnel.

Field	Description
O15APQ	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing the activity between the Process Router and the APPC function. This trace provides a before and after view of all APPC traffic. This trace can generate massive volumes of output data and should not be used unless directed to do so by Sterling Commerce Customer Services personnel.
O15RPEOB	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing End of Batch application agent rules processing activity.
O15RPWKT	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing Wake Up Terminate application agent rules processing activity.
O15RPLOG	An 8-character field containing either ACTIVE or INACTIVE indicating the status of tracing Logging application agent rules processing activity.
O15LID	An 8-character field containing the BSC Line ID.
O15RMT	An 8-character field containing the SNA Remote name O15RMT and O15LID share the same eight positions of storage in the DSECT.
O15COND	A 6-character field containing either OPEN or CLOSED indicating the condition of the BSC line. This field is not used for SNA Session entries.
O15SESST	An 8-character field containing either ACTIVE or INACTIVE indicating the status of the line (for BSC Line entries) or of the session (for SNA Session entries).
O15FRAC	A 1-character field containing either Y or N indicating if this BSC Line or SNA Session activity was initiated through an Auto Connect.
O15BID	An 8-character field containing the Mailbox ID.
O15ACLST	An 8-character field containing either the Auto Connect list name or the Remote name.
O15THRD#	An 8-character field containing the FTP thread number.
O15TYPE	A 3-character field containing either BSC or SNA to indicate the type of entry.

If the $$LIST request specified Auto Connect Queue, the Auto Connect Queue Status Display is returned in DSECT C$O17.

Field	Description
O17#ENT	A half-word aligned half-word format field that contains the number of Auto Connect Queue entries that are included in the response IPS.
O17LNAM	An 8-character field containing the queued Auto Connect list name.
O17QDAT	A 5-character field containing the date (YYDDD) that the Auto Connect list was queued.
O17QTIM	A 5-character field containing the time (HH:MM) that the Auto Connect list was queued.
O17PRTY	A 10-character field containing the priority value used in controlling the sequence of starting queued Auto Connects.
O17REAS	A 20-character field containing a brief description of why the Auto Connect was queued.

Issuing Connect:Enterprise $$SHUTDOWN Command

Issuing an Connect:Enterprise shutdown requires the following information:

Field	Description
O16CMD	A character format command line field that must contain the $$SHUTDOWN or $$SHUTDOWN,I command for Connect:Enterprise to initiate. The format of the command line data is identical to the syntax used for actual console entry.

Restarting a Closed Line ($$START) Command

Restarting a closed line requires the following information:

Field	Description
O19CMD	A character format command line field that must contain the $$START command and BSC Line ID for Connect:Enterprise to restart. The format of the command line data is identical to the syntax used for actual console entry.

Stopping an Auto Connect or Remote Connect Command

Stopping an Auto Connect or Remote Connect requires the following information.

Field	Description
O21CMD	A character format command line field that must contain the $$STOP command and appropriate parameters to define the Auto Connect or Remote for Connect:Enterprise to terminate. The format of the command line data is identical to the syntax used for actual console entry.

Stop/Start Traces Command

Connect:Enterprise traces can be required for debugging. Sterling Commerce Customer Services personnel may ask you to turn on some traces. Several of these trace facilities can be resource intensive and can cause system performance degradation. Do not start traces (or allow traces to remain active) without a specific purpose.

Issuing a Stop or Start traces command requires the following information.

Field	Description
O23TRCID	An 8-character field containing the Trace ID of the trace to activate.

Field	Description
O23IDOFF	A 1-character flag field that contains a 2 if the current Trace ID should be eliminated (turned off) or a 1 if the Trace ID should remain unchanged.
O23ALLTP	A 1-character field containing either 1 (to activate) or 2 (to deactivate) the tracing of ALLTP activity.
O23SNA	A 1-character field containing either 1 (to activate) or 2 (to deactivate) the tracing of SNA activity.
O23VSAM	A 1-character field containing either 1 (to activate) or 2 (to deactivate) the tracing of VSAM activity.
O23EXITS	A 1-character field containing either 1 (to activate) or 2 (to deactivate) the tracing of information exchanged with user exits.
O23RPEOB	A 1-character field containing either 1 (to activate) or 2 (to deactivate) the tracing of End of Batch application agent rules processing activity.
O23ACONN	A 1-character field containing either 1 (to activate) or 2 (to deactivate) the tracing of the initiation and completion of Auto Connect activity.
O23PR	A 1-character field containing either 1 (to activate) or 2 (to deactivate) the tracing of processor routing (entry/exit) activity.
O23CP	A 1-character field containing either 1 (to activate) or 2 (to deactivate) the tracing of TP activity associated with certain command processors.
O23RPLOG	A 1-character field containing either 1 (to activate) or 2 (to deactivate) the tracing of Logging application agent rules processing activity.
O23APO	A 1-character field containing either 1 (to activate) or 2 (to deactivate) the tracing (SNAP dumping) of all APPC activity. This trace can generate massive volumes of output data and should not be used unless you are asked to run it by Sterling Commerce Customer Services personnel.
O23APQ	A 1-character field containing either 1 (to activate) or 2 (to deactivate) the tracing of activity between the Process Router and the APPC function. This trace provides a before and after view of all APPC traffic. This trace can generate massive volumes of output data and should not be used unless directed to do so by Sterling Commerce Customer Services personnel.
O23RPWKT	A 1-character field containing either 1 (to activate) or 2 (to deactivate) tracing Wake Up Terminate application agent rules processing activity.
O23TCPSC	A 1-character field containing either 1 (to activate) or 2 (to deactivate) tracing of the TCP Scheduler.
O23FTPB	A 1-character field containing either 1 (to activate) or 2 (to deactivate) tracing of the FTP trace buffer.
O23FTPD	A 1-character field containing either 1 (to activate) or 2 (to deactivate) tracing of the FTP dialog buffer.
O23FTPTR	The FTP trace remote name.
O23FTPDR	The FTP dialog remote name.

Requesting a Files Listing ($$LIST FILES Command):

Requesting a Connect:Enterprise system files listing command requires the following information:

Field	Description
O66MXENT	A half-word aligned half-word format field that must contain the maximum number of file entries that the user-written API is prepared to handle at a specific instance. This format relates directly to the size of the response buffer (the length of the linked COMMAREA or the maximum TSQ record length) that is provided to the CICS interface. To request a maximum of 24 file entries this field would be specified as a half-word 24, requiring a buffer size computed as follows: H00HLEN+O67LNGTH+(O67FLEN*24) Because these DSECT labels are subject to change without prior notice, use the specified labels in your API program to calculate the actual buffer size.
O66DIRCT	A 1-character field that describes in which direction to search the list of system file names. Use the equated values O66FWD and O66BKWD (supplied in the DSECT) to specify the desired direction. This indicator is used in conjunction with the field O66FNAM to define the beginning key and direction for processing the list of system files.
O66FNAM	An 8-character field that contains a file ID name (VCF, VPF, VBQ01 through VBQ20, VLF1 or VLF2). This file ID name is used in conjunction with the field O66DIRCT to control processing the list of system files. To search the list of system files from the beginning, set this field to low values and indicate forward (O66FWD) in the O66DIRCT field. To request file information for only the log files, set this field to 'VLF1 ', indicate forward search in O66DIRCT and specify 2 files in O66MXENT. The data specified in this DSECT is used to issue an equivalent $$LIST FILES command to Connect:Enterprise. The $$LIST FILES result is returned to the user-written API in the C$O67 DSECT.

Requesting a Files Listing ($$LIST FILES Command response)

Requesting a Connect:Enterprise system files listing command response requires the following information:

Field	Description
O67BFNAM	An 8-character field (not all characters are currently used) that contains the file ID name that describes the beginning point within the list of system files for data returned in this C$O67 DSECT. See O67SCIND for an explanation of the validation process and how this field can be used.
O67EFNAM	An 8-character field (not all characters are currently used) that contains the file ID name that describes the ending point within the list of system files for data returned in this C$O67 DSECT. See O67SCIND for an explanation of the validation process and how this field can be used.

Field	Description
O67#ENT	A half-word aligned half-word format field that contains the actual number of file entries returned in the variable portion of this trailer DSECT.
O67SCIND	A 1-character field that describes the current placement within the list of system files relative to your request. If you requested a single file entry, this field is not used. In a multiple file entry request, if the first file entry returned in this trailer DSECT is not the first file in the list of system files, O67BKWD is set indicating you can search backward in the list for more files. If the last file entry returned in this trailer DSECT is not the last file in the list of system files, O67FWD is set indicating you can search forward in the list for more files. Use the equated values O67FWD and O67BKWD (supplied in the DSECT) to interpret this field.

If forward files are available, the field O67EFNAM contains a valid file ID name to use for searching forward. This file ID name, placed into field O66FNAM in conjunction with field O66DIRCT being set to O66FWD, instructs Connect:Enterprise to search the list of system files forward for the number of file entries specified in O66MXENT.

If backward files are available, the field O67BFNAM contains a valid file ID name to use for searching backward. This file ID name, placed into field O66FNAM in conjunction with field O66DIRCT being set to O66BKWD, tells Connect:Enterprise to search the list of system files backward for the number of file entries specified in O66MXENT.

Note: The remaining fields described in the C$O67 DSECT below represent a single file entry. This data is repeated within the actual buffer area the number of times defined by O67#ENT.

Field	Description
O67FNAME	An 8-character field (not all characters are currently used) that contains the file ID name of this file.
O67FSTAT	A 1-character field that describes the status of this file. A file is either allocated or not allocated. If allocated, it is either the current collection file or not the current collection file. Use the equated values O67ALLOC and O67CURC (supplied in the DSECT) to interpret this field.
O67FDSN	A 44-character field containing the VSAM Cluster name of this file.

Issuing Connect:Enterprise $$ALLOC Command

Issuing a Connect:Enterprise system file allocation requires the following information:

Field	Description
O68CMD	A character format command line field that must contain the command $$ALLOC xxxxx or $$ALLOC xxxxx,C (where xxxxx is VBQ01 through VBQ20, VLF1 through VLF8) for execution by Connect:Enterprise. The format of the command line data is identical to the syntax used for actual console entry.

Issuing Connect:Enterprise $$DALLOC Command

Issuing a Connect:Enterprise system file deallocation requires the following information:

Field	Description
O69CMD	A character format command line field that must contain the command $$DALLOC xxxxx (where xxxxx is VBQ01 through VBQ20, VLF1 through VLF8) for execution by Connect:Enterprise. The format of the command line data and the rules of execution are identical to those used for actual console entry.

Requesting a Space Allocation Listing ($$SPACE Command)

Requesting a Connect:Enterprise system File Space Allocation Display requires the following information:

Field	Description
O70MXENT	A half-word aligned half-word format field that must contain the maximum number of file entries that the user-written API is prepared to handle at a specific instance. This format relates directly to the size of the response buffer (the length of the LINKed COMMAREA or the maximum TSQ record length) that is provided to the CICS interface. To request a maximum of 24 file entries this field would be specified as a half-word 24, requiring a buffer size computed as follows: H00HLEN+O71LNGTH+(O71FLEN*24) Because these DSECT labels are subject to change without prior notice, it is recommended that the specified labels be used in your API program to calculate the actual buffer size.
O70DIRCT	A 1-character field that describes the direction in which to search the list of system file names. Use the equated values O70FWD and O70BKWD (supplied in the DSECT) to specify the desired direction. This indicator is used in conjunction with the field O70FNAM to define the beginning key and direction for processing the list of system files.

Field	Description
O70FNAM	An 8-character field that contains a file ID name (VCF, VPF, VBQ01 through VBQ20, VLF1 or VLF2). This file ID name is used in conjunction with the field O70DIRCT to control processing the list of system files. To search the list of system files from the beginning, set this field to low values and indicate forward (O70FWD) in the O70DIRCT field. To request file information for only the log files, set this field to 'VLF1 ', indicate forward search in O70DIRCT and specify 2 files in O70MXENT. The data specified in this DSECT is used to issue an equivalent $$SPACE command to Connect:Enterprise. The $$SPACE result is returned to the user-written API in the C$O71 DSECT.

Requesting a File Space Allocation Listing ($$SPACE Command Response)

Requesting a File Space Allocation Listing requires the following information:

Field	Description
O71BFNAM	An 8-character field (not all characters are currently used) that contains the file ID name that describes the beginning point within the list of system files for data returned in this C$O71 DSECT. See O71SCIND for an explanation of the validation process and how this field can be used.
O71EFNAM	An 8-character field (not all characters are currently used) that contains the file ID name that describes the ending point within the list of system files for data returned in this C$O71 DSECT. See O71SCIND below for an explanation of the validation process and how this field can be used.
O71#ENT	A half-word aligned half-word format field that contains the actual number of file entries returned in the variable portion of this trailer DSECT.
O71SCIND	A 1-character field that describes the current placement within the list of system files relative to your request. If you requested a single file entry, this field is not used. In a multiple file entry request, if the first file entry returned in this trailer DSECT is not the first file in the list of system files, O71BKWD is set indicating you can search backward in the list for more files. If the last file entry returned in this trailer DSECT is not the last file in the list of system files, O71FWD is set indicating you can search forward in the list for more files. Use the equated values O71FWD and O71BKWD (supplied in the DSECT) to interpret this field.

If forward files are available, the field O71EFNAM contains a valid file ID name to use for searching forward. This file ID name, placed into field O70FNAM in conjunction with field O70DIRCT being set to O70FWD, instructs Connect:Enterprise to search the list of system files forward for the number of file entries specified in O70MXENT.

If backward files are available, the field O71BFNAM contains a valid file ID name to use for searching backward. This file ID name, placed into field O70FNAM in conjunction with field

O70DIRCT being set to O70BKWD, instructs Connect:Enterprise to search the list of system files backward for the number of file entries specified in O70MXENT.

Note: The remaining fields described in the C$O71 DSECT below represent a single file entry. This data is repeated within the actual buffer area the number of times defined by O71#ENT.

Field	Description
O71FNAME	An 8-character field (not all characters are currently used) that contains the file ID name of this file.
O71TYPE	A 1-character field that describes the format of the data within this file entry. Information in this file entry is either valid space allocation information or an error entry. Use the equated values O71VSPC and O71ERRE (supplied in the DSECT) to interpret this field.
The following fields constitute a valid space allocation information entry.	
O71PCT	A 3-character field that contains, in decimal characters (0–9), a percentage of the VSAM data component capacity that is currently used. In other words, how full this file is. The valid range for this data is C‘000’ through C‘100’.
O71HRBA	A 10-character field that contains, in decimal characters (0–9), the high allocated relative byte address of the end of the data component (for example, the last available byte in the data set). The valid range for this data is C‘0000000000’ through C‘4294967295’.
O71ERBA	A 10-character field that contains, in decimal characters (0–9), the ending relative byte address of the space used in the data component (for example, the last used byte in the data set at the current time). The valid range for this data is C‘0000000000’ through C‘4294967295’.
O71FREE	A 10-character field that contains, in decimal characters (0–9), the number of bytes of available space in the data component. The valid range for this data is C‘0000000000’ through C‘4294967295’.
O71XTNT	A 3-character field that contains, in decimal characters (0–9), the number of extents now allocated to the data component. The valid range for this data is character C‘001’ through character C‘123’.
O71XTNTP	A 1-character field that contains either a blank (X‘40’) or a plus sign (+). The plus sign indicates that the data set has gone into additional extents since the VSAM Server last opened the file.
The following fields constitute a VSAM error entry.	
O71FUNC	An 8-character field that contains a character string identifier describing the VSAM function (OPEN, CLOSE, or SHOWCB) that failed during the attempt to retrieve the space allocation information.
O71RC	A 4-character field that contains, in decimal characters (0–9), the Register 15 value returned following the VSAM error. The valid data for this field is any Return Code generated by VSAM for OPEN, CLOSE, or SHOWCB.
O71RSN	An 8-character field that contains, in hexadecimal characters (0–F), the Reason Code value returned following the VSAM error. The valid data for this field is any Reason Code generated by VSAM for OPEN, CLOSE, or SHOWCB.

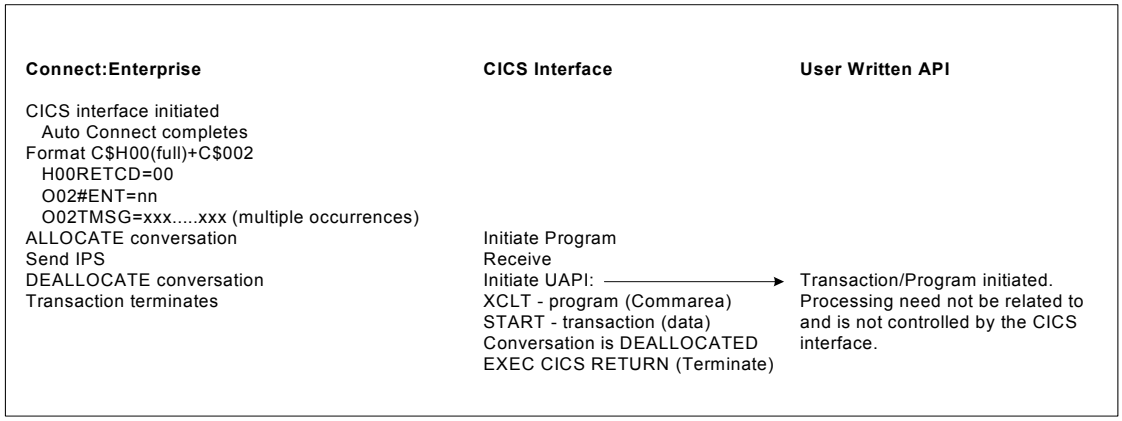
Field	Description
O71ERR	An 8-character field that contains, in hexadecimal characters (0–F), the Error Code value returned following the VSAM OPEN or CLOSE error. The valid data for this field is any Error Code generated by VSAM.

Requesting Auto Connect Completion Messages

This response is returned to the transaction or program that was defined in the header of the Initiate an Auto Connect command (DSECT C$O03). The use of the header field H00CDEFN and related fields to define the recipient of this DSECT was explained in *Initiating an Auto Connect Command* on page 228.

Field	Description
O02#ENT	A half-word aligned half-word format field that contains the number of message record entries that are included in the response IPS.
O02STMSG	A 79-character format field that contains an Connect:Enterprise console message. The format of the message data is identical to the format used for actual console displays.

The following diagram shows what events occur when an Auto Connect (initiated by an command from the CICS interface) completes. The steps the user completes as well as the order in which these steps are completed are important to understanding the processing flow of the Auto Connect completion delayed response transaction.

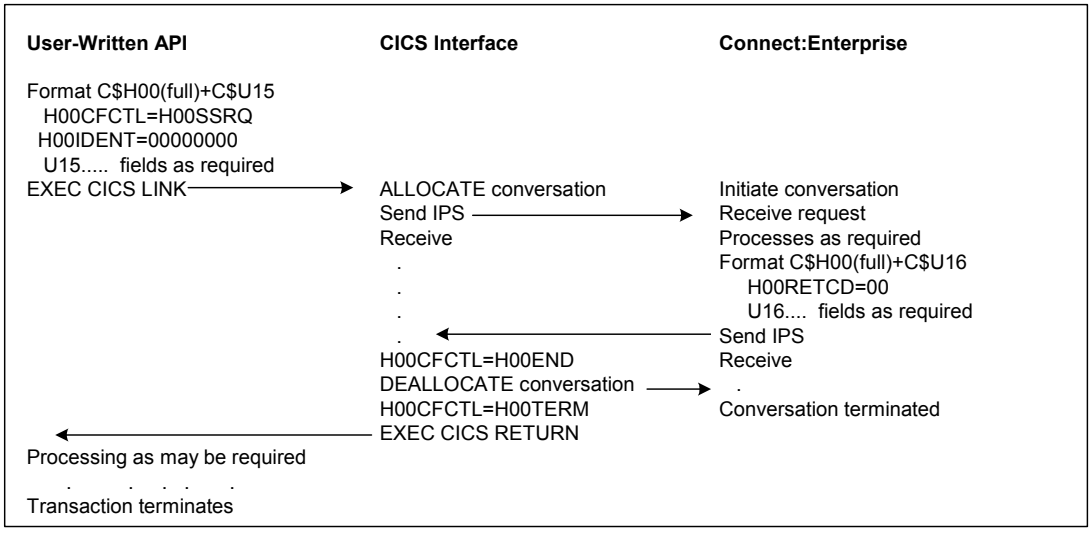


Using the Directory Listing Transaction

This facility allows a user-written transaction to receive information describing the content of Batch Queues from Connect:Enterprise. The header on each IPS must be a full header. See the *Interface Parameter Structure Content* on page 206 for a description of header data fields and content.

The C$U15 DSECT is used to specify selection criteria, thereby reducing the volume of directory information that is returned by Connect:Enterprise. The VSAM record key that controls reading the Batch Queue(s) is also specified in this DSECT. The C$U16 DSECT contains directory information returned by Connect:Enterprise. The VSAM record keys (beginning and ending) that describe the returned data are also included in this DSECT. These keys are used (in DSECT C$U15) to control subsequent reading of the VSAM Batch Queue(s). The response (C$U16 DSECT) returned by Connect:Enterprise is formatted with a full header.

The following diagram depicts the events that occur when a user-written API initiates a Directory Listing transaction to retrieve information about current content of the Batch Queues from Connect:Enterprise. Understanding the steps that the user must complete as well as the order in which the steps are completed will help you understand the Directory Listing processing flow. The following diagram illustrates a single Directory Listing request being serviced by Connect:Enterprise. The flow control flag (C$U15 DSECT) is shown as SEND, RECEIVE then QUIT (H00SSRQ). Additional requests are executed in the identical manner except the beginning VSAM key within the C$U15 DSECT is set from data returned in the previous response DSECT.



Fixed Trailer Data

The trailer data below is present in both DSECTs and is described in the sequence it appears in the DSECT and is identified by the DSECT field label where (nn) represents the last two characters in the specific DSECT.

Field	Description
UnnTLNG	A half-word aligned half-word format field containing the length of the trailer portion of the IPS. The length of the C$U15 DSECT is fixed while the length of the C$U16 DSECT is variable.
UnnTRLID	An 8-character identifier stamp that must always contain C$Unn followed by three blanks.

Requesting a Directory Listing

Requesting a directory listing requires the following information.

Field	Description
U15MXENT	A half-word aligned half-word format field that must contain the maximum number of directory entries that the user-written API is prepared to handle at a specific instance. This format relates directly to the size of the response buffer (the length of the linked COMMAREA or the maximum TSQ record length) that is provided to the CICS interface. To request a maximum of 100 directory entries this field would be specified as a half-word 100, requiring a buffer size computed as follows: H00HLEN+U16LNGT+(U16SEGL*100) Because these DSECT labels are subject to change without prior notice, use the specified labels in your API program to calculate the actual buffer size.
U15REMID	An 8-character field containing a Mailbox ID or blanks. If the field is nonblank, any directory entries returned match the Mailbox ID specified. A generic Mailbox ID can be specified by entering an asterisk (*) as the final character in the Mailbox ID field. If the field is blank, the directory entries returned are not restricted to any specific Mailbox ID.
U15USBID	A 24-character field containing a user batch ID or blanks. If the field is nonblank, any directory entries returned match the user batch ID specified. If the field contains blanks, the directory entries returned are not restricted to any specific user batch ID. **Note:** This field is only generated if FORMAT=1 is specified (as a 24 byte field).
The following flag bytes are similar in function. Each field is a 1-character field indicating a specific batch status. If the field contains a 1, any directory entries returned match the specified status. If the field contains a 2, any directory entries returned cannot match the specified status. If the field contains a blank, the specified status is not used to influence whether or not the batch becomes a directory entry that is returned.	
U15RFLG	The Online Requestable status.
U15DFLG	The Flagged for Deletion status.
U15TFLG	The Online Transmitted status.

Field	Description
U15EFLG	The Extracted status.
U15MFLG	The Multiple Transmit status.
U15IFLG	The Incomplete Batch status.
U15XFLG	The Transparent Data status.
U15DIRCT	A 1-character field that describes the type of VSAM read. Use the equated values U15FWD and U15BKWD (supplied in the DSECT) to specify the desired direction. This indicator is used in conjunction with the field U15BEGKY to define the beginning key and direction for reading the VSAM Batch Queue.
U15BEGKY	An 18-character field that contains the VSAM Batch Queue identifier and the key for the VSAM Batch Queue. This key is used in conjunction with the field U15DIRCT to control reading the VSAM Batch Queue(s). Initially this field is low values to begin searching the Batch Queue at the first record. With each directory data response from Connect:Enterprise, a key is returned that defines the beginning and ending record for the returned data. When the API program has completed processing the directory entries and is ready to request the next group of directory entries, the ending key (U16ENDKY) from the current directory response is placed into this field. See the description of U16BEGKY and U16ENDKY for additional information.
U15VBQTP	A 1-character field that describes the Batch Queue(s) to use for the directory listing data. Use the equated values U15CCQ, U15QNN or U15ALLQ (supplied in the DSECT) to specify the desired Batch Queue. When the current collection queue (U15CCQ) is initially requested, Connect:Enterprise returns a specific Batch Queue number in the response. If a scroll request is issued (to logically continue the directory listing) you must use the specific Batch Queue number that was returned to ensure the same Batch Queue is being accessed. When a specific Batch Queue (U15QNN) is requested, the associated field U15BQNN must contain the identifier number for the target Batch Queue. When all Batch Queues (U15ALLQ) is requested, the key fields (U16BVBQ and U16EVBQ) in each response identify the current location within the Batch Queues.
U15BQNN	A 1-character field that describes a specific Batch Queue to use for the directory listing data. This field must contain the Batch Queue number (01–20) in binary format whenever U15VBQTP is set to U15QNN.
U15BID64	A 64-character field that specifies the user batch ID. **Note:** This field is only generated if FORMAT=2 is specified.
U15BID64L	A 1-character field that specifies the length of the U15BID64 field. **Note:** This field is only generated if FORMAT=2 is specified.
U15FMTID	A 1-character field that must contain c'F2'. **Note:** This field is only generated if FORMAT=2 is specified.

Field	Description
U15UBLEN	A 2-character field that defines the length of the user batch ID field when the FORMAT=1 parameter is specified. The field contains the length (as a binary value) of the data stored in the field U15USBID. The length can be interpreted as follows: 0 = no user batch ID data is supplied. The field U15USBID should be blanks. 01 - 23 = generic user batch ID is supplied. The field U15USBID contains a User Batch ID of the length specified followed by low-values. 24 = specific user batch ID is supplied. The field U15USBID contains a 24-character user batch ID. **Note:** This field is only generated if FORMAT=1 is specified.
U15FDATE	A 7-character field that contains the From Date used in a date range search of the Batch Queues. The field can contain one of three date formats: YYYYDDD = a modified Julian format date that combines a 4-digit year with a 3-digit day. YYDDD = a conventional Julian format date. Left justify and pad with blanks. NNN = a number of days to subtract from the current date resulting in the desired From Date. Left justify and pad with blanks. The field can be filled with blanks to indicate the From Date includes the oldest batch date on file.
U15FDFOR	A 1-character field that describes the contents of the U15FDATE field. Use the equated values U15FD7, U15FD5 or U15FD3 (supplied with the DSECT) to indicate which format U15FDATE contains. When U15FDATE is blank, this field contains low-values.
U15TDATE	A 7-character field that contains the To Date used in a date range search of the Batch Queues. The field can contain one of three date formats: YYYYDDD = a modified Julian format date that combines a 4-digit year with a 3-digit day. YYDDD = a conventional Julian format date. Left justify and pad with blanks. NNN = a number of days to subtract from the current date resulting in the desired To Date. Left justify and pad with blanks. The field can be filled with blanks to indicate the To Date includes the newest batch date on file.
U15TDFOR	A 1-character field that describes the contents of the U15TDATE field. Use the equated values U15TD7, U15TD5 or U15TD3 (supplied with the DSECT) to indicate which format U15TDATE contains. When U15TDATE is blank, this field contains low-values.
U15FTIME	A 4-character field that contains the From Time used to search the Batch Queues. The field contains military format (24 hour clock) hour and minute. For example, 11:59 in the morning (one minute before noon) is specified as 1159 while 11:59 at night (one minute before midnight) is specified as 2359. The field can be filled with blanks to specify the From Time as midnight. See the associated field U15TMFOR for a description of how the time value is used during the search process.
U15TTIME	A 4-character field that contains the To Time used to search the Batch Queues. The field contains military format (24 hour clock) hour and minute. The field can be filled with blanks to specify the To Time as the current time when Connect:Enterprise is actually processing the IPS. See the associated field U15TMFOR for a description of how the time value is used during the search process.

Field	Description
U15TMFOR	A 1-character field that describes how the From Time and To Time fields are used. Use the equated values U15TWIN or U15TABS (supplied with the DSECT) to indicate which selection method is used. ◆ If you specify a time window (U15TWIN), the From Time and To Time are applied each day of the specified date range. For example, if you specify From 0800 To 1730, batches that were created between 8:00am and 5:30pm on each day of the specified date range are qualified for selection based upon the date criteria. ◆ If you specify an absolute time (U15TABS), the From Time is the beginning time on the first day of the date range and To Time is the ending time on the last day of the date range. For example, if you specify From 0800 To 1730, batches that were created between 8:00am on the first day of the specified date range and 5:30pm on the last day of the specified date range are qualified for selection based upon the date criteria.
U15BNOBG	A 7-character field that contains the range beginning Batch Number or blanks. If the field is nonblank, any directory entries returned are within the Batch Number range specified. When this field is nonblank U15BNOEN must also be specified. If the field contains blanks, the directory entries returned are not restricted to a specific range of batch numbers.
U15BNOEN	A 7-character field that contains the range ending Batch Number or blanks. If the field is nonblank, any directory entries returned are within the Batch Number range specified. When this field is nonblank U15BNOBG must also be specified. If the field contains blanks, the directory entries returned are not restricted to a specific range of batch numbers.
U15ORUS	A 1-character field that describes the selection criteria relationship. Use the equated values U15ANY or U15ALL (supplied with the DSECT) to indicate which selection method is used. ◆ If you specify U15ANY, directory entries that match any of your specified selection criteria qualify to be returned. ◆ If you specify U15ALL, only those directory entries that match all of your specified selection criteria qualify to be returned. This field is required and must be specified.
U15SCAMT	A 4-byte binary scroll amount.
U15WILD	A 1-byte wild card character for user batch ID selection.
U15CASE	A 1-character field indicating if the selection on the Mailbox ID should be case sensitive.
U15UORL	A 1-character field indicating if upper/lower case is used.
The following flag bytes are a continuation of the Batch Status Selection Criteria described earlier in this DSECT definition. As indicated earlier, each field is a 1-character field indicating a specific batch status. If the field contains a 1, any directory entries returned matches the specified status. If the field contains a 2, any directory entries returned do not match the specified status. If the field contains a blank, the specified status is not used to influence whether or not the batch becomes a directory entry that is returned.	
U15AFLG	The Added Offline status.
U15BFLG	The BSC Transmission status.
U15CFLG	The Collected Online status.
U15QFLG	The EBCDIC (API) added status.
U15SFLG	The SNA Transmission status.

Field	Description
U15NFLG	The Non-Transmittable status.
U15UFLG	The Un-Extractable status.
U15FFLG	The FTP transmission status.
U15LFLG	Reserved
U15GFLG	The File Structure status.
U15HFLG	The SSL status.

Requesting Directory Information

Requesting directory information requires the following information.

Field	Description
U16#ENT	A half-word aligned half-word format field that contains the actual number of directory entries returned in the variable portion of this trailer DSECT.
U16BCHID	A 24-character field containing the user batch ID associated with the batch. **Note:** This field is only generated if FORMAT=1 is specified.
U16BCH64	A 64-character field containing a user batch ID or blanks used for each returned batch. **Note:** This field is only generated if FORMAT=2 is specified.
U16REMID	An 8-character field containing a Mailbox ID or blanks. It echoes the value specified for U15REMID.
U16USBID	A 24-character field containing a user batch ID or blanks. If the field is nonblank, any directory entries returned match the user batch ID specified. If the field contains blanks, the directory entries returned are not restricted to any specific user batch ID. **Note:** This field is only generated if FORMAT=1 is specified.
U16BID64	A 64-character field that specifies the user batch ID. **Note:** This field is only generated if FORMAT=2 is specified.
U16BID64L	A 1-character field that specifies the length of the U16BID64 field. **Note:** This field is only generated if FORMAT=2 is specified.
U16RFLG	The Online Requestable status.
U16DFLG	The Flagged for Deletion status.
U16TFLG	The Online Transmitted status.
U16EFLG	The Extracted status.
U16MFLG	The Multiple Transmit status.
U16IFLG	The Incomplete Batch status.

Field	Description
U16XFLG	The Transparent Data status.
U16SCIND	A 1-character field that describes the current placement within the VSAM Batch Queue. Use the equated values U16FWD and U16BKWD (supplied in the DSECT) to interpret this field.

If forward records are available, the field U16ENDKY contains a valid record key to be used for reading forward. This key value, placed into field U15BEGKY in conjunction with field U15DIRCT being set to U15FWD, instructs Connect:Enterprise to read the VSAM Batch Queue forward for the number of directory entries specified in U15MXENT.

If backward records are available, the field U16BEGKY contains a valid record key to be used for reading backward. This key value, placed into field U15BEGKY in conjunction with field U15DIRCT being set to U15BKWD, instructs Connect:Enterprise to read the VSAM Batch Queue backward for the number of directory entries specified in U15MXENT.

Field	Description
U16BEGKY	An 18-character field that contains the VSAM Batch Queue identifier (U16BVBQ) and the key for the VSAM Batch Queue (U16BGKY). This key describes the record prior to (if any) the first directory entry contained within this C$U16 DSECT. This key does not necessarily echo the value specified for U15BEGKY. See U16SCIND for an explanation of the validation process and how this field can be used.
U16ENDKY	An 18-character field that contains the VSAM Batch Queue identifier (U16EVBQ) and the key for the VSAM Batch Queue (U16ENKY). This key describes the record following (if any) the last directory entry contained within this C$U16 DSECT. This field has no equivalent in the C$U15 DSECT. See U16SCIND for an explanation of the validation process and how this field can be used.
U16VBQTP	A 1-character field that describes the Batch Queue(s) to be used for the directory listing data. It echoes the value specified for U15VBQTP unless U15VBQTP was set to U15CCQ. In this case, U16VBQTP is set to U16QNN and U16BQNN contains the Batch Queue identification number of the current collection batch queue.
U16BQNN	A 1-character field that describes a specific Batch Queue to be used for the directory listing data. It echoes the value specified for U15BQNN except as explained for U16VBQTP when U15VBQTP was set to U15CCQ.
U16UBLEN	A 1-byte hex field that defines the length of the user batch ID field. It echoes the value specified in U15UBLEN. **Note:** If C$U16 FORMAT=1 is specified, U16UBLEN values = 0, 1-23, 24.This field is only generated if FORMAT=1 is specified.
U16FMTID	A 1-character field that must contain c'F2' when the FORMAT=2 parameter is specified.
U16FDATE	A 7-character field that contains the From Date or blanks used in a date range search of the Batch Queues. It echoes the value specified in U15FDATE.
U16FDFOR	A 1-character field that describes the contents of the U16FDATE field. It echoes the value specified in U15FDFOR.

Field	Description
U16TDATE	A 7-character field that contains the To Date or blanks used in a date range search of the Batch Queues. It echoes the value specified in U15TDATE.
U16TDFOR	A 1-character field that describes the contents of the U16TDATE field. It echoes the value specified in U15TDFOR.
U16FTIME	A 4-character field that contains the From Time or blanks used to search the Batch Queues. It echoes the value specified in U15FTIME.
U16TTIME	A 4-character field that contains the To Time or blanks used to search the Batch Queues. It echoes the value specified in U15TTIME.
U16TMFOR	A 1-character field that describes how the From Time and To Time fields are used. It echoes the value specified in U15TMFOR.
U16BNOBG	A 7-character field that contains the range beginning Batch Number or blanks. It echoes the value specified in U15BNOBG.
U16BNOEN	A 7-character field that contains the range ending Batch Number or blanks. It echoes the value specified in U15BNOEN.
U16ORUS	A 1-character field that describes the selection criteria relationship. It echoes the value specified in U15ORUS.
The following flag bytes are a continuation of the Batch Status Selection Criteria described earlier in this DSECT definition. Each field is a 1-character field indicating a specific batch status. The field contains a 1, a 2, or is left blank. Each field echoes the value specified for the equivalent field in the C$U15 DSECT.	
U16AFLG	The Added Offline status.
U16BFLG	The BSC Transmission status.
U16CFLG	The Collected Online status.
U16QFLG	The EBCDIC (API) added status.
U16SFLG	The SNA Transmission status.
U16NFLG	The Not-Transmittable status.
U16UFLG	The Un-Extractable status.
U16UORL	A 1–character field indicating if upper/lower case is used.
U16FFLG	The FTP transmission status.
U16GFLG	The File Structure status.
U16HFLG	The SSL status.

Field	Description
U16ADJ	An equated value that defines the length of the additional data that was added to the fixed portion of the C$U16 DSECT by the Version 2.1 modifications. By using this label as a negative adjustment with any labels from U16VCFKY to U16FILL you can correctly address a C$U16 trailer created by a Version 1.1 Connect:Enterprise while using the new C$U16 format DSECT. For example, assume you have can address the first byte of the C$U16 data and you wish to reference field U16KYID. The location of U16KYID is different in Version 1.1 than it is in Version 2.1. Once you determine the source of the C$U16 data you can access the appropriate data location as follows: **Data from Version 2.1** — **Data from Version 1.1** LA Rn,U16KYID — LA Rn,U16KYID-U16ADJ MVC .....,U16KYID — MVC .....,U16KYID-U16ADJ CLC U16KYID,..... — CLC U16KYID-U16ADJ,.....
U16VCFKY	A 4-byte packed decimal field that contains the batch number.
U16KYID	An 8-character field containing a Mailbox ID. This data is part of the 18-byte VSAM Batch Queue number and record key.
U16KYBNO	A 4-byte packed decimal field that contains the batch number. This data is part of the 18-byte VSAM Batch Queue number and record key.
U16KYRNO	A 5-byte packed decimal field that contains the record number (always zero for a directory listing). This data is part of the 18-byte VSAM Batch Queue number and record key.
U16KEY	This field redefines the U16KY... key data fields as a single length field containing the non-VCF file VSAM Batch Queue number and record key.
U16#BLKS	A 5-byte packed decimal field that contains the number of blocks in the batch.
The following flag bytes are each 1-character fields indicating information about the batch through bit flags in these flag bytes. Use the equated values (supplied in the DSECT) to interpret each of these flag bytes.	
U16FLAG1	See C$U16 DSECT for interpretation values and comments.
U16FLAG2	See C$U16 DSECT for interpretation values and comments.
U16FLAG3	See C$U16 DSECT for interpretation values and comments.
U16FLAG4	See C$U16 DSECT for interpretation values and comments.
U16NEXT#	A 4-byte packed decimal field that contains the batch number of a related batch.
U16CDATE	A 6-character field that contains the year and day (Julian format YYDDD) recorded as the batch creation date. The right-most position of this field is blank.
U16OPRID	An 8-character field that contains an operator ID. This field is reserved for future use.
U16RTYPE	A 1-character field that describes the Remote type for batches that were collected online. Use the equated values (supplied in the DSECT) to interpret this field.

Field	Description
U16ERCL	A 2-character field that contains the exchange record length value when MEDIA=Basic Exchange.
U16EOB@	A 1-character field that contains an indication that the End of Batch exit has been driven for this batch. Use the equated value (supplied in the DSECT) to interpret this field.
U16MEDIA	A 1-character field that indicates the received batch media type. Use the equated values (supplied in the DSECT) to interpret this field.
U16KYBQN	A 1-character field that contains the Batch Queue number on which this entry resides. This character is a binary value from 01 (x'01') through 20 (x'14').
U16CTIM	A 6-character field that contains the hour, minute and second (HHMMSS) recorded as the batch creation time.

Appendix A

IPS Trailers

The following table identifies all IPS trailers used by the ISPF interface, CICS interface and CICS User API. You can use this information when diagnosing problems, creating user-written programs that use the CICS User API, or determining what functions need to be protected with the security interface or security exits.

Note: The third character for some of the items in the Trailer ID column is the letter O. There are no C$0xx IPS Trailers (using the numeral 0).

Trailer ID:	Screen Number:	Screen Title:	Screen Type:	Screen Function:
C$A20	GENERIC	NONE	INPUT/OUTPUT (Connect:Enterprise REQUEST & RESPONSE)	USER API "ADD" REQUEST AND RESPONSE
C$O02	GENERIC	GENERIC RESPONSE	OUTPUT (Connect:Enterprise RESPONSE)	RETURN CONSOLE CM.XXXX MESSAGES
C$O03	3.1.1	AUTO CONNECT INITIATION REQUEST	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES $$CONNECT REQUEST
C$O05	3.1.2	ONLINE SNAP DUMP REQUEST	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES $$DUMP REQUEST
C$O07	3.1.3	LIST REQUEST	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES $$LIST REQUEST
C$O074	3.1.3.7	FTP SESSIONS STATUS DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY STATUS OF FTP SESSIONS

Trailer ID:	Screen Number:	Screen Title:	Screen Type:	Screen Function:
C$O078	CEBR BROWSE	C:E Connect:Enterprise RESOURCE UTILIZATION DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY RESOURCE UTILIZATION STATISTICS
C$O079	CEBR BROWSE	C:E Connect:Enterprise STORAGE MAP DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY STORAGE USAGE
C$O08	3.1.3.1	TRACES STATUS DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY STATUS OF $$LIST TRACES
C$O09	3.1.3.2	BSC LINES STATUS DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY STATUS OF $$LIST LINES
C$O10	3.1.3.1	RULES STATUS DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY STATUS OF $$LIST RULES
C$O11	3.1.12	REFRESH FILES REQUEST	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES $$REFRESH FILES COMMAND
C$O12	3.1.13	INVOKE RULES REQUEST	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES $$INVOKE RULES COMMAND
C$O13	3.1.3.3	SNA SESSION STATUS DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY STATUS OF $$LIST SESSIONS
C$O14	3.1.12.1	REFRESH RULES REQUEST	INPUT (Connect:Enterprise REQUEST)	REFRESH RULES REQUEST
C$O15	3.1.3.4	SNA SESSION/BSC LINES/TRACES STATUS DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY STATUS OF $$LIST ALL
C$O16	3.1.4	Connect:Enterprise SHUTDOWN REQUEST	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES $$SHUTDOWN REQUEST
C$O17	3.1.3.5	AUTO CONNECT QUEUE STATUS DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY STATUS OF $$LIST ACQ

Trailer ID:	Screen Number:	Screen Title:	Screen Type:	Screen Function:
C$O19	3.1.5	START LINES/RULES REQUEST	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES $$START COMMAND
C$O21	3.1.6	STOP A/C R/C RULES REQUEST	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES $$STOP COMMAND
C$O23	3.1.7	TRACE MANAGEMENT REQUEST	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES $$TRACE REQUEST
C$O25	3.2.1	ACTIVE SESSIONS SUMMARY REQUEST	INPUT (Connect:Enterprise REQUEST)	REQUEST ACTIVE SESSION SUMMARY
C$O26	3.2.1.1	ACTIVE SESSIONS SUMMARY DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY SUMMARY OF ACTIVE SESSIONS
C$O27	3.2.1.1 and 3.2.2.1.1	ACTIVE SESSIONS SUMMARY DETAIL REQUEST	INPUT (Connect:Enterprise REQUEST)	REQUEST ACTIVE SESSION DETAIL
C$O29	3.2.1.1.1	ACTIVE SESSION DETAIL DISPLAY OF REMOTE AC/RC	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY DETAIL OF AUTO CONNECT REMOTE
C$O31	3.2.2	ACTIVE A/C SUMMARY REQUEST	INPUT (Connect:Enterprise REQUEST)	REQUEST ACTIVE A/C SUMMARY DISPLAY
C$O311	3.2.2.2	QUEUED A/C SUMMARY DISPLAY	OUTPUT (Connect:Enterprise RESPONSE DATA)	DISPLAY LIST OF QUEUED AUTO CONNECTS
C$O32	3.2.2.1	ACTIVE A/C SUMMARY DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY SUMMARY OF ACTIVE AUTO CONNECTS
C$O321	3.2.2.2	QUEUED A/C SUMMARY DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY SUMMARY OF QUEUED AUTO CONNECTS
C$O33	3.2.2.1	ACTIVE A/C REMOTE SUMMARY DISPLAY REQUEST	INPUT (Connect:Enterprise REQUEST)	REQUEST ACTIVE A/C REMOTE SUMMARY DISPLAY FROM

Trailer ID:	Screen Number:	Screen Title:	Screen Type:	Screen Function:
C$O34	3.2.2.1.1	ACTIVE A/C REMOTE SUMMARY DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY SUMMARY OF ACTIVE AUTO CONNECT REMOTES
C$O35	3.3.2	SECURITY RECORD SEARCH	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES SEARCH REQUEST
C$O36	3.3.2.1	SECURITY ID DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY SECURITY ID'S
C$O37	3.3.2.1	SECURITY ID UPDATE	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES UPDATE REQUEST
C$O38	3.3.1	OPTIONS DISPLAY	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES OPTIONS DISPLAY REQUEST
C$O39	3.3.1	OPTIONS DISPLAY/UPDATE	OUTPUT (Connect:Enterprise RESPONSE)	Connect:Enterprise RESPONSE TO OPTIONS DISPLAY/UPDATE
C$O40	3.3.5	SIGNON RECORD DISPLAY	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES SIGNON RECORD
C$O41	3.3.5	SIGNON RECORD DISPLAY/UPDATE	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY/UPDATE SIGNON RECORDS
C$O42	3.3.4	REMOTES DISPLAY	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES REMOTES DISPLAY REQUEST
C$O421	3.3.6	POOLS RECORD SELECTION REQUEST	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES POOLS DISPLAY REQUEST
C$O43	3.3.4.1	REMOTES LIST DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	Connect:Enterprise RESPONSE TO REMOTES DISPLAY
C$O431	3.3.6.1	POOLS SELECTION LIST DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	Connect:Enterprise RESPONSE TO POOLS DISPLAY
C$O44	3.3.4.1.1	REMOTES UPDATE	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise REQUEST TO UPDATE REMOTE

Trailer ID:	Screen Number:	Screen Title:	Screen Type:	Screen Function:
C$O441	3.3.6.1	POOLS RECORD SELECTION LIST	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise REQUEST TO DELETE POOL
C$O45	3.3.3	AUTO CONNECT LISTNAME SEARCH	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES SEARCH REQUEST
C$O46	3.3.3.1; 3.3.3.1.1; 3.3.3.1.2	AUTO CONNECT LISTNAME DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	Connect:Enterprise RESPONSE TO LISTNAME SEARCH
C$O47	3.3.3.1.3	AUTO CONNECT REMOTE SEARCH	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES SEARCH REQUEST
C$O48	3.3.3.1.3	AUTO CONNECT REMOTE SEARCH RESPONSE	OUTPUT (Connect:Enterprise RESPONSE)	Connect:Enterprise RESPONSE TO LISTNAME REMOTE SEARCH
C$O49	3.3.3.1.4 ; 3.3.3.1.5	AUTO CONNECT REMOTE MODIFY	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise REQUEST TO MODIFY LISTNAME REMOTE
C$O50	3.3.3.1.1 ; 3.3.3.1.2	AUTO CONNECT LISTNAME MODIFY	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise REQUEST TO MODIFY LISTNAME
C$O51	3.3.3.1.4 ; 3.3.3.1.5	AUTO CONNECT REMOTE DETAIL REQUEST	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise REQUEST FOR LISTNAME REMOTE DETAIL
C$O52	3.3.3.1.4; 3.3.3.1.5	AUTO CONNECT REMOTE DETAIL RESPONSE	OUTPUT (Connect:Enterprise RESPONSE)	Connect:Enterprise RESPONSE TO LISTNAME REMOTE DETAIL
C$O53	3.3.3.1.6	AUTO CONNECT BSC LINE SEARCH	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES SEARCH REQUEST

Trailer ID:	Screen Number:	Screen Title:	Screen Type:	Screen Function:
C$O54	3.3.3.1.6	AUTO CONNECT BSC LINE SEARCH RESPONSE	OUTPUT (Connect:Enterprise RESPONSE)	Connect:Enterprise RESPONSE TO LINE SEARCH
C$O55	3.3.3.1.6	AUTO CONNECT BSC LINE UPDATE	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES UPDATE REQUEST
C$O56	3.3.3.1.7	AUTO CONNECT TIME SEARCH	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES SEARCH REQUEST
C$O561	3.3.6.1.1	POOLS RECORD LUNAME UPDATE	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES SEARCH REQUEST
C$O57	3.3.3.1.7	AUTO CONNECT TIME SEARCH RESPONSE	OUTPUT (Connect:Enterprise RESPONSE)	Connect:Enterprise RESPONSE TO TIME SEARCH
C$O571	3.3.6.1.1	POOL RECORD LUNAME UPDATE	OUTPUT (Connect:Enterprise RESPONSE)	Connect:Enterprise RESPONSE TO POOL
C$O58	3.3.3.1.7	AUTO CONNECT TIME UPDATE	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES UPDATE REQUEST
C$O581	3.3.6.1.1	POOLS RECORD LUNAME UPDATE	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise REQUEST TO UPDATE POOL LUNAME
C$O59	3.3.3.1.1 OR 3.3.3.1.2 (AFTER 1=UPDATE ON 3.3.3.1)	CONNECT RECORD BSC/SNA PARAMETER UPDATE	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES COPY REMOTE REQUEST
C$O60	3.3.3.1.1 OR 3.3.3.1.2 - UPDATE (PF3)	CONNECT RECORD BSC/SNA PARAMETER UPDATE	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise UPDATES
C$O61	3.3.3.1.1 OR 3.3.3.1.2 (AFTER 2=COPY ON 3.3.3.1)	CONNECT RECORD BSC/SNA PARAMETER UPDATE	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES COPY REQUEST - COPY A

Trailer ID:	Screen Number:	Screen Title:	Screen Type:	Screen Function:
C$O62	3.3.3.1.1 OR 3.3.3.1.2 - CANCEL (PF12) OR ANY	CONNECT RECORD BSC/SNA PARAMETER UPDATE	INPUT (Connect:Enterprise REQUEST)	ABANDONS "*CONNECT" UPDATE
C$O63	3.3.3.1 (DELETE ONLY)	AUTO CONNECT SELECTION LIST	INPUT (Connect:Enterprise REQUEST)	LIST A/C LISTNAMES FOR MODIFY, COPY OR DELETE
C$O64	3.2.2.2	QUEUED A/C SUMMARY DISPLAY	QUEUE MODIFICATION INPUT (Connect:Enterprise REQUEST)	MODIFY QUEUED A/C ELEMENTS
C$O66	3.1.8.1	Connect:Enterprise FILES DISPLAY	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES LISTA REQUEST
C$O67	3.1.8.1	Connect:Enterprise FILES DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY SYSTEM FILES AND STATUS
C$O68	3.1.10	ALLOCATE FILE REQUEST	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES $$ALLOC REQUEST
C$O69	3.1.11	DEALLOCATE FILE REQUEST	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES $$DALLOC REQUEST
C$O70	3.1.9.1	FILE SPACE ALLOCATION DISPLAY REQUEST	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES SPACE REQUEST
C$O71	3.1.8.1	FILE SPACE ALLOCATION DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY SYSTEM FILES SPACE ALLOCATIONS
C$O71X	3.1.8.1	FILE SPACE ALLOCATION DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY SYSTEM FILES SPACE ALLOCATIONS (Extended VSAM support)
C$O72	3.3.7	CALENDAR RECORD SELECTION REQUEST	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES CALENDAR DISPLAY
C$O73	3.3.7.1	CALENDAR SELECTION LIST DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	Connect:Enterprise RESPONSE TO CALENDAR DISPLAY

Trailer ID:	Screen Number:	Screen Title:	Screen Type:	Screen Function:
C$O74	3.3.7.1	CALENDAR RECORD SELECTION LIST	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise REQUEST TO DELETE CALENDAR
C$O75	3.3.7.1.1	CALENDAR RECORD UPDATE	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise RECEIVES SEARCH REQUEST
C$O76	3.3.7.1.1	CALENDAR RECORD UPDATE	OUTPUT (Connect:Enterprise RESPONSE)	Connect:Enterprise RESPONSE TO CALENDAR SEARCH
C$O77	3.3.7.1.1	CALENDAR RECORD UPDATE	INPUT (Connect:Enterprise REQUEST)	Connect:Enterprise REQUEST TO UPDATE CALENDAR
C$R20	GENERIC	NONE	INPUT/OUTPUT (Connect:Enterprise REQUEST & RESPONSE)	USER API "REQUEST" REQUEST AND RESPONSE
C$U03	2.1.1	AUTO CONNECT SUMMARY REQUEST	INPUT (Connect:Enterprise REQUEST)	REQUEST AUTO CONNECT SUMMARY HISTORY INFORMATION
C$U04	2.1.1.1	AUTO CONNECT SUMMARY SELECTION LIST DATA	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY SUMMARY RECORD DATA
C$U06	2.1.2	AUTO CONNECT DETAIL REQUEST	INPUT (Connect:Enterprise REQUEST)	REQUEST AUTO CONNECT DETAIL HISTORY INFORMATION
C$U061	2.1.5	QUEUED AUTO CONNECT REQUEST	INPUT (Connect:Enterprise REQUEST)	REQUEST QUEUED AUTO CONNECT HISTORY INFO
C$U07	2.1.2.1	AUTO CONNECT DETAIL SELECTION LIST DATA	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY DETAIL RECORD DATA
C$U071	2.1.5.1	QUEUED AUTO CONNECT DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY DETAIL RECORD DATA
C$U09	2.1.3	REMOTE CONNECT SUMMARY REPORT	INPUT (Connect:Enterprise REQUEST)	REQUEST REMOTE CONNECT SUMMARY HISTORY INFORMATION

Trailer ID:	Screen Number:	Screen Title:	Screen Type:	Screen Function:
C$U10	2.1.3.1	REMOTE CONNECT SUMMARY SELECTION LIST DATA	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY SUMMARY RECORD DATA
C$U12	2.1.4	REMOTE CONNECT DETAIL REQUEST	INPUT (Connect:Enterprise REQUEST)	REQUEST REMOTE CONNECT DETAIL HISTORY INFORMATION
C$U13	2.1.4.1	REMOTE CONNECT DETAIL DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY DETAIL RECORD DATA
C$U15	2.2.1	BATCH QUEUE DIRECTORY REQUEST	INPUT (Connect:Enterprise REQUEST)	REQUEST BATCH QUEUE DIRECTORY LIST
C$U16	2.2.1.1	BATCH QUEUE DIRECTORY LIST	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY BATCH QUEUE DIRECTORY LIST
C$U18	2.2.2	BATCH UTILIZATION REQUEST	INPUT (Connect:Enterprise REQUEST)	REQUEST BATCH UTILIZATION STATISTICS
C$U19	2.2.2	BATCH UTILIZATION LIST	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY BATCH UTILIZATION STATISTICS
C$U22	2.2.1.2	SELECT ONE BATCH FILE TO BROWSE REQUEST (CICS ONLY)	INPUT (Connect:Enterprise REQUEST)	REQUEST TO BROWSE THE DATA FROM ONE BATCH FILE.
C$U25	2.2.4	BATCH STATUS FLAGS REQUEST	INPUT (Connect:Enterprise REQUEST)	REQUEST FOR BATCH STATUS FLAGS
C$U26	2.2.4.1	BATCH STATUS FLAGS LIST	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY BATCH STATUS FLAGS FOR BROWSING
C$U28	2.2.1.3 OR 2.2.1.1 (MASS CHANGE)	BATCH STATUS FLAGS REQUEST	UPDATE (Connect:Enterprise REQUEST)	BATCH STATUS FLAGS UPDATE
C$U29	2.4.1.0.1, 2.4.1.0.2 OR 2.4.1.0.3 (ADD)	BATCH SUBMISSION (ADD/EXTRACT /GENERIC)	OUTPUT (Connect:Enterprise RESPONSE)	Connect:Enterprise RECEIVES BATCH FOR SUBMISSION

Trailer ID:	Screen Number:	Screen Title:	Screen Type:	Screen Function:
C$U31		SELECT ONE BATCH FILE TO BROWSE REQUEST	INPUT (Connect:Enterprise REQUEST)	REQUEST TO BROWSE THE DATA FROM ONE BATCH FILE.
C$U32		BROWSE BATCH BATCH FILE DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY BATCH FILE DISPLAY FOR BROWSING
C$U33	2.2.1.1	BATCH DETAIL REQUEST	INPUT (Connect:Enterprise REQUEST)	REQUEST FOR BATCH DETAIL INFORMATION
C$U34	2.2.D.1 2.2.D.2 2.2.D.3 2.2.D.4	BATCH DETAIL DISPLAY	OUTPUT (Connect:Enterprise RESPONSE)	DISPLAY BATCH DETAIL INFORMATION
C$W00		OUTPUT (Connect: Enterprise RESPONSE)	CICS WAKE UP INITIATE REQUEST	NONE

Glossary

A

ACQUEUE

Specifies the disposition of an Auto Connect session that is unable to be initiated because there is no BSC line, SNA session, or FTP thread available or the Auto Connect session is currently active. When the parameter ACQUEUE=YES is specified, the Auto Connect session is queued and initiation is attempted at a later time. Otherwise, the Auto Connect session is terminated with an error condition.

ADD Utility

A set of instructions used to submit the Connect:Enterprise ADD utility. The ADD utility is used to add batches to the VSAM batch files for access by the remote sites.

APPL (Application)

See *VTAM Application Program*.

Application Agent

A Connect:Enterprise interface that allows the customization of Connect:Enterprise execution. Each application agent is driven by a user-defined set of rules. The rules can display system console messages, issue system console commands, execute programs, and submit jobs. Connect:Enterprise supports the following application agents: Console, End Of Batch, Logging, Scheduler, and Wake Up Terminate.

Auto Connect

A Connect:Enterprise feature that allows host-initiated data communications to one or more remote sites. The host and remote sites may be connected using SNA, FTP, bisync manual dial, auto dial, or nonswitched lines. The Auto Connect session may be fully automated by time of day, or controlled with the $$CONNECT console command. Full reporting of Auto Connect activity is available.

Auto Dial

Refers to the capability of the host computer to automatically dial the remote site to establish a connection on a switched line. The Auto Dial feature is usually generated for the Transmission Control Unit or front-end processor of the host site on a line-by-line basis.

B

Batch

A set of related data collected by or added to Connect:Enterprise and maintained on the VSAM Batch Files at the host.

Batch Number

A unique 7-digit number assigned internally by Connect:Enterprise to each individual batch on the VSAM Batch Files. The number may be obtained by the $$DIRECTORY function or the offline utilities LIST function.

Batch Queue

See *VBQ (VSAM Batch Queue)*.

Batch Security

Optional Connect:Enterprise method of providing security for remote site access to the system. Mailbox IDs are assigned to remote sites and defined as valid at the host site. If Batch Security is used, remote sites must supply a valid ID as part of the $$ commands that access the Connect:Enterprise data files. (Formerly called ID Validation.)

Batch Status

A set of flags maintained for each batch on the VSAM Batch Files. The Batch Status flags are displayed in the LIST offline utility report or the $$DIRECTORY output data. Some of the Batch Status indicators are incomplete batch, deleted batch, batch transmitted to remote site, and batch extracted at the host site.

Batch Type

Used to indicate which batches to recall from Connect:Enterprise. Types include batches containing data received from remote sites and batches containing data to be transmitted.

Blank Compression

A method of replacing strings of contiguous blanks with control characters indicating the number of blanks removed. Commonly used to shorten the amount of data sent over telecommunications lines. Connect:Enterprise uses standard 3780 blank compression techniques on BSC lines and standard SNA blank and character compression on SNA sessions.

Blank Truncation

A method of dropping trailing blanks from the end of fixed length data records before sending the data over telecommunications lines. Used by Connect:Enterprise as an option to shorten the amount of data sent over telecommunications lines.

BSC (Binary Synchronous)

A standard telecommunications line protocol used to transmit blocks of data over telecommunications lines between host computers and remote sites. Binary Synchronous (also known as bisync) allows a faster transmission rate than a start/stop protocol, because its ratio of data bits to checking bits is higher. This line protocol is used by Connect:Enterprise.

BTAM (Basic Telecommunications Access Method)

A standard IBM access method used by Connect:Enterprise to read and write data over telecommunications lines to a variety of terminals and devices.

BTAM ID Verification

An optional BTAM feature that enables the exchange and verification of host site and remote site IDs. Available on switched lines only, the feature provides added security in a Connect:Enterprise system. Both the host site and the remote site must be capable of implementing the option. Connect:Enterprise allows the host site ID to be sent, the remote site ID to be received, or both IDs to be exchanged.

C

Clear Control Channel (CCC)

A command that enables Connect:Enterprise to negotiate a clear-text control channel after the user ID and password have been transmitted in encrypted format. The control channel remains in clear-text until the connection ends. All data and objects transferred between the client and server remain encrypted. Both ends of the connection must support the use of this command.

Compression

See *Blank Compression*.

Connection ID

The CICS definition that describes the remote system in terms of Netname (APPLID). The connection ID is a local name (within the local CICS only) that is used to define the remote partner system (Connect:Enterprise).

Cross System Client Utility (CSCU)

A Connect:Enterprise utility that enables you to use a subset of the offline utilities to access the VSAM batch and log files from a remote logical partitioning (LPAR), unlike offline utilities which must run from the same LPAR as the Connect:Enterprise VSAM File Server. CSCU control and output is similar to the offline utilities.

D

Data Collection

The process in which Connect:Enterprise collects data from remote sites and stores it in the VSAM Batch Files. Data Collection means data is input from a remote site to Connect:Enterprise at the host computer.

Data Repository

The component that transmits and collects data from BSC, FTP, and SNA sites. The repository handles all session activity and accepts service requests from the console, the user API, the ISPF interface, the CICS interface, and the Connect:Enterprise FTP server.

Data Transmission

The process in which Connect:Enterprise transmits data from the VSAM Batch Files to remote sites. Data transmission means data is output from Connect:Enterprise at the host computer to the remote site.

Directory

A formatted listing of control information for batches on the Connect:Enterprise VSAM Batch Files. It is obtained from the $$DIRECTORY command.

Disconnect Interval

The number of seconds a session may be inactive before forcing session termination. This may differ for each remote site defined to Connect:Enterprise. This safety feature, which is implemented using the DISCINTV parameter, is used to reduce the use of resources by remote sites that have no current activity and to prevent an Auto Connect session from suspending if a remote site does not respond.

EXTRACT Utility Model

A set of JCL statements and parameter (specification) data submitted by Connect:Enterprise CICS or ISPF interface to initiate execution of the Connect:Enterprise EXTRACT utility. The EXTRACT utility is used to retrieve batches from VSAM batch files to a sequential output file.

F

FMH (Function Management Header)

A standard SNA feature that allows a data stream to be sent to a specific destination and controls the way the data is presented at the destination. Connect:Enterprise supports FMH Type 1, a 6-character field sent at the start and the end of a data stream. This FMH selects the media used for the data, marks the beginning and end of a Connect:Enterprise batch, and further describes the format of the data.

FTP (File Transfer Protocol)

An international standard for reading and writing files across a TCP/IP network.

FTP Server

The capability of Connect:Enterprise to function as an FTP server. This enables remote FTP client sites to access, retrieve, and send data to the Connect:Enterprise batch queues through standard FTP commands.

G

GSKKYMAN

An IBM utility that is used to create and maintain the SSL key database.

H

Host

The main processing computer where Connect:Enterprise is running and where you send your data batches. Also referred to as the host site or host computer.

I

IRS (Inter-Record Separator)

A special character used to separate multiple records in a block of data being transmitted over a telecommunications line. Connect:Enterprise allows either X'1E' or X'1F' as the inter-record separator on BSC lines, and allows only X'1E' for SNA sessions. Also referred to as an IRS.

J

Job Entry Subsystem (JES)

A system facility for spooling, job queuing, and managing job-related data.

L

Leased Line

Refers to telecommunications lines on which connection is not established through a switched network. Connect:Enterprise Leased Line support is point-to-point and therefore allows data to be exchanged only between the host site and a single remote site. Leased Multipoint lines are not supported by BSC connections in Connect:Enterprise.

Line ID

Uniquely identifies a BSC line that is accessed during Auto and Remote Connects. This is a BSC-only entry generated by a nonswitched M$LINE or M$LINEX macro in the User Assembly.

List Name

The Auto Connect List Name defined in the Connect:Enterprise ODF.

Log Facility

A Connect:Enterprise feature that provides file logging and full reporting for remote-initiated transactions. An additional option provides host system console log messages both for host-initiated and for remote-initiated connections and disconnections.

LOGOFF

The process of ending a remote site session with a host site program such as Connect:Enterprise. A LOGOFF may be a text command or a control function from a remote device.

LOGON

The process of establishing a session between a remote site and a host site program such as Connect:Enterprise. A LOGON may be automatic after a connection is established, or may be entered as a text command or a control function. In Connect:Enterprise, either the remote site or the host site may attempt to initiate the LOGON process.

Logon Mode Table

A table defined to VTAM containing a set of entries that provide session parameters, or the rules for controlling SNA communications. The LOGON that attempts to establish a session causes access to this table to obtain the session rules.

LOGON Security

An optional Connect:Enterprise/SNA method of providing security during a remote site's attempt to LOGON to Connect:Enterprise. The LUNAME (assigned to the remote site as part of the VTAM definition process) is provided to and validated by Connect:Enterprise when a LOGON is attempted.

LU (Logical Unit)

A logical unit provides the port for user access to an SNA network. Each remote device that can establish a session with Connect:Enterprise is a logical unit.

LU1RJE (LU Type 1 RJE)

A device emulating 3770, or a similar device or software package that uses Logical Unit Type 1 protocols and is used primarily for data transfer or RJE (Remote Job Entry) purposes. The devices typically have multiple I/O devices, such as printers, card readers, and storage devices. An operator console for messages or interactive use is often present.

M

Mailbox ID

The 1–8 character ID which defines batches in the VSAM Batch Files.

Mailbox Name

The 8-character symbolic name used to identify individual Connect:Enterprise systems to the user interface.

Mailbox Password

A security password used to control access to Connect:Enterprise systems.

Mailbox User ID

An 8-character field used to identify each user to Connect:Enterprise. In order for a user to access a Connect:Enterprise system, the User ID must be defined and assigned. The CICS and ISPF Interface panel displays the current user in the upper right corner.

Manual Dial

Refers to the method the host site uses to dial remote sites to establish a connection on a switched line. With Manual Dial, an operator at the host site must manually dial the telephone number of the remote site if the connection is initiated by the host site.

If the connection is initiated by the remote site, the manual dialing at the host is not used.

Media

An input/output device on a terminal, such as a printer, card reader, card punch, keyboard, display, or diskette. Commonly available on LU Type 1 RJE terminals, and supported by Connect:Enterprise/SNA.

MLU (Multiple Logical Unit)

A terminal designed to allow the operation of more than one session between a remote terminal and a host site such as Connect:Enterprise. A single terminal may actually appear as multiple devices, and may have concurrent inbound and outbound data streams active for each. Some 3770-type devices have this capability. Connect:Enterprise supports up to six MLU sessions per remote site.

N

NCP (Network Control Program)

The Network Control Program, generated by host site personnel, that controls the operations of a communications controller such as a 37x5.

Non-Switched Line

A telecommunications line on which connection is not established through a switched network. Sometimes referred to as a Leased Line.

NPSI (Network Control Program Packet Switching Interface)

An IBM licensed program that allows SNA users to communicate over packet switching data networks that have interfaces complying with CCITT Recommendation X.25. It allows SNA programs to communicate with SNA or non-SNA equipment over such networks.

O

(ODF) Options Definition File

A file containing Connect:Enterprise control records and keyword parameters that specify options in effect for the current execution of online Connect:Enterprise. The file contains options that control security, password, Auto Dial telephone numbers, SIGNON records, Auto Connect, SNA sites, and other system options.

Offline Utilities

The Connect:Enterprise utilities used to access and maintain the data batches on the VSAM Batch Files. The offline utilities allow you to LIST control information for batches, ADD batches, EXTRACT batches, DELETE batches, ERASE batches, alter batch status flags (STATFLG), MOVE batches from one VBQ to another, and REPORT on session activity.

P

Password

See *Mailbox Password*.

PLU (Primary Logical Unit)

In a particular session between two LUs, one LU adheres to a set of SNA-defined primary protocols and is known as the primary logical unit (PLU) for that session. The other LU adheres to a set of secondary protocols and is known as the secondary logical unit (SLU) for that session. More than one session can exist between two LUs. Multiple concurrent sessions between the same two LUs are referred to as parallel sessions. Not all LUs have parallel session capability.

Point-to-Point Line

A telecommunications line connection that allows data exchange between two points on the connection, usually the host site and a remote site. When a dialed connection is established on a switched network, the connection is considered point-to-point. Leased lines where the remote site is a single station are also considered point-to-point.

R

RDW (Record Descriptor Word)

A 4-byte field used to define the length of variable length records within a data file. For batch data coming into Connect:Enterprise (ADD), the RDW may be removed or retained. For batch data sent from Connect:Enterprise (REQUEST) the RDW may be created or not created.

Remote Name

A 1–8 character name assigned to identify a remote site that may be contacted by the host site during an Auto Connect session. Also used to identify every remote site that can establish a session with Connect:Enterprise.

Remote Site

Any terminal, computer, or software that can connect with Connect:Enterprise in the host computer.

REXX (Restructured Extended Executor) Language

A general-purpose, procedural language for scripting end-user programs designed for IBM systems.

RFC (Request for Comments)

One of a series, begun in 1969, of numbered Internet informational documents and standards widely followed by commercial software and freeware in the Internet and UNIX communities.

S

Session

A logical connection between Connect:Enterprise at the host site and another logical unit, such as a 3770 device. When a LOGON is completed between Connect:Enterprise and a remote site, they are said to be in session.

SIGNON

A special format data record sent by some remote BSC terminals designed to communicate with RJE software (such as JES or VSE POWER) in the host computer. The SIGNON record may be required by Connect:Enterprise provided Connect:Enterprise has been configured to do so when installed. The SIGNON format(s) used must also be specified at installation. A SIGNON is not required and not supported for SNA remote sites.

SLU (Secondary Logical Unit)

See *PLU (Primary Logical Unit)*.

SNA (Systems Network Architecture)

A set of rules, procedures, and structures for a communications network.

Socket Number

A two way connection identified by the unique combination of IP addresses and port numbers in a given connection. For example, the following combination illustrates the unique ID representing a complete socket: Client IPAddress/Port Number - Server IPAddress/Port Number.

SPLITCOUNT

Specifies a 1–4 digit numeric count of records to be contained in an added batch, allowing you to split a large sequential input file into several smaller batches with the same batch identifiers. Sequential input records are read and added to the output batch until the SPLITCOUNT limit is reached. Connect:Enterprise then closes out the batch and begins a new batch with the same identifiers.

SSL (Secure Sockets Layer)

A protocol for transmitting private documents over the Internet. SSL uses a private key to encrypt data that is transferred over the SSL connection.

Status Codes

The status flag indicators for a batch. Codes include the following: D, deleted; T, transmitted; R, Requestable; E, Extracted; M, Multxmit (for a list of these codes, see information on VSAM Batch Status Flags in the *Connect:Enterprise for z/OS User's Guide*).

Switched Line

A telecommunications line on which connection is established over a switched (dialup) telephone line.

T

TLS (Transport Layer Security)

A protocol based on SSL 3.0 protocol specification and designed to provide privacy and data integrity between two communicating applications.

TRACE

In Connect:Enterprise, the capability to create a snapshot dump of internal Connect:Enterprise control information for communications activity, User Exit calls, or VSAM Batch Files access.

Transparency

A method of transmitting data over a telecommunications line wherein special line control characters embedded in the data are transparent and do not function in their normal capacity as line control characters. Transparency is used when non-text data (such as object modules or other binary data) must be sent over telecommunications lines. Connect:Enterprise supports both BSC transparency and SNA transparency.

Truncation

See *Blank Truncation*.

$TURNLINE$

An optional feature in Connect:Enterprise that provides for a limited conversational mode transmission. When a $TURNLINE$ record is encountered in data being sent to a remote site, the sender temporarily

stops sending and issues the proper BSC protocol to turn around the line and begin receiving. After all data is received, sending resumes with the record following $TURNLINE$.

U

User

See *Mailbox User ID*.

User Assembly

A series of macros used to define a network of BSC lines to be used by Connect:Enterprise. The macros are generated by each user to define their requirements and input to the Assembler to create a module for use by Connect:Enterprise BSC connections. A User Assembly is not required by SNA connections.

User Batch ID

A 1–64 character free-form batch identifier used to describe the contents of a batch of data on the Connect:Enterprise VSAM Batch Files.

User Exits

A user-written program called by online Connect:Enterprise, offline utilities, and the CICS interface at appropriate times during the processing of a transaction. The user-supplied program can thereby alter the standard processing done by Connect:Enterprise. User Exits may be supplied to examine all input data from a remote site, to examine output data to a remote site, to provide unique security processing, or to examine and alter data in Connect:Enterprise $$ commands. No alteration of data is possible by a user exit in the offline utilities and the CICS interface processing.

USS Table

A table defined to VTAM that provides conversion of character-coded LOGON or LOGOFF to field-formatted LOGON or LOGOFF. You may need to provide this table to VTAM to allow a remote site to establish and terminate SNA sessions with Connect:Enterprise.

V

VBQ (VSAM Batch Queue)

The Connect:Enterprise data set used for storing batches of data collected from remote sites during online Connect:Enterprise. These batches may be available for transmission to remote sites, and are always available for extraction at the host site. The VSAM Batch Queue may be defined as a single VSAM cluster or up to 20 VSAM clusters that are processed as a single repository for batch data. The VSAM Batch Queue contains multiple individual batches of data which can be accessed by their Mailbox ID.

VBQ Blocking

A Connect:Enterprise feature that blocks multiple records or collection buffers into a single VBQ record for transmission. This improves transmission performance by reducing the disk I/O overhead.

VCF (VSAM Control File)

The Connect:Enterprise data set that contains control information for batches stored on the VSAM Batch Queue.

VLF (VSAM Log File)

The Connect:Enterprise data set that contains logged information on the progress of a Connect:Enterprise execution.

VPF (VSAM Pointer File)

The Connect:Enterprise data set that contains control information for every file defined in the Connect:Enterprise system and locator information for every existing batch.

VSAM (Virtual Storage Access Method)

A standard IBM access method for creating and maintaining data sets at the host. Used by Connect:Enterprise for the VSAM Batch Files.

VSAM Batch Files

A term used for the group of up to 24 files used by the Connect:Enterprise system for storing and maintaining data. The VSAM Batch Files consist of the VSAM Control File, the VSAM Pointer File, the VSAM Batch Queue Files (up to 20), and the VSAM Log Files (up to 2).

VTAM (Virtual Telecommunications Access Method)

An SNA access method used by Connect:Enterprise to receive and send data to a variety of SNA devices or application programs.

VTAM Application Program

A program, such as Connect:Enterprise, that is defined to VTAM and can establish sessions with SNA devices or other VTAM application programs.

X

Xmit once

Specifies that the batch cannot be extracted and that it can be transmitted only one time. After a successful transmit, the batch is permanently locked.

Index

Symbols

A

B

C

D

E

F

I

L

M

N

O

P

Q

R

S

T

U

V

W

X